CAPITOLUL 7

PROIECTAREA UNEI INTERFEȚE DE INTRARE/IEȘIRE

În acest capitol se prezintă interfața serială SPI (*Serial Peripheral Interface*) și se descrie proiectarea unui controler pentru această interfață. Controlerul SPI va fi utilizat pentru transmiterea unor comenzi accelerometrului de pe o placă de dezvoltare pentru citirea unor registre ale accelerometrului.

7.1 Interfața SPI

7.1.1 Prezentare generală

Interfața SPI (*Serial Peripheral Interface*) a fost dezvoltată de firma Motorola, dar a fost adoptată ulterior de numeroși producători ca o interfață de comunicație între microcontrolere și diferite circuite periferice din sistemele înglobate. O interfață care este echivalentă funcțional cu SPI este *Microwire*, acest nume fiind utilizat de firma National Semiconductor. SPI este o interfață serială sincronă, spre deosebire de interfața serială UART, care este una asincronă. În cazul unei interfețe asincrone, nu se transmite un semnal de ceas împreună cu datele, iar receptorul utilizează un semnal de ceas generat local pentru eșantionarea liniei seriale la momente de timp corespunzătoare vitezei de comunicație care a fost stabilită pentru transmițător și receptor. Pot exista anumite diferențe între frecvențele ceasurilor utilizate de transmițător și de receptor, dar se realizează resincronizarea ceasului receptorului la recepția bitului de START care precedă fiecare caracter. În cazul unei interfețe sincrone, datele transmise sunt însoțite de un semnal de ceas al cărui front crescător sau descrescător se utilizează de către receptor pentru eșantionarea corectă a liniei seriale. Deoarece se transmite un semnal separat de ceas, specificarea unei anumite viteze de comunicație

nu este importantă, deși diferitele circuite pot avea viteze maxime diferite la care pot funcționa.

Interfața SPI funcționează în modul de comunicație duplex, în care datele se transmit în ambele direcții simultan. Se utilizează patru linii de comunicație, spre deosebire de alte interfețe sau magistrale seriale, ca I^2C (*Inter-Integrated Circuits*) sau SMBus (*System Management Bus*), care utilizează doar două linii de comunicație. Posibilitatea comunicației duplex este utilizată rareori în practică. De exemplu, unele dispozitive, cum sunt controlerele afișajelor cu cristale lichide, au numai rol de receptor și nu transmit niciodată informații de stare. Alte dispozitive recepționează o comandă de citire, după care transmit datele solicitate, fără a recepționa în același timp alte comenzi. Deci, deși comunicația este duplex, în multe cazuri datele recepționate nu sunt luate în considerare sau sunt transmise date care nu vor fi utile. Există însă dispozitive care pot recepționa și transmite date în mod simultan, de exemplu, senzori care acceptă o nouă comandă de citire în timp ce returnează datele solicitate de o comandă precedentă.

Arhitectura interfeței SPI este de tip *master-slave*, în care poate exista un singur dispozitiv *master* și unul sau mai multe dispozitive *slave*. Toate transferurile sunt inițiate de către un dispozitiv *master*; acesta va selecta dispozitivul *slave* cu care se va realiza comunicația și va genera semnalul de ceas pentru sincronizare.

Interfața SPI este utilizată de numeroase dispozitive periferice, de exemplu, diferite tipuri de senzori, convertoare digital/analogice și analog/digitale, memorii *flash* și EEPROM, controlere de comunicație (Ethernet, USB, IEEE 802.11, IEEE 802.15.4), ceasuri de timp real, codificatoare/decodificatoare audio, controlere pentru afișaje cu cristale lichide, cartele de memorie SD (*Secure Digital*). Microcontrolerele conțin interfețe SPI care pot funcționa în mod *master* sau în mod *slave*. Unele circuite integrate utilizează interfața SPI pentru comunicația între diferite module interne.

7.1.2 Semnalele interfeței

Interfața SPI utilizează următoarele patru semnale:

- SCLK (*Serial Clock*);
- MOSI (*Master Output, Slave Input*);
- MISO (*Master Input, Slave Output*);
- SS (*Slave Select*).

Semnalul SCLK este semnalul de ceas generat de către dispozitivul *master*, care se utilizează pentru recepția și transmisia datelor de către dispozitivele *slave*. Dispozitivul *master* trebuie să cunoască numărul ciclurilor de ceas pe care trebuie să le genereze, astfel încât, dacă dispozitivul *slave* trebuie să returneze date, acesta să poată returna toate datele cerute. Semnalul MOSI se utilizează pentru transmisia datelor de la dispozitivul *master* la un dispozitiv *slave*, iar semnalul MISO se utilizează pentru transmisia datelor de la un dispozitiv *slave* la dispozitivul *master*. Semnalul SS este

activat de către dispozitivul *master* pentru a selecta dispozitivul *slave* care va participa la următorul transfer de date. În starea activă, acest semnal are nivelul logic 0. Atunci când nu se transferă date, semnalul SS este menținut la nivelul logic 1, ceea ce va deconecta dispozitivul sau dispozitivele *slave* de la interfața SPI.

Figura 7.1 ilustrează o configurație cu un singur dispozitiv *slave*.

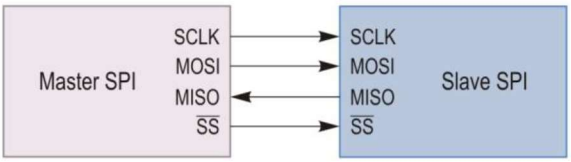

Figura 7.1. Configurație a interfeței SPI cu un singur dispozitiv *slave*.

Dacă există mai multe dispozitive *slave*, există două posibilități pentru selecția acestora. Prima posibilitate este de a se utiliza semnale de selecție separate pentru fiecare dispozitiv *slave*. Dispozitivul *master* va activa un singur semnal de selecție la un moment dat pentru a evita ca mai multe dispozitive *slave* să transmită date pe linia MISO. Figura 7.2 ilustrează o configurație cu trei dispozitive *slave* pentru care se utilizează semnale de selecție separate. Deoarece pinii MISO ai dispozitivelor *slave* sunt conectați împreună, acești pini trebuie să fie cu trei stări.

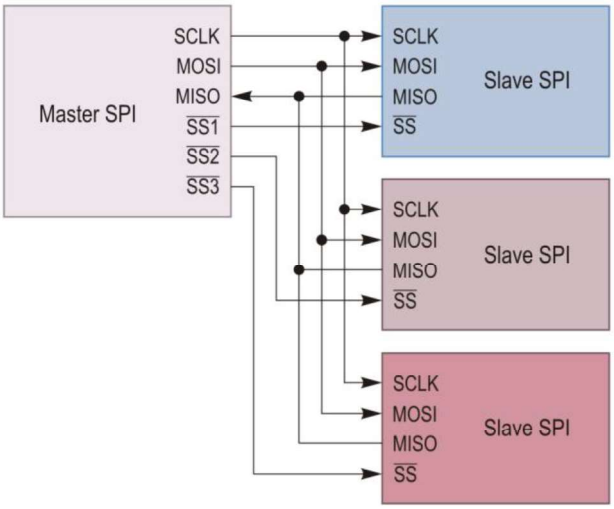

Figura 7.2. Configurație a interfeței SPI cu trei dispozitive *slave* selectate independent.

A doua posibilitate este de a se utiliza o configurație în lanț, ilustrată în figura 7.3 pentru un număr de trei dispozitive *slave*. În acest caz, este necesar un singur semnal de selecție pentru toate dispozitivele *slave*, în locul unor semnale de selecție separate. Pinul MISO al unui dispozitiv *slave* este conectat la pinul MOSI al urmă-

torului dispozitiv. Această configurație se utilizează, în mod tipic, pentru dispozitive *slave* care nu returnează date la dispozitivul *master*, cum sunt controlerele pentru afișaje, caz în care pinul MISO al dispozitivului *master* rămâne neconectat. Dacă trebuie returnate date la dispozitivul *master*, pinul MISO al ultimului dispozitiv *slave* va fi conectat la pinul MISO al dispozitivului *master*. Fiecare dispozitiv *slave* va transmite la al doilea impuls de ceas ceea ce a recepționat la primul impuls de ceas. Întregul lanț funcționează ca un registru de deplasare utilizat pentru comunicație. Aceasta permite accesul la un grup de intrări sau de ieșiri ale unui circuit prin interfața serială SPI.

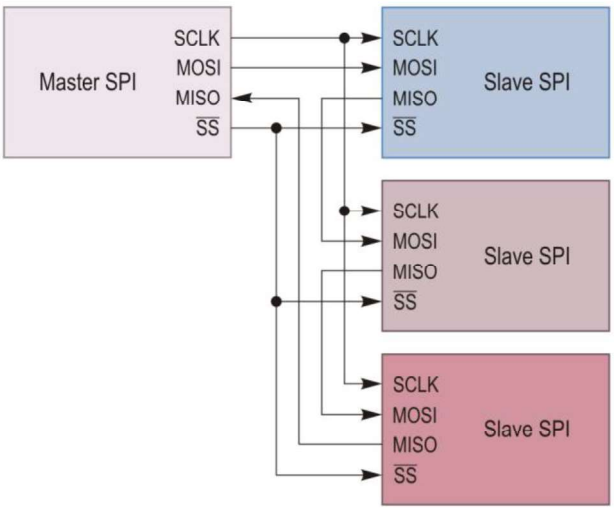

Figura 7.3. Configurație a interfeței SPI cu trei dispozitive *slave* conectate în lanț.

Diferiți producători pot utiliza nume alternative pentru semnalele circuitelor prevăzute cu interfața SPI. Exemple de nume alternative sunt următoarele:

- SCLK: SCK, CLK;
- MOSI: SDI (*Serial Data In*), DI, DIN, SI;
- MISO: SDO (*Serial Data Out*), DO, DOUT, SO;
- SS: nSS, CS, nCS, EN, STE (*Slave Transmit Enable*).

Atunci când se utilizează denumirile SDI/SDO, pinul SDO al dispozitivului *master* este conectat la pinul SDI al unui dispozitiv *slave*.

7.1.3 Transmisia și recepția datelor

Înainte de începerea comunicației cu un dispozitiv *slave*, dispozitivul *master* stabilește frecvența semnalului de ceas care se va utiliza. Această frecvență trebuie să fie mai mică decât frecvența maximă permisă pentru interfața SPI a dispozitivului

slave cu care se va realiza comunicația. În mod tipic, frecvența semnalului de ceas este cuprinsă între 1 MHz și 10 MHz. Apoi, dispozitivul *master* selectează dispozitivul *slave* prin activarea semnalului SS. Dacă este necesară o perioadă de așteptare, de exemplu, pentru terminarea unei conversii analog/digitale, dispozitivul *master* trebuie să aștepte un timp corespunzător înainte de a genera impulsuri de ceas.

În timpul fiecărui ciclu de ceas, dispozitivul *master* transmite un bit pe linia MOSI, iar dispozitivul *slave* citește un bit de pe aceeași linie. În același ciclu de ceas, dispozitivul *slave* transmite un bit pe linia MISO, iar dispozitivul *master* citește un bit de pe aceeași linie. Această comunicație bidirecțională este menținută și atunci când este necesară doar o comunicație unidirecțională, astfel încât nu toate datele transmise sau recepționate vor fi utile.

Pentru transmisia și recepția datelor, se utilizează două registre de deplasare, unul de către dispozitivul *master* și unul de către dispozitivul *slave*. Dimensiunea registrelor de deplasare depinde de dimensiunea cuvântului, care nu este fixată prin specificațiile SPI, ci poate fi aleasă în funcție de aplicația pentru care se utilizează interfața. Deși, de multe ori, dimensiunea cuvântului este de opt biți, există situații în care această dimensiune este de 12 biți (pentru numeroase convertoare digital/analogice sau analog/digitale) sau de 16 biți (pentru controlere ale afișajelor sensibile la atingere sau codificatoare/decodificatoare audio).

De obicei, primul bit transmis este bitul cel mai semnificativ al cuvântului, astfel încât cele două registre se vor deplasa cu o poziție spre stânga în fiecare ciclu de ceas. Un bit recepționat este depus în poziția cea mai puțin semnificativă a registrelor de deplasare, după cum se ilustrează în figura 7.4. Transmisia și recepția continuă pentru un număr variabil de cuvinte, număr care depinde de lungimea blocurilor de date transferate. La terminarea transferului, dispozitivul *master* oprește generarea impulsurilor de ceas și, de obicei, deselectează dispozitivul *slave*.

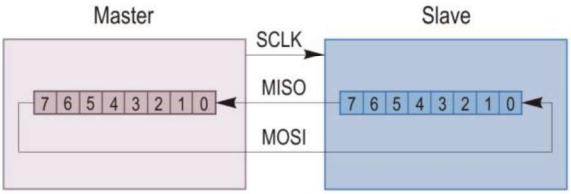

Figura 7.4. Utilizarea registrelor de deplasare pentru transmisia și recepția unor cuvinte de opt biți.

7.1.4 Polaritatea și faza semnalului de ceas

Pe lângă stabilirea frecvenței semnalului de ceas, dispozitivul *master* trebuie să configureze *polaritatea* și *faza* semnalului de ceas. Polaritatea semnalului de ceas, notată cu CPOL, se referă la nivelul logic inițial (0 sau 1) al semnalului de ceas. Faza semnalului de ceas, notată cu CPHA, se referă la frontul crescător sau descrescător pe care sunt citite și modificate datele. Posibilitățile pentru alegerea polarității și a fazei semnalului de ceas sunt ilustrate în figura 7.5.

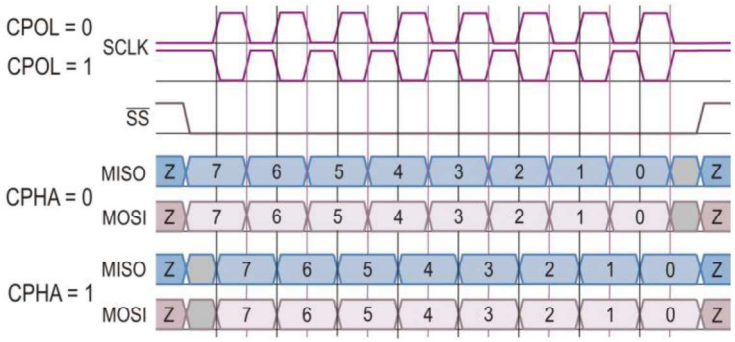

Figura 7.5. Polaritatea (CPOL) și faza (CPHA) semnalului de ceas.

Dacă polaritatea CPOL este 0, nivelul logic inițial (inactiv) al semnalului de ceas este 0, iar nivelul logic activ este 1. În acest caz, cele două posibilități pentru faza CPHA sunt următoarele:

- Dacă faza este 0, datele sunt citite pe frontul crescător al semnalului de ceas și sunt modificate pe frontul descrescător al semnalului de ceas.
- Dacă faza este 1, datele sunt citite pe frontul descrescător al semnalului de ceas și sunt modificate pe frontul crescător al semnalului de ceas.

Dacă polaritatea CPOL este 1, nivelul logic inițial (inactiv) al semnalului de ceas este 1, iar nivelul logic activ este 0. În acest caz, cele două posibilități pentru faza CPHA sunt următoarele:

- Dacă faza este 0, datele sunt citite pe frontul descrescător al semnalului de ceas și sunt modificate pe frontul crescător al semnalului de ceas.
- Dacă faza este 1, datele sunt citite pe frontul crescător al semnalului de ceas și sunt modificate pe frontul descrescător al semnalului de ceas.

Deci, după cum rezultă din figura 7.5, dacă CPHA = 0, datele sunt citite pe primul front al semnalului de ceas, iar dacă CPHA = 1, datele sunt citite pe al doilea front al semnalului de ceas, indiferent dacă acest front este crescător sau descrescător. Va exista întotdeauna o întârziere de o jumătate de ciclu de ceas între momentul în care datele au fost transmise la ieșire de către un dispozitiv și momentul în care ele vor fi citite de către celălalt dispozitiv, întârziere necesară pentru stabilizarea datelor. Se poate observa că dacă CPHA = 0, datele trebuie să fie stabile cu o jumătate de ciclu de ceas înaintea primului ciclu de ceas. Nivelul logic inițial al semnalului de ceas trebuie să fie stabil înainte de activarea semnalului de selecție SS.

Combinația dintre polaritatea și faza semnalului de ceas este numită *mod* al comunicației SPI. Specificațiile diferitelor dispozitive periferice indică modurile în care

se poate realiza comunicația cu acele dispozitive. În general, pentru codificarea modului se utilizează convenția indicată în tabelul 7.1.

Tabelul 7.1. Convenția utilizată pentru codificarea modului de comunicație SPI.

Mod	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

7.2 Proiectarea unui controler pentru interfața SPI

În această secțiune se descrie proiectarea unui controler cu rol de *master* pentru interfața SPI. Controlerul proiectat va avea următoarele caracteristici principale:

- Frecvența semnalului de ceas SCLK poate fi specificată printr-un generic.
- Dimensiunea cuvântului poate fi specificată printr-un generic.
- Primii biți transmiși și recepționați vor fi biții cei mai semnificativi ai cuvintelor.
- Controlerul va permite comunicația prin interfața SPI în modul 0, ceea ce înseamnă că polaritatea semnalului de ceas (CPOL) și faza semnalului de ceas (CPHA) vor fi ambele 0.

7.2.1 Schema bloc simplificată a controlerului SPI

Controlerul SPI proiectat va avea o interfață simplă cu exteriorul. Pentru transmisia unui cuvânt pe linia MOSI, utilizatorul controlerului va trebui să aplice cuvântul respectiv la intrarea *TxData* a controlerului, să activeze semnalul *Start* pentru a indica începerea transmisiei cuvântului, iar apoi să monitorizeze starea semnalului *TxRdy* pentru a detecta terminarea transmisiei cuvântului. Pentru recepția unui cuvânt pe linia MISO, utilizatorul controlerului va trebui să activeze același semnal *Start* pentru a indica începerea recepției cuvântului, iar apoi să monitorizeze starea semnalului *RxRdy* pentru a detecta terminarea recepției cuvântului. Atunci când s-a detectat activarea semnalului *RxRdy*, cuvântul recepționat poate fi preluat de la ieșirea *RxData* a controlerului. Deci, activarea semnalului *Start* reprezintă pentru controlerul SPI o comandă atât pentru începerea transmisiei unui cuvânt, cât și pentru începerea recepției unui cuvânt, comunicația fiind duplex.

Controlerul SPI va genera semnalul de ceas SCLK necesar interfeței seriale prin divizarea frecvenței semnalului de ceas al sistemului. Pentru transmisia și recepția serială, controlerul va conține un registru de deplasare. Ieșirea serială a acestui registru va fi conectată la linia MOSI a interfeței, iar intrarea serială a registrului va fi conectată la linia MISO a interfeței. Registrul va fi deplasat la stânga cu o poziție în

fiecare ciclu de transmisie/recepție, deoarece primii biți transmiși și recepționați vor fi biții cei mai semnificativi ai cuvântului. Controlerul va mai conține un registru pentru memorarea cuvântului care trebuie transmis și o unitate de control pentru generarea semnalelor de comandă, a semnalelor de stare (*TxRdy*, *RxRdy*) și a semnalului de selecție SS al interfeței.

Figura 7.6 prezintă schema bloc simplificată a controlerului SPI. Modulul Gen_Sclk generează semnalul de ceas SCLK, care va fi transmis la ieșire prin intermediul unui buffer. Registrul TxData_reg memorează cuvântul aplicat la intrarea *TxData* a controlerului atunci când semnalul *Start* este activat. Registrul de deplasare TxRx_reg are o intrare paralelă pentru încărcarea unui cuvânt care trebuie transmis, o ieșire paralelă pentru accesul la un cuvânt recepționat, o intrare serială pentru memorarea unui bit recepționat pe linia MISO și o ieșire serială pentru transmisia unui bit pe linia MOSI. Unitatea de control generează semnalul de validare al bufferului pentru semnalul de ceas SCLK, semnalul de selecție SS al dispozitivului *slave* și semnalele de stare *TxRdy* și *RxRdy*. Semnalul de stare *TxRdy* se activează la terminarea transmisiei unui cuvânt, indicând faptul că se poate aplica un nou cuvânt la intrarea *TxData* și se poate activa semnalul *Start*. Semnalul de stare *RxRdy* se activează la terminarea recepției unui cuvânt, indicând faptul că datele de la ieșirea *RxData* sunt valide.

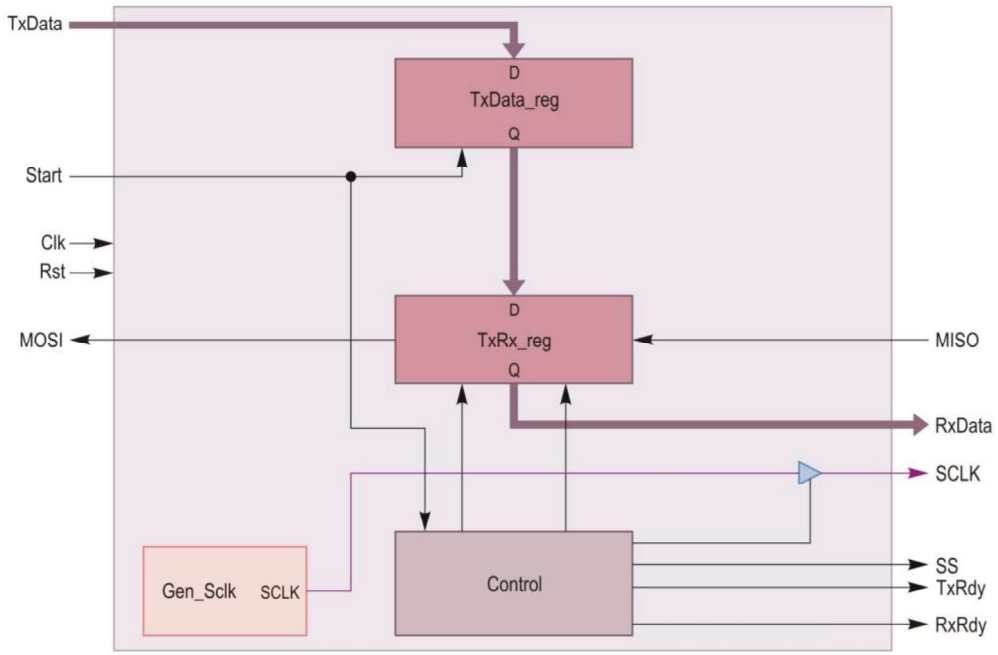

Figura 7.6. Schema bloc simplificată a controlerului SPI.

7.2.2 Schema bloc detaliată a controlerului SPI

Pentru proiectarea corectă a controlerului SPI, trebuie să se țină cont de unele detalii importante ale funcționării interfeței SPI. În primul rând, în cazul interfeței SPI, datele seriale sunt citite și sunt transmise pe fronturi diferite ale semnalului de ceas SCLK, în funcție de polaritatea și faza semnalului de ceas. Pentru varianta proiectată a controlerului, la care polaritatea și faza semnalului de ceas sunt ambele 0, datele sunt citite pe frontul crescător al semnalului de ceas SCLK și sunt transmise pe frontul descrescător al acestui semnal de ceas. De aceea, linia MISO nu se poate conecta direct la intrarea serială a registrului de deplasare TxRx_reg, deoarece acest registru va fi deplasat pe frontul descrescător al semnalului de ceas SCLK, atunci când se depune pe linia MOSI un nou bit pentru a fi transmis. Linia MISO poate fi conectată la un bistabil care va fi înscris pe frontul crescător al semnalului de ceas SCLK, ieșirea bistabilului fiind conectată la intrarea serială a registrului de deplasare, astfel încât bitul citit va fi înscris în poziția 0 a registrului de deplasare pe frontul descrescător al semnalului de ceas SCLK.

Un alt aspect important este că diferitele componente secvențiale ale controlerului SPI trebuie să funcționeze la frecvențe diferite. Registrul de date TxData_reg din figura 7.6 trebuie să funcționeze la frecvența ridicată a semnalului de ceas al sistemului (de exemplu, de 100 MHz în cazul multor plăci de dezvoltare), în timp ce registrul de deplasare TxRx_reg și unitatea de control trebuie să funcționeze la frecvența mai redusă a semnalului de ceas SCLK (de maxim 10 MHz pentru majoritatea dispozitivelor cu interfață SPI). O primă posibilitate ar fi utilizarea semnalului de ceas SCLK, care este generat prin divizarea semnalului de ceas al sistemului, ca semnal de ceas pentru componentele care trebuie să funcționeze la frecvența mai redusă. Această soluție nu este însă recomandată, deoarece utilizarea ca semnal de ceas a unui semnal generat printr-o logică combinațională poate crea probleme de sincronizare din cauza întârzierilor introduse la rutarea semnalului, ceea ce poate reduce fiabilitatea și frecvența maximă de funcționare a circuitului proiectat.

O soluție mai avantajoasă este utilizarea aceluiași semnal de ceas al sistemului pentru toate componentele secvențiale ale controlerului și generarea unui semnal de validare pentru componentele care trebuie să funcționeze la frecvență mai redusă. Acest semnal de validare poate fi generat sub forma unor impulsuri periodice, fiecare impuls având durata unei perioade a semnalului de ceas al sistemului. Toate operațiile acestor componente vor fi condiționate de semnalul de validare, funcționarea fiind echivalentă cu cazul în care s-ar utiliza un semnal de ceas cu o frecvență mai redusă.

Deoarece diferitele componente secvențiale trebuie să execute operațiile fie pe frontul crescător al semnalului de ceas SCLK, fie pe frontul descrescător al acestui semnal de ceas, se vor genera două semnale de validare diferite pentru aceste componente. Unul din semnalele de validare va consta din impulsuri periodice generate la fiecare front crescător al semnalului de ceas SCLK. Al doilea semnal de validare va consta din impulsuri periodice generate la fiecare front descrescător al semnalului

de ceas SCLK. Perioada de generare a impulsurilor poate fi determinată pe baza frecvenței de ceas a sistemului și a genericului care specifică frecvența semnalului de ceas SCLK.

Un alt detaliu este legat de semnalul *Start* al controlerului. Activarea acestui semnal trebuie detectată de unitatea de control care funcționează la frecvența mai redusă a semnalului SCLK. Deci, semnalul *Start* ar trebui menținut activ un număr suficient de cicluri de ceas pentru a fi detectat de unitatea de control. O posibilitate ar fi ca unitatea de control să genereze un semnal de confirmare a faptului că a detectat activarea semnalului *Start*, astfel încât acest semnal să poată fi dezactivat la detectarea semnalului de confirmare. Pentru a simplifica interfața controlerului, se va utiliza un bistabil pentru a memora activarea semnalului *Start*, astfel încât acest semnal va putea fi menținut activ pe durata unui singur impuls al semnalului de ceas al sistemului și nu va fi necesar un semnal de confirmare.

Pe baza acestor observații, rezultă o schemă bloc mai detaliată a controlerului SPI, schemă care este prezentată în figura 7.7.

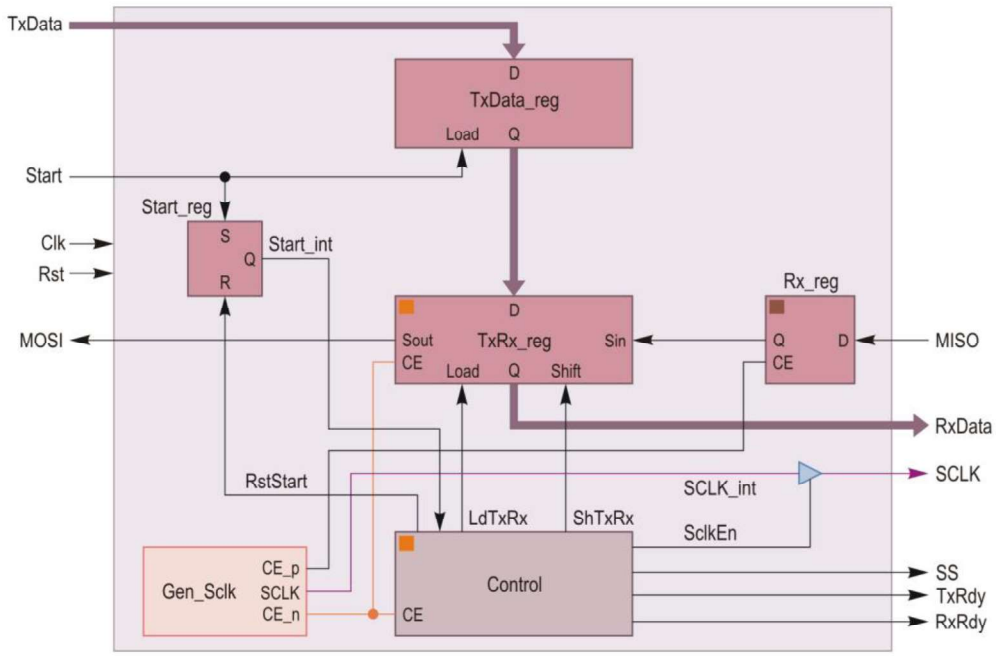

Figura 7.7. Schema bloc detaliată a controlerului SPI.

Modulul Gen_Sclk generează semnalul de ceas SCLK și două semnale de validare, *CE_p* și *CE_n*. Semnalul de validare *CE_p* se aplică la intrarea de validare *CE* a bistabilului Rx_reg care memorează un bit recepționat pe linia MISO la frontul crescător al semnalului SCLK. Semnalul de validare *CE_n* se aplică la intrarea de

validare *CE* a registrului de deplasare TxRx_reg și la intrarea de validare *CE* a unității de control. Componentele secvențiale care funcționează la frecvența mai redusă a semnalului de ceas SCLK sunt puse în evidență în figura 7.7 prin pătrate de culoare portocalie în colțul din stânga sus (pentru componentele care execută operațiile pe frontul descrescător al semnalului SCLK) sau de culoare maro (pentru bistabilul Rx_reg care se încarcă pe frontul crescător al semnalului SCLK).

Bistabilul Start_reg nu are intrare de date, ci doar o intrare de setare sincronă (S) și o intrare de resetare sincronă (R). Acest bistabil memorează activarea semnalului *Start* care este aplicat la intrarea de setare sincronă S a bistabilului. Resetarea bistabilului se realizează cu semnalul *RstStart* generat de unitatea de control.

Semnalele generate de modulul Gen_Sclk sunt ilustrate în figura 7.8, în care se presupune că frecvența semnalului SCLK este de 8 ori mai redusă față de frecvența semnalului de ceas *Clk* al sistemului. Semnalul de validare *CE_p* constă din impulsuri generate la fiecare front crescător al semnalului SCLK. Semnalul de validare *CE_n* constă din impulsuri generate la fiecare front descrescător al semnalului SCLK.

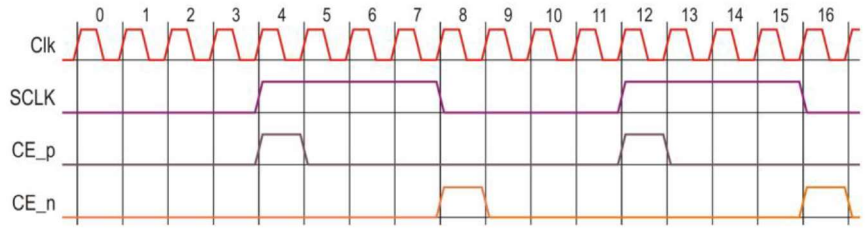

Figura 7.8. Semnalul de ceas SCLK şi semnalele de validare (*CE_p*, *CE_n*) generate de modulul Gen_Sclk.

7.2.3 Diagrama de stare pentru unitatea de control

Unitatea de control poate fi implementată printr-un automat cu diagrama de stare prezentată în figura 7.9. Toate operațiile executate de unitatea de control și tranzițiile între stări sunt condiționate, pe lângă frontul crescător al semnalului de ceas SCLK, de semnalul de validare *CE_n*, după cum rezultă din schema bloc detaliată a controlerului SPI. Pentru contorizarea biților transmiși și recepționați se utilizează variabila *CntBit*. Atunci când se detectează activarea semnalului *Start_int*, care reprezintă semnalul *Start* memorat în bistabilul Start_reg, automatul trece în starea `load`, în care se încarcă registrul TxRx_reg cu cuvântul care urmează să fie transmis. În aceeași stare se inițializează variabila *CntBit* cu $n - 1$, unde n este genericul care specifică dimensiunea cuvântului. De asemenea, se activează semnalul de selecție SS pentru dispozitivul *slave* și se resetează bistabilul Start_reg prin activarea semnalului *RstStart*.

În starea `tx_rx` se realizează transmisia și recepția unui bit. Pentru aceasta, se deplasează registrul TxRx_reg la stânga cu o poziție, introducând în poziția sa cea mai puțin semnificativă bitul care a fost recepționat anterior la frontul crescător al

semnalului de ceas SCLK și care a fost memorat în bistabilul Rx_reg. În aceeași stare se validează generarea semnalului de ceas SCLK prin activarea semnalului *SclkEn* și se decrementează contorul de biți *CntBit*. Dacă acest contor nu a ajuns la zero, se continuă cu transmisia și recepția biților următori, iar în caz contrar se trece în starea `bit0`, în care se transmite și se recepționează ultimul bit al cuvântului. În starea `bit0`, pe lângă operațiile executate în starea `tx_rx`, se activează semnalul de stare *TxRdy*, care indică faptul că se poate iniția transmiterea unui nou cuvânt prin aplicarea cuvântului la intrarea de date *TxData* și activarea semnalului *Start*.

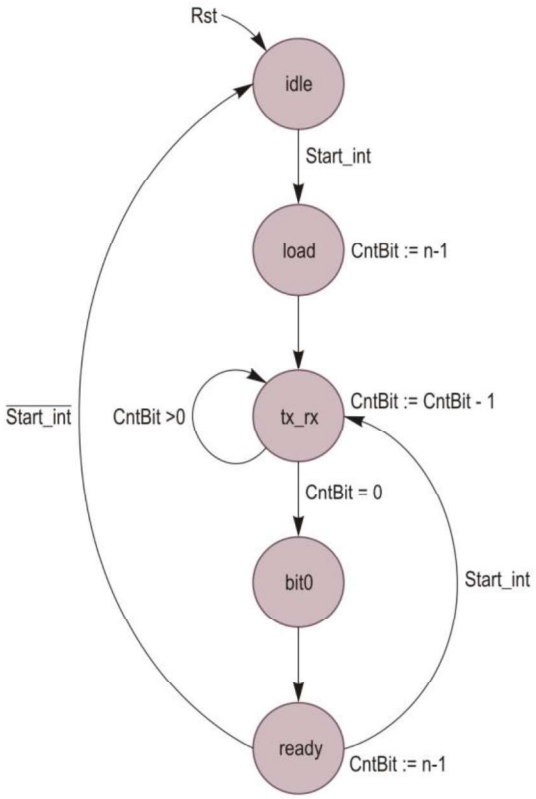

Figura 7.9. Diagrama de stare a unității de control.

Din starea `bit0` automatul trece necondiționat în starea `ready`, în care se activează semnalul de stare *RxRdy*, indicând faptul că datele de la ieșirea de date *RxData* sunt valide. În starea `ready` se reinițializează variabila *CntBit* cu $n - 1$ și se testează starea semnalului *Start_int*. Dacă acest semnal este activ, se încarcă registrul TxRx_reg cu următorul cuvânt care trebuie transmis și se trece în starea `tx_rx` pentru a continua transmiterea cuvântului; în caz contrar, se revine în starea `idle`.

7.3 Accelerometrul Analog Devices ADXL362

Circuitul ADXL362 este un accelerometru cu trei axe care poate furniza date despre accelerație cu o rezoluție de 12 biți sau opt biți. Domeniul de măsurare este selectabil între ±2 *g*, ±4 *g* și ±8 *g*, rezoluția fiind de 1 mg pentru domeniul de ±2 *g*. Accelerometrul conține un număr de 36 de registre de opt biți și o memorie FIFO (*First In, First Out*) care poate păstra 512 eșantioane ale datelor măsurate. De asemenea, accelerometrul conține un senzor de temperatură integrat care poate monitoriza temperatura internă sau poate îmbunătăți stabilitatea cu temperatura a circuitului prin operații de calibrare.

7.3.1 Interfața SPI a accelerometrului ADXL362

Circuitul ADXL362 conține o interfață serială SPI. Semnalul de selecție SS al interfeței SPI este denumit CS (*Chip Select*). Modul de comunicație SPI este 0, polaritatea CPOL și faza CPHA semnalului de ceas fiind ambele 0. Frecvența recomandată a semnalului de ceas SCLK este cuprinsă între 1 MHz și 8 MHz.

Accelerometrul ADXL362 recunoaște următoarele comenzi:

- 0x0A: Scriere registru;
- 0x0B: Citire registru;
- 0x0D: Citire memorie FIFO.

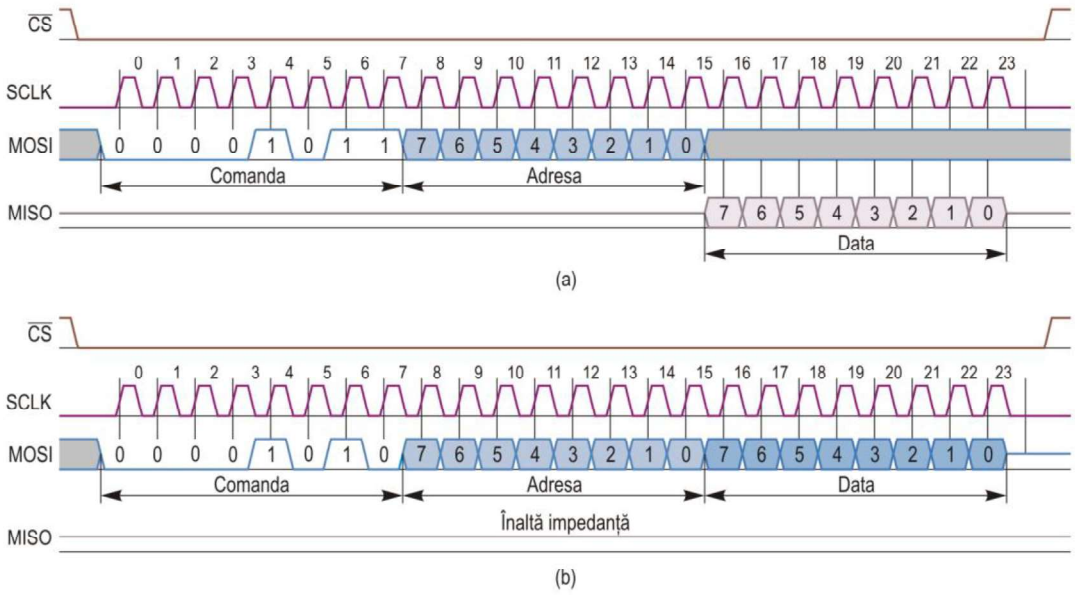

Figura 7.10. Comenzi de citire și de scriere a unui registru al accelerometrului ADXL362: (a) comandă de citire; (b) comandă de scriere.

Comenzile de scriere și de citire a unui registru trebuie urmate de adresa de opt biți a registrului care va fi scris sau citit. Adresele registrelor sunt cuprinse între 0x00 și 0x2E. Figura 7.10 ilustrează o comandă de citire a unui registru și o comandă de scriere a unui registru. Linia MISO se află în starea de înaltă impedanță, cu excepția intervalelor în care accelerometrul transmite date citite dintr-un registru.

Pentru toate comenzile se pot utiliza transferuri în mod exploziv (*burst*), în care se transmite circuitului doar adresa primului registru care trebuie scris sau citit, adresa fiind incrementată în mod automat după fiecare octet transferat. Transferul continuă cât timp se furnizează circuitului impulsuri ale semnalului de ceas SCLK. De exemplu, transferurile în mod exploziv se pot utiliza pentru citirea unui set complet de date ale accelerației pe axele *x*, *y* și *z* (și, opțional, a valorii temperaturii).

7.3.2 Registrele accelerometrului ADXL362

Se prezintă în continuare o parte din cele 36 de registre ale accelerometrului ADXL362. Tabelul 7.2 prezintă adresele hexazecimale ale acestor registre, denumirea lor și valoarea cu care sunt inițializate registrele după resetarea accelerometrului.

Tabelul 7.2. Exemple de registre ale accelerometrului ADXL362.

Adresa	Nume	Valoare inițială
0x00	DEVID_AD	0xAD
0x01	DEVID_MST	0x1D
0x02	PARTID	0xF2
0x03	REVID	0x01
0x08	XDATA	0x00
0x09	YDATA	0x00
0x0A	ZDATA	0x00
0x0B	STATUS	0x40

Registrele DEVID_AD, DEVID_MST și PARTID conțin identificatori specifici pentru firma Analog Devices și pentru accelerometrul ADXL362. Registrul REVID conține numărul de revizie al accelerometrului, număr care este 0x01 pentru prima revizie și este incrementat pentru fiecare nouă revizie. Registrele XDATA, YDATA și ZDATA conțin cei opt biți mai semnificativi ai valorilor accelerației pe axele *x*, *y*, respectiv *z*; aceste registre se pot utiliza atunci când rezoluția de opt biți pentru valorile accelerației este suficientă. Valorile complete ale accelerației sunt disponibile în alte registre care nu sunt prezentate aici. Registrul STATUS conține indicatori de stare ai accelerometrului.

Aplicații

7.1 Răspundeți la următoarele întrebări:

a. Care este deosebirea dintre o interfață serială sincronă, cum este SPI, și o interfață serială asincronă, cum este UART?

b. Care sunt posibilitățile pentru selecția dispozitivelor *slave* multiple în cazul interfeței SPI?

c. Ce reprezintă polaritatea semnalului de ceas și faza semnalului de ceas la interfața SPI?

7.2 Implementați în limbajul VHDL controlerul pentru interfața SPI descris în secțiunea 7.2. Parcurgeți etapele următoare pentru implementarea controlerului.

1. În mediul de proiectare Vivado, creați un nou proiect pentru placa de dezvoltare utilizată.
2. Creați un fișier sursă VHDL pentru modulul controlerului SPI. Porturile de intrare și de ieșire ale modulului sunt cele ilustrate în figura 7.7.
3. În entitatea modulului creat, adăugați un generic de tip `INTEGER` pentru frecvența semnalului de ceas SCLK și inițializați genericul cu valoarea 5_000_000 corespunzătoare frecvenței de 5 MHz.
4. În aceeași entitate, adăugați un alt generic de tip `INTEGER` pentru dimensiunea cuvântului și inițializați genericul cu valoarea 8.
5. Declarați o constantă de tip `INTEGER` pentru frecvența semnalului de ceas *Clk* și inițializați constanta cu 100_000_000 corespunzătoare frecvenței de 100 MHz a semnalului de ceas.
6. Declarați o constantă de tip `INTEGER` și inițializați constanta cu valoarea cu care trebuie divizată frecvența semnalului de ceas *Clk* pentru a se obține frecvența semnalului SCLK. Această valoare se poate obține împărțind cu operatorul "/" frecvența semnalului de ceas *Clk* la frecvența semnalului SCLK.
7. Declarați semnalele interne care apar în schema bloc din figura 7.7 și alte semnale care sunt necesare. Inițializați toate semnalele cu '0'.
8. Implementați printr-un proces modulul Gen_Sclk care generează semnalul de ceas SCLK și semnalele de validare *CE_p* și *CE_n*. Diagrama de timp a semnalelor care trebuie generate este cea din figura 7.8. În această diagramă s-a presupus că frecvența semnalului de ceas *Clk* trebuie divizată cu valoarea 8 pentru a obține frecvența semnalului SCLK, dar valoarea reală este dată de constanta inițializată în etapa 6. Utilizați o variabilă de tip `INTEGER` pentru contorizarea impulsurilor de ceas *Clk*.
9. Implementați registrele și bistabilele din figura 7.7 prin câte un proces, în același modul al controlerului SPI și nu în entități separate.
10. Implementați automatul de stare pentru unitatea de control cu diagrama de stare din figura 7.9 printr-un proces. Diagrama de stare este descrisă în secțiunea 7.2.3. Funcționarea automatului este condiționată de semnalul de validare *CE_n*.

În procesul secvențial, utilizați o variabilă *CntBit* de tip `INTEGER` pentru contorizarea biților transmiși și recepționați. Nu asignați semnalele de comandă care trebuie generate de unitatea de control în acest proces secvențial.

11. Utilizați instrucțiuni concurente pentru asignarea semnalelor de comandă necesare registrelor și bistabilelor, iar apoi pentru asignarea semnalelor de ieșire ale modulului.
12. Realizați elaborarea proiectului și corectați erorile, dacă există.

Observații

1. Operațiile registrelor și bistabilelor trebuie condiționate de frontul crescător al semnalului de ceas *Clk*.
2. Registrele și bistabilele trebuie să aibă și un semnal de resetare sincronă *Rst*, cu excepția bistabilului Start_reg, pentru care semnalul de resetare sincronă este *RstStart*.
3. Operațiile registrului de deplasare TxRx_reg sunt condiționate de semnalul de validare *CE_n*, iar operațiile bistabilului Rx_reg sunt condiționate de semnalul de validare *CE_p*.
4. Nu combinați condiția pentru frontul crescător al semnalului de ceas *Clk* cu nicio altă condiție.
5. În starea activă, semnalul SS al interfeței SPI are nivelul logic 0.

7.3 Completați proiectul creat pentru aplicația 7.2 cu un modul VHDL care transmite accelerometrului ADXL362 de pe placa de dezvoltare Nexys4 DDR sau Nexys A7 o comandă de citire și recepționează un număr de opt octeți cu conținutul primelor opt registre ale acestuia. Intrările modulului sunt semnalul de ceas *Clk*, semnalul de resetare sincronă *Rst*, un semnal *Read* care indică începerea operației de citire și semnalul MISO al interfeței SPI. Ieșirile modulului sunt semnalele MOSI, SCLK, SS ale interfeței SPI și doi vectori de câte 32 de biți *Data1* și *Data2*, care vor conține datele recepționate de la accelerometru, date care vor fi memorate într-un set de registre. După crearea fișierului VHDL al modulului, executați operațiile următoare pentru implementarea acestuia.

- Instanțiați entitatea controlerului SPI creat pentru aplicația 7.2, iar apoi declarați și inițializați semnalele necesare conectării porturilor acestuia.
- Declarați și inițializați cu 0 un semnal de tip `INTEGER` care se va utiliza pentru contorizarea octeților recepționați de la accelerometru.
- Scrieți un proces secvențial pentru un automat de stare care va reprezenta unitatea de control a modulului. Operațiile care ar trebui executate de acest automat

sunt descrise în continuare; fiecare operație trebuie executată într-o stare separată. Nu asignați în acest proces secvențial semnalele de comandă; acestea vor fi asignate prin instrucțiuni concurente.

1. Într-o stare inițială, se inițializează contorul de octeți cu 0 și se așteaptă activarea semnalului *Read*.
2. Se aplică la intrarea *TxData* a controlerului SPI codul comenzii de citire a unui registru (x"0B").
3. Se activează semnalul *Start* care este conectat la intrarea controlerului SPI, menținând codul comenzii de citire la intrarea *TxData* a acestuia.
4. Se așteaptă activarea semnalului *TxRdy* de către controlerul SPI, ceea ce indică faptul că se poate aplica la intrarea *TxData* următorul octet care trebuie transmis.
5. Se aplică la intrarea *TxData* a controlerului SPI adresa primului registru care trebuie citit (x"00").
6. Se activează semnalul *Start*, menținând adresa registrului la intrarea *TxData*.
7. Se așteaptă activarea semnalului *RxRdy* de către controlerul SPI, ceea ce indică terminarea recepției unui octet, acesta fiind disponibil la ieșirea *RxData* a controlerului SPI.
8. Se activează un semnal de comandă pentru înscrierea octetului recepționat într-un registru din setul de opt registre care va păstra datele recepționate de la accelerometru. De asemenea, se incrementează contorul de octeți.
9. Se testează contorul de octeți; dacă acesta a ajuns la valoarea 8, se revine în starea inițială, iar în caz contrar se revine în starea în care se așteaptă activarea semnalului *TxRdy*.

- Utilizați instrucțiuni concurente pentru asignarea semnalului *Start*, a semnalului de comandă pentru înscrierea octetului recepționat în setul de opt registre și a vectorului *TxData*. Acest vector va fi asignat în permanență cu adresa primului registru care trebuie citit (x"00"), cu excepția stărilor în care trebuie asignat cu codul comenzii de citire.
- Declarați un tip pentru setul de registre care se va utiliza pentru memorarea datelor recepționate de la accelerometru, ca un tablou de opt vectori de câte opt biți fiecare. Declarați apoi un semnal de acest tip.
- Scrieți un proces pentru setul de registre utilizat pentru memorarea datelor recepționate. În acest proces, înscrieți datele de la ieșirea *RxData* a controlerului SPI în registrul cu indexul dat de contorul de octeți actualizat în automatul de stare; înscrierea trebuie validată de semnalul de comandă corespunzător.

- Utilizați instrucțiuni concurente pentru asignarea semnalelor de ieșire *Data1* și *Data2* ale modulului. Semnalul *Data1* va fi asignat cu concatenarea conținutului primelor patru registre din setul de registre cu datele recepționate, iar semnalul *Data2* va fi asignat cu concatenarea conținutului celorlalte registre.
- Realizați elaborarea proiectului și corectați erorile, dacă există.

7.4 Completați proiectul creat pentru aplicația 7.3 cu un modul principal pentru testarea pe placa de dezvoltare Nexys4 DDR sau Nexys A7 a controlerului SPI și a modulului de comunicație cu accelerometrul ADXL362. Intrările și ieșirile modulului principal sunt similare cu cele ale modulului creat pentru aplicația 7.3, cu excepția vectorilor de ieșire *Data1* și *Data2*, care sunt înlocuiți cu vectorii *An* și *Seg* de câte opt biți, necesari pentru afișajul cu șapte segmente al plăcii. În plus, modulul va avea ca intrare un semnal de selecție *Sel*, care va fi conectat la un comutator de pe placă pentru a permite selecția datelor care vor fi afișate. Executați operațiile următoare pentru crearea și implementarea modulului principal.

- Creați un fișier sursă VHDL pentru modulul principal, fără a specifica porturile de intrare și de ieșire ale acestuia.
- Copiați declarația de entitate a modulului de comunicație cu accelerometrul în fișierul sursă al modulului principal. Modificați numele entității, adăugați portul de intrare *Sel* la declarația de entitate, modificați numele porturilor de ieșire *Data1* și *Data2* în *An*, respectiv *Seg*, și modificați dimensiunea acestor porturi la 8 biți.
- Adăugați la proiect fișierul sursă al unui modul pentru filtrarea oscilațiilor unui buton și fișierul sursă al multiplexorului `displ7seg` pentru afișajul cu șapte segmente al plăcii.
- Instanțiați entitatea modulului de comunicație cu accelerometrul, entitatea modulului pentru filtrarea oscilațiilor butonului și entitatea multiplexorului pentru afișajul cu șapte segmente. Declarați și inițializați semnalele necesare conectării porturilor acestora.
- Asignați la semnalul conectat la portul *Data* al modulului `displ7seg` unul din semnalele conectate la porturile *Data1* și *Data2* ale modulului de comunicație cu accelerometrul, în funcție de starea semnalului de selecție *Sel*.
- Adăugați la proiect fișierul de constrângeri al plăcii de dezvoltare și modificați fișierul pentru conectarea porturilor modulului principal la pinii circuitului FPGA. Conectați portul *Sel* la un comutator al plăcii (de exemplu, SW0), portul *Read* la butonul BTNU și portul *Rst* la butonul BTND. Conectați porturile *Seg* și *An* la afișajul cu șapte segmente și porturile interfeței SPI (MISO, MOSI,

SCLK, SS) la pinii corespunzători ai accelerometrului (portul SS trebuie conectat la pinul CSN).

- Realizați elaborarea proiectului și corectați erorile, dacă există.
- Setați opțiunile pentru sinteza și implementarea proiectului. De exemplu, puteți selecta strategia *Flow_RuntimeOptimized* atât pentru sinteză, cât și pentru implementare.
- Realizați sinteza și implementarea proiectului, iar apoi generați fișierul cu șirul de biți pentru configurarea circuitului FPGA.
- Conectați o placă de dezvoltare la calculator, configurați circuitul FPGA și verificați funcționarea proiectului, urmărind dacă datele recepționate corespund cu conținutul inițial al registrelor indicat în tabelul 7.2 (primii doi octeți recepționați vor fi 0xFF).

7.5 Adăugați la proiectul creat pentru aplicația 7.4 un modul ILA pentru vizualizarea semnalelor controlerului SPI, în modul descris în capitolul 6, secțiunea 6.4.7, cu următoarele observații:

- În fișierul sursă al controlerului SPI specificați atributul `keep` pentru semnalele *Start*, *Start_int*, *CE_p*, *CE_n*, semnalul de stare al automatului și semnalele de la ieșirile registrelor TxData_reg și TxRx_reg.
- În procesul automatului de stare, specificați atributul `keep` pentru variabila *CntBit*.
- Setați opțiunea *Mark Debug* pentru următoarele semnale și variabile din modulul controlerului SPI: semnalele și variabilele pentru care s-a specificat atributul `keep`, iar apoi pentru semnalele *RxRdy*, *TxRdy*, MISO, SCLK și SS.
- În modulul de comunicație cu accelerometrul, setați opțiunea *Mark Debug* pentru semnalul MOSI, iar în modulul principal pentru semnalul *Read* filtrat prin modulul pentru filtrarea oscilațiilor butonului.
- În fereastra de dialog *ILA Core Options*, specificați valoarea 2048 (în locul valorii 1024) pentru numărul eșantioanelor de date care vor fi capturate și bifați opțiunea *Capture Control*.

  După salvarea constrângerilor, închideți proiectul sintetizat. În panoul *Design Runs* din partea de jos a ecranului, executați un clic cu butonul din dreapta pe linia *synth_1* și selectați opțiunea **Force Up-to-Date** pentru a evita rularea din nou a sintezei proiectului. Realizați implementarea proiectului și generați șirul de biți pentru configurarea circuitului FPGA. Conectați o placă de dezvoltare la calculator și configurați circuitul FPGA. Adăugați semnalul *Read* filtrat în fereastra *Trigger Setup* selectând iconița *Add probe(s)*  . În aceeași fereastră

Trigger Setup, în câmpul *Value* selectați **R (0-to-1 transition)**. Selectați iconița *Run trigger for this ILA core*  în fereastra *Status*, după care apăsați butonul BTNU de pe placa de dezvoltare. Vizualizați forma de undă a semnalelor din fereastra *Waveform*. Se poate observa că generarea semnalului de ceas SCLK este oprită o perioadă de timp între octeții consecutivi care sunt transmiși și recepționați. Această perioadă corespunde stării `ready` a automatului de stare din figura 7.9, stare în care generarea semnalului SCLK nu este validată. Din această cauză, controlerul SPI implementat nu permite comunicația la viteza maximă.

7.6 Modificați descrierea controlerului SPI creat în aplicația 7.2 pentru a elimina perioada de timp între cuvintele consecutive transmise și recepționate în care nu se generează impulsuri ale semnalului de ceas SCLK. Generați din nou fișierul cu șirul de biți pentru configurarea circuitului FPGA și verificați funcționarea proiectului vizualizând semnalele în fereastra *Waveform*.

Indicații

1. Adăugați la controlerul SPI un registru RxData_reg pentru memorarea unui cuvânt recepționat. Validați încărcarea acestui registru în starea `bit0` a automatului de stare al controlerului SPI. În aceeași stare, validați și încărcarea registrului TxRx_reg.
2. În starea `ready` a automatului de stare al controlerului SPI, dacă semnalul *Start_int* este activ, trebuie continuată deplasarea registrului TxRx_reg (care este încărcat în acest moment cu următorul cuvânt care trebuie transmis) și trebuie continuată generarea impulsurilor semnalului de ceas SCLK.