CID2 - Platformă Sistem APB I

1. Introducere

Un sistem de tip microcontroller este compus, de obicei, dintr-un nucleu de procesare (denumit de cele mai multe ori UCP – Unitatea Centrală de Prelucrare) și mai multe periferice, cu diferite funcții. Conectarea acestor periferice cu Procesorul se face, de obicei, prin intermediul unor interfețe standard. De asemenea, pentru o interacțiune mai ușoară, Procesorul comunică cu perifericele prin intermediul unor scrieri și citiri la anumite adrese. În funcție de adresă, vor fi accesate resursele unui anumit periferic. Această tehnică se numește maparea perifericelor în memorie și se bazează pe împărțirea spațiului de adrese și alocarea fiecărui interval unui anumit periferic.

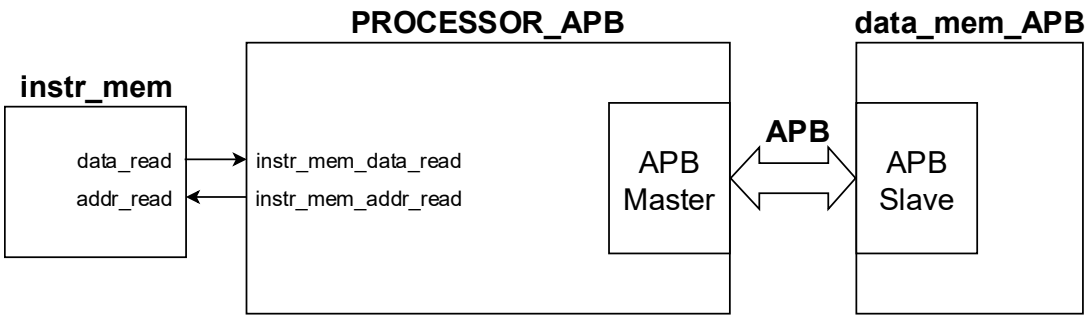
Pentru a exemplifica un astfel de sistem, vom modifica arhitectura Calculatorului implementat în etapa anterioară, astfel încât Memoria de Date să comunice cu Procesorul printr-o interfață ce implementează protocolul APB.

Protocolul APB face parte din standardul ARM AMBA (Advanced Microcontroller Bus Architecture) și este conceput ca o interfață cu complexitate redusă și consum minim de putere, prin care se pot conecta diversele componente ale unui sistem. Acest protocol nu oferă posibilitatea de transfer eficient a unor cantități mari de date, de aceea el este folosit în general pentru conectarea perifericelor a căror interacțiune se rezumă la transferul unor cantități reduse de date (de exemplu, configurarea de către Procesor a unor registre de control și citirea unor rezultate sau a unor registre de stare).

În practică, pentru conectarea unei memorii la UCP este folosit un protocol mai complex, care permite transferul eficient al unor cantități mari de date. Totuși, din motive didactice, vom conecta Memoria de Date prin APB.

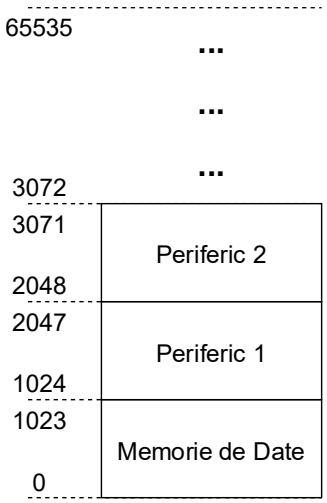
Pentru început, vom considera că Memoria de Date va fi singurul periferic conectat la Procesor prin APB. Pentru a realiza conexiunea, vom implementa un APB Master în zona Procesorului și un APB Slave în zona Memoriei. APB Master va iniția transferurile, iar APB Slave va răspunde fie prin scrierea datelor in memorie, în cazul unui acces de tip scriere, fie prin citirea din memorie a datelor solicitate și punerea lor pe APB, în cazul unui acces de tip citire.

Sistemul dezvoltat anterior se va modifica astfel:



2. Definirea spațiului de adrese

Deoarece toate perifericele sistemului trebuie să fie acoperite de spațiul de adrese disponibil (adresa sistemului dezvoltat anterior este pe 16 biți, ceea ce rezultă în 65536 adrese unice), vom reduce dimensiunea Memoriei de Date la 1024 locații pe 16 biți, astfel încât vom folosi doar 10 biți pentru adresa efectivă.



3. Interfața APB

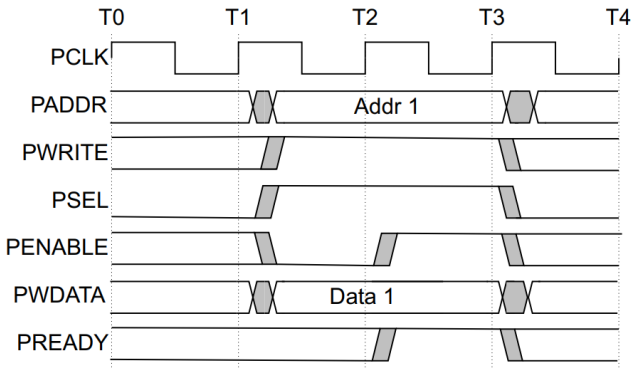
Specificația protocolului APB este disponibilă gratuit online. În specificație vom găsi detaliile despre funcționarea acestei interfețe. Deși protocolul APB presupune folosirea mai multor semnale, la acest laborator vom implementa o formă simplificată, ce conține doar următoarele semnale:

Semnal	Dimensiune	Descriere
pclk	1	Semnalul de ceas
preset	1	Semnalul de reset
paddr	10	Magistrală de adrese folosită atât pentru citire, cât și pentru scriere. Dimensiunea este dată de dimensiunea necesară a adresei.
pselx (psel_mem)	1	Semnal de selecție a perifericului ce va fi accesat. Vom avea câte un semnal de selecție pentru fiecare periferic conectat la APB. Pentru că momentan avem un singur periferic, vom senumi acest semnal *psel_mem*.
penable	1	Semnal de enable. Este 0 în primul ciclu de ceas al transferului și 1 în rest.
pwrite	1	Semnal ce arată tipul transferului: 1 pentru scriere și 0 pentru citire.
pwdata	16	Magistrală de date de scriere. Dimensiunea este dată de dimensiunea datelor cerută de arhitectura sistemului.
prdata	16	Magistrală de date de citire. Dimensiunea este dată de dimensiunea datelor cerută de arhitectura sistemului.
pready	1	Semnal prin care perifericul semnalizează dacă este pregătit să dea curs accesului solicitat de APB Master.

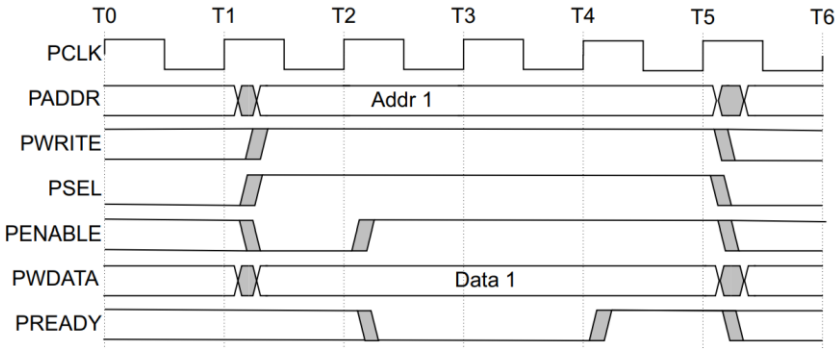
Mai jos, sunt prezentate formele de undă ale acceselor de scriere și citire, extrase din documentația protocolului.

Acces de scriere:

a) fără așteptare (periferic disponibil imediat):



b) cu așteptare (perifericul nu este disponibil imediat):

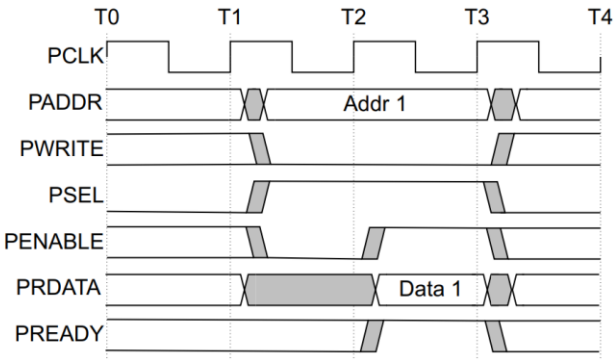


Concluzii legate de accesul de scriere:

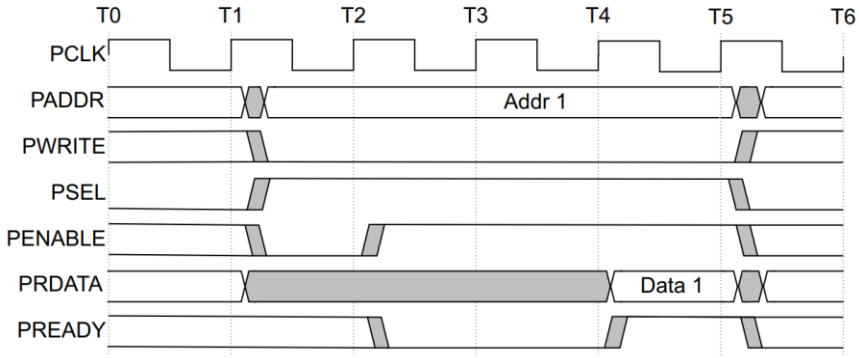
1) Accesul durează minim două cicluri de ceas (T1-T3 în primul caz și T1-T5 în al doilea caz).
2) Adresa (*paddr*) și data (*pwdata*) vor fi disponibile pe tot parcursul accesului.
3) *pwrite*, care indică o scriere și *psel*, care selectează perifericul, sunt 1 pe tot parcursul accesului.
4) *penable* este 0 în primul ciclu de ceas al transferului și 1 în celelalte cicluri.
5) Perifericul poate duce linia *pready* în 0 dacă Masterul încearcă să îl acceseze, dar nu este disponibil. Accesul rămâne în așteptare până când perifericul semnalizează disponibilitatea prin urcarea lui *pready* în 1 logic. Accesul se termină la primul front crecător al ceasului în care se detectează *pready* egal cu 1.

Acces de citire:

c) fără așteptare (periferic disponibil imediat):



d) cu așteptare (perifericul nu este disponibil imediat):



Concluzii legate de accesul de citire:

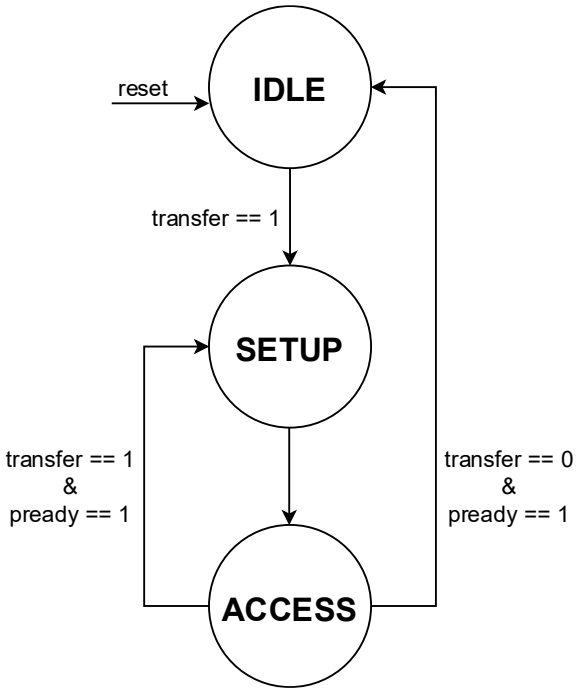
1) Accesul durează minim două cicluri de ceas (T1-T3 în primul caz și T1-T5 în al doilea caz).
2) Adresa (*paddr*) va fi disponibilă pe tot parcursul accesului.
3) *pwrite* va fi 0 și *psel* va fi 1 pe tot parcursul accesului.
4) *penable* este 0 în primul ciclu de ceas al transferului și 1 în celelalte cicluri.
5) Perifericul poate duce linia *pready* în 0 dacă Masterul încearcă să îl acceseze, dar nu este disponibil. Accesul rămâne în așteptare până când perifericul semnalizează disponibilitatea prin urcarea lui *pready* în 1 logic. Accesul se termină la primul front crecător al ceasului în care se detectează *pready* egal cu 1.
6) Data citită este disponibilă pe magistrala *prdata* în ultimul ciclu de ceas al transferului (odată cu urcarea lui *pready* în 1).

4. APB Master

APB Master este modulul din Procesor care va controla semnalele interfeței APB. Urmărind formele de undă prezentate anterior, putem defini 3 stări ale interfeței:

- IDLE: stare în care nu se efectuează niciun acces. În această stare, toate semnalele APB au valori implicite.
- SETUP: stare ce acoperă primul ciclu din transferul propriu-zis (T1-T2 în formele de undă).
- ACCESS: stare ce acoperă restul ciclurilor de ceas din transfer.

Graful unui automat care implementează funcționalitatea de mai sus este prezentat în figura următoare:



Tranziția stărilor:

- IDLE: Din această stare se va trece în SETUP doar la apariția unei cereri de transfer (scriere - STORE sau citire - LOAD).
- SETUP: Deoarece această stare corespunde primului ciclu dintr-un transfer, trecerea în starea următoare (ACCESS) se va face necondiționat, la următorul front crescător al ceasului.
- ACCESS: În această stare se va sta până când perifericul accesat va semnaliza prin *pready* că este disponibil. Dacă apare un nou transfer după ce *pready* devine 1, se va trece direct în starea SETUP. În caz contrar, se va trece în IDLE, în așteptarea unui nou transfer.

Interfațarea APB Master cu Procesorul și APB:

Ținând cont de specificul interfeței APB și a semnalelor definite pentru interfațarea memoriei în cadrul implementării din laboratoarele anterioare, automatul va comunica cu restul sistemului prin următoarele semnale:

Semnal	Dimensiune	Tip semnal	Descriere
Interfațarea cu Procesorul			
data_mem_addr_read	16	input	Adresa de la care se va face citirea
data_mem_addr_write	16	input	Adresa la care se va face scrierea
data_mem_data_write	16	input	Data ce va fi scrisă
data_mem_data_read	16	output	Data citită
wr_transfer	1	input	Semnal ce va indica prezența unei instrucțiuni STORE, ce declanșează o scriere pe APB.
rd_transfer	1	input	Semnal ce va indica prezența unei instrucțiuni LOAD, ce declanșează o citire pe APB.
block_pc	1	output	Semnal ce va bloca actualizarea PC. Deoarece un transfer durează minim 2 cicluri de ceas, nu trebuie să aducem o nouă instrucțiune până când transferul nu este terminat.
Interfațarea cu APB			
paddr	10	output	Magistrală de adrese folosită atât pentru citire, cât și pentru scriere. Dimensiunea este dată de dimensiunea adresei necesară.
psel_mem	1	output	Semnal de selecție a perifericului ce va fi accesat. Vom avea câte un semnal de selecție pentru fiecare periferic conectat la APB. Pentru că momentan avem un singur periferic, vom senumi acest semnal *psel_mem*.
penable	1	output	Semnal de enable. Este 0 în primul ciclu de ceas al transferului și 1 în rest.
pwrite	1	output	Semnal ce arată tipul transferului: 1 pentru scriere și 0 pentru citire.
pwdata	16	output	Magistrală de date de scriere. Dimensiunea este dată de dimensiunea datelor cerută de arhitectura sistemului.
prdata	16	input	Magistrală de date de citire. Dimensiunea este dată de dimensiunea datelor cerută de arhitectura sistemului.

pready	1	input	Semnal prin care perifericul semnalizează dacă este pregătit să dea curs accesului solicitat de APB Master.

Controlul ieșirilor automatelor:

Semnal	Descriere
Interfațarea cu Procesorul	
data_mem_data_read	Data citită. Va lua valoarea *prdata* atunci când starea automatului este ACCESS și *pready* este 1.
block_pc	Semnal ce va bloca actualizarea PC. Va fi 1 în timpul stării IDLE dacă apare un transfer, în timpul stării SETUP indiferent de starea intrărilor și în timpul stării ACCESS dacă *pready* este 0.
Interfațarea cu APB	
paddr	Magistrală de adrese folosită atât pentru citire, cât și pentru scriere. Pe tot parcursul transferului (stările SETUP și ACCESS) va lua valoarea *data_mem_addr_write* dacă se efectuează o scriere, sau *data_mem_addr_read* dacă se efectuează o citire.
psel_mem	Semnal de selecție a perifericului ce va fi accesat. Va fi 1 pe tot parcursul transferului (stările SETUP și ACCESS).
penable	Semnal de enable. Este 0 în primul ciclu de ceas al transferului (starea SETUP) și 1 în rest (starea ACCESS).
pwrite	Semnal ce arată tipul transferului. Va fi 1 pe tot parcursul transferului (stările SETUP și ACCESS) dacă avem scriere și 0 dacă avem citire.
pwdata	Magistrală de date de scriere. Va lua pe toată durata transferului valoarea lui *data_mem_data_write*.

Obs: În orice alt caz, semnalele de mai sus pot lua valoarea 0.