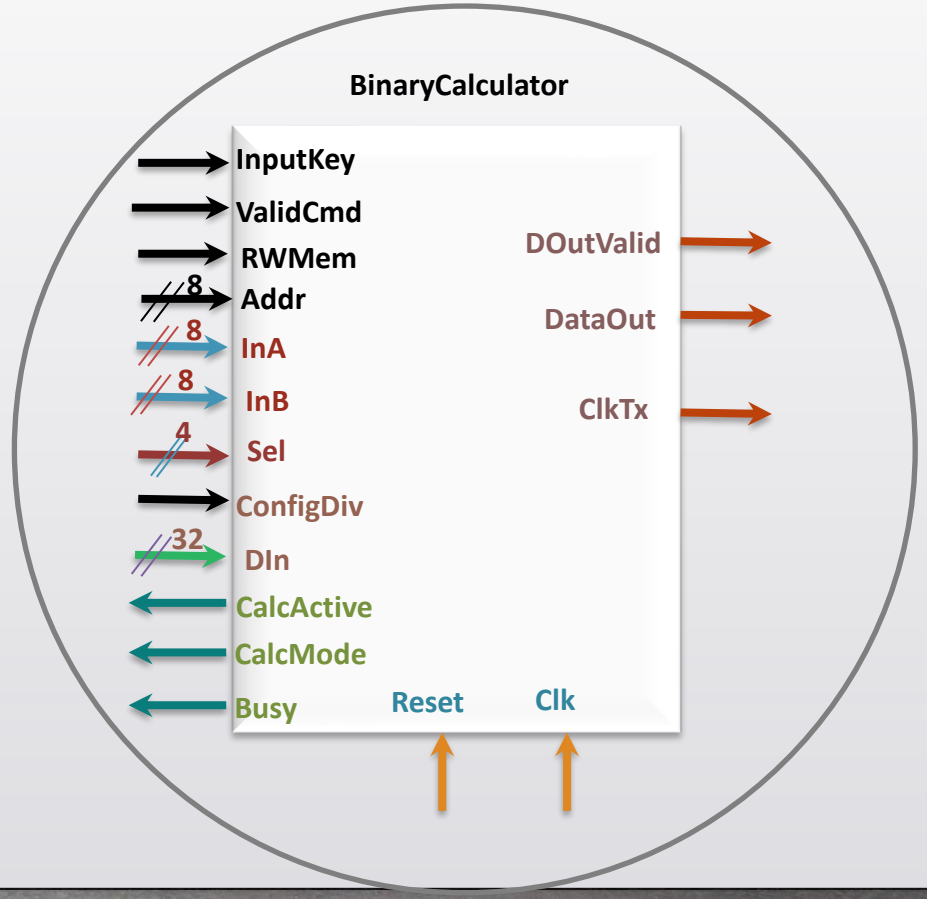
AMD – Summer practice 2023

Proiect – Grupa Verilog

Ce ne propunem?

- Realizarea unui proiect
  - Calculator Binar



Funcționalități

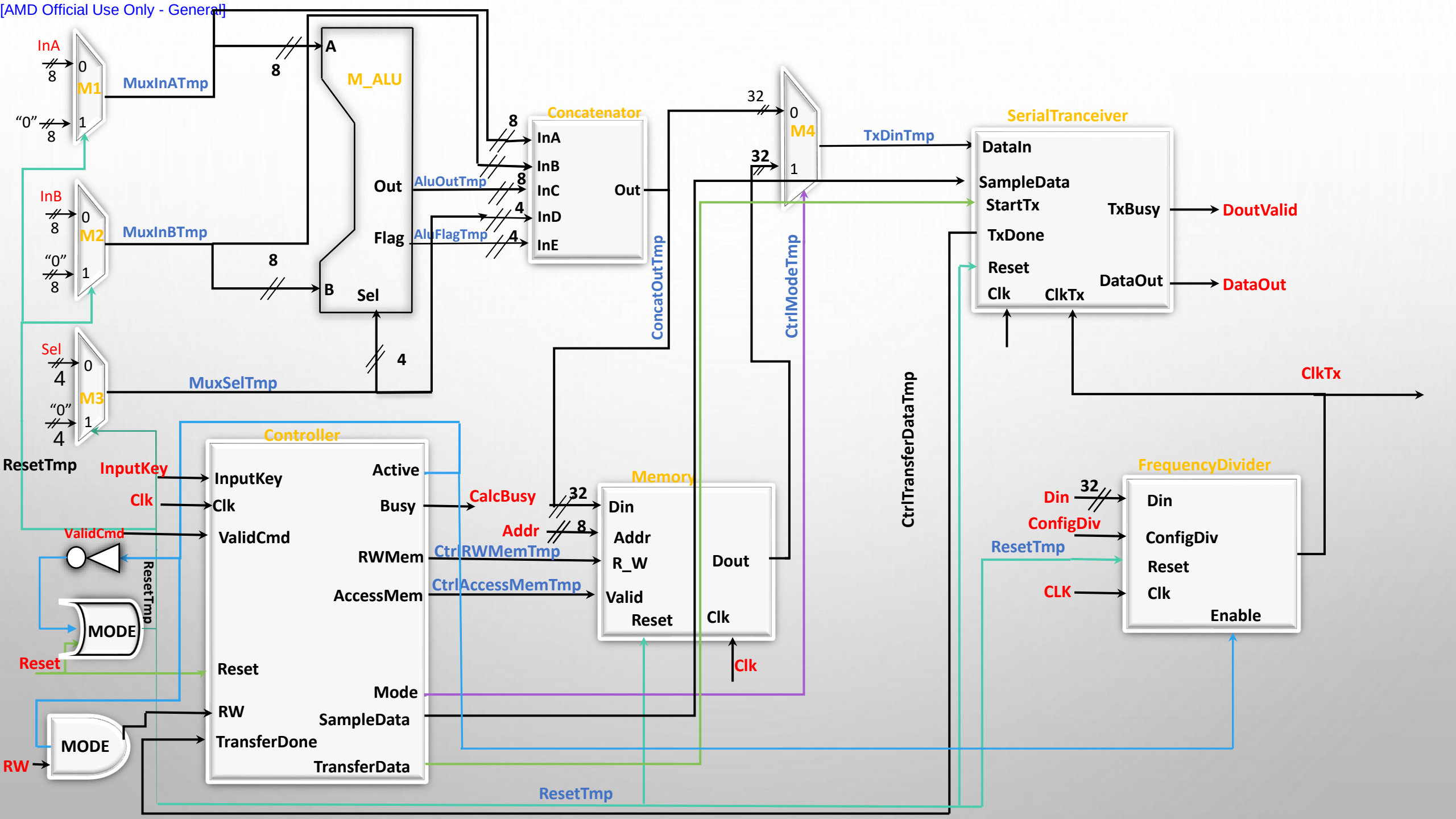
2 moduri de operare:

- Calcul -> *Memorare* -> Transfer.
- Calcul + Transfer.

Activare utilizând o ***frază secretă.***

Posibilitate de stocare de până la 256 de calcule.

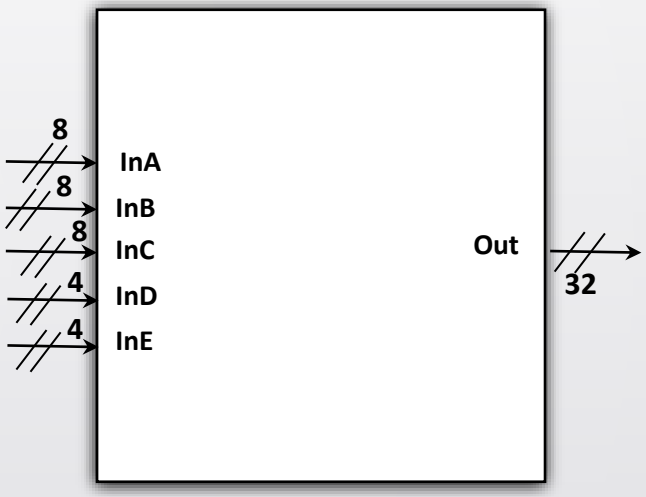
Transfer serial la ***frecvențe variabile.***



Concatenator

Mod de funcționare:

- Circuit combinațional
- Out = { E,D,C,B,A} => concatenarea celor 5 semnale
  - Exemplu:
    - A = 8'h01
    - B = 8'h02;
    - C = 8'h03;
    - D = 4'h4;
    - E = 4'h5;
      - OUT = 32'h5403_0201



ALU

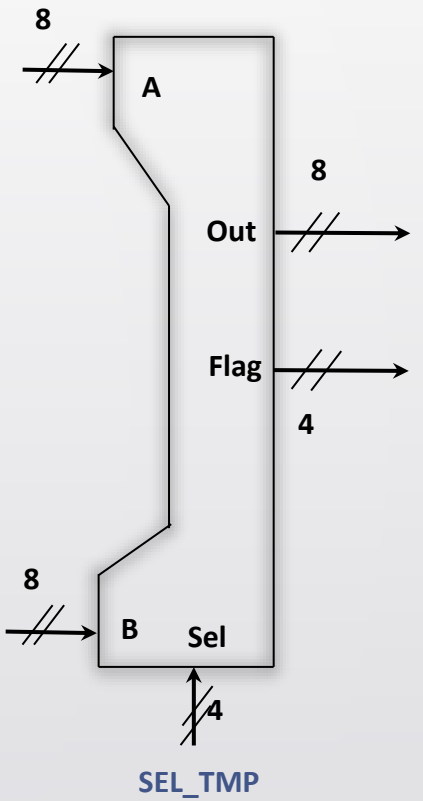
M_ALU

Operații:

- Sel= 4'h0 => Adunare:
  - Dacă A+B nu se pot reprezenta pe 8 biți:
    - Activare CarryFlag.
- Sel = 4'h1 => Scădere:
  - Dacă A – B < 0:
    - Se va activa UnderFlowFlag.
- Sel = 4'h2 => Înmulțire:
  - Dacă intrările A > 8’hF și B > 8’hF sau orice
  - alti operanzi care depasesc rezultatul pe 8 biti:
    - Se va activa OverflowFlag.
- Sel = 4’h3 => Împărțire:
  - Dacă intrările A < B:
    - Se va activa UnderFlowFLag.
- Sel = 4’h4 => Shiftare Stânga:
  - Dacă se dorește shiftare stânga (A << B), se poate utiliza bit-ul CarryFlag.
- Sel = 4’h5 => Shiftare Dreapta
  - Dacă se dorește shiftare dreapta (A >> B), se poate utiliza bit-ul CarryFlag.
- Sel = 4’h6 => AND
- Sel = 4’h7 => OR
- Sel = 4’h8 => XOR
- Sel = 4’h9 => NXOR
- Sel = 4’hA => NAND
- Sel = 4'hB => NOR
- Sel [4'hC : 4'hF] => Default value:
  - OUT = 8'h00;
  - FLAG = 4'h0;

Flag:

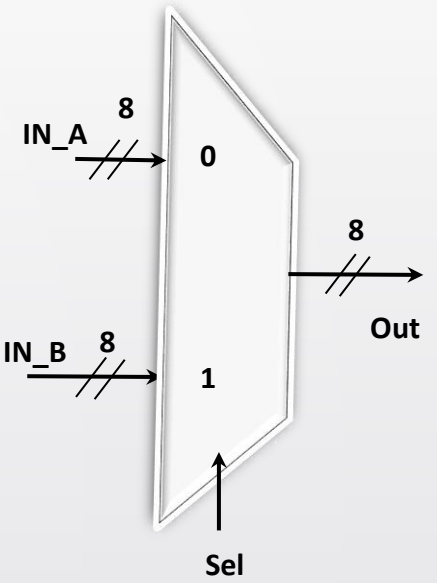
- Bit 0 : ZeroFlag
  - Dacă Out == 0 && Sel este Valid (0:B):
    - Se va activa ZeroFlag
- Bit 1: CarryFlag
- Bit 2: OverflowFlag
- Bit 3: UnderflowFlag



MUX-uri

Mod de utilizare:

- Circuit combinațional cu 3 intrări și o ieșire. Poate fi modelat ca un comutator.
- Intrări:
  - Sel= bit de selecție pentru a conecta una din intrări la ieșire.
  - I0, I1 = intrări pe 8 biți de date.
- Out: ieșire pe 8 biți.



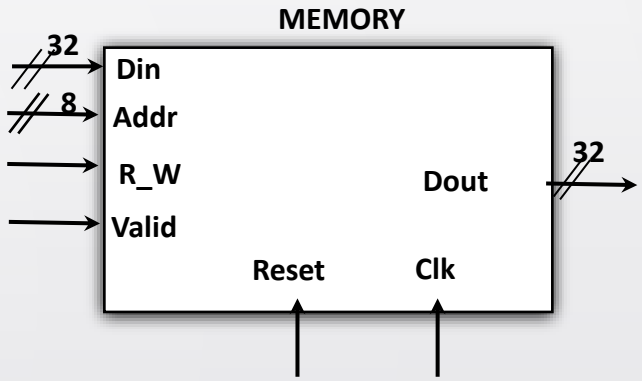
Scop in cadrul proiectului:

- **M1->M3** : Mux-uri utilizate pentru a calcula o operație:
  - SEL- Mux3: reprezintă intrare selecție operatie.Tipul operației: adunare, scădere, înmulțire etc.
  - IN_A : primul operand al operației.
  - IN_B: al doilea operand al operației.
  - În cazul în care semnalul ResetTmp este activ => intrările operației vor fi legate la "0" logic.
- **M4**: Mux utilizat în conexiunea blocului logic Register_Shiftare_Cu_ParallelLoad cu unul dintre celelalte 2 blocuri logice (M_ALU, Memorie)
  - Dacă calculatorul nu este în modul de memorare (CTRL_MODE = 0) datele de intrare vor fi preluate de la M_ALU.
  - Dacă calculatorul este în modul de memorare (CTRL_MODE = 1), datele de intrare vor fi preluate din Memorie.

Memorie

Mod de funcționare:

- Bloc logic activ la frontul pozitiv al semnalului Clk.
- Bloc **parametrizabil**. WIDTH reprezentând numărul de adrese din memorie.
  - WIDTH - dimensiunea bus-ului de adresă va fi în funcție de WIDTH. Implicit va avea valoarea 8.
  - DinLENGTH - parametru utilizat în modelarea dimensiunii bus-ului de date Din și Dout. Valoarea implicită va fi 32.
- Reset asincron activ pe frontul pozitiv.
  - Toate adresele for fi resetate.
  - Ieșirea Dout va fi resetată.
- Reset=0
  - Valid=0
    - Memoria nu va putea fi accesată
  - Valid=1
    - R/W=0(Read)
      - Data din Addr va fi returnată semnalului de iesire Dout.
    - R/W=1(Write)
      - Data eșantionată a semnalului Din va fi stocată la adresa specificată de semnalul Addr.



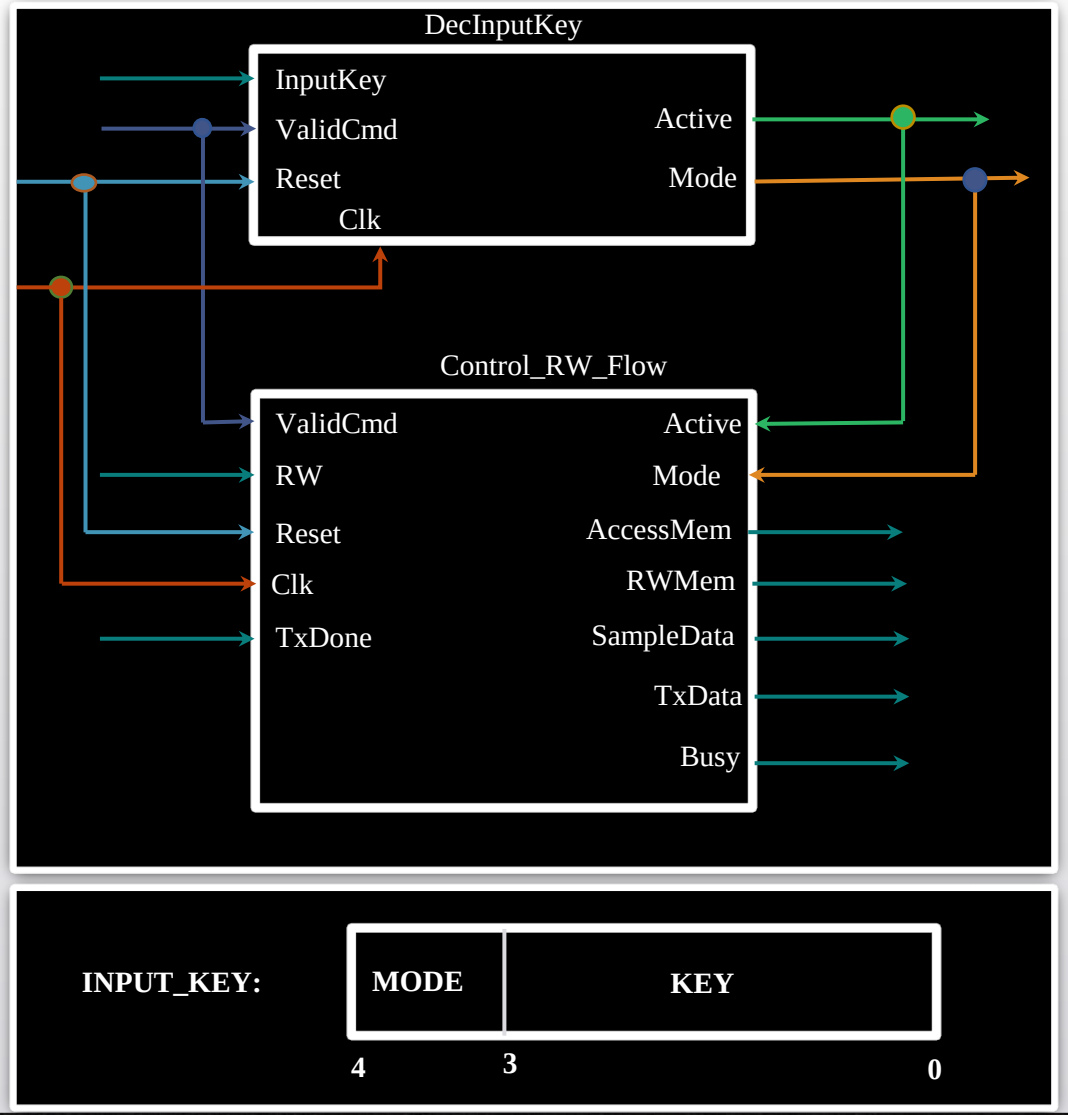
Unitate de control

DecInputKey

Mod de funcționare:

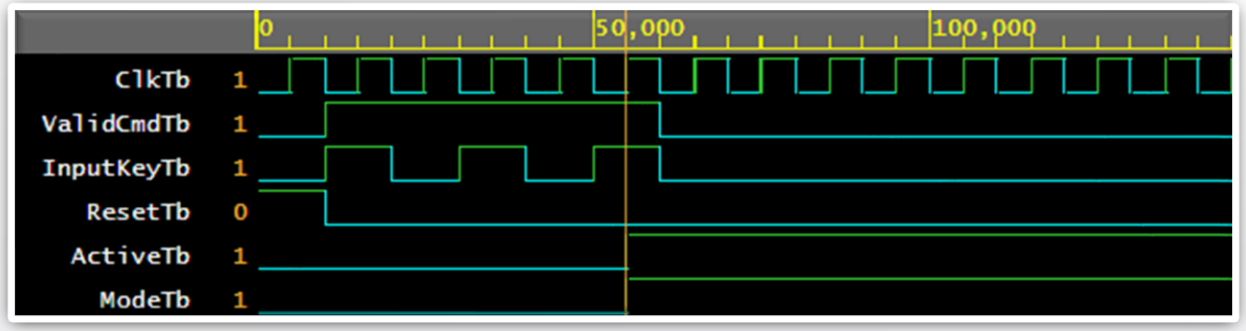
Fraza secretă : **1(LSB) – 0 – 1 – 0 - M(MSB).**

- Ordinea de introducere LSB –> MSB.
- Se va introduce câte un bit la fronturi pozitive succesive de ceas.
- Dacă **Reset=1**
  - Decodificatorul va fi resetat: Active=0, Mode=0.
  - Registrii interni vor fi resetați;
- Dacă **Reset=0**
  - **ValidCmd=1**
    - Se poate introduce INPUT_KEY(utilizat la activarea Controller-ului) ce reprezintă fraza de decodificare.
    - După ce s-a introdus corect InputKey, la fiecare front pozitiv de ceas se introduce Modul:
      - **Mode=0:** Mod de calcul+transfer.
        - Ieșirile vor fi: Active = 1, Mode = 0
      - **Mode=1:** Mod de calcul + memorare la adresa specificată.
        - Ieșirile vor fi: Active = 1, Mode = 1.

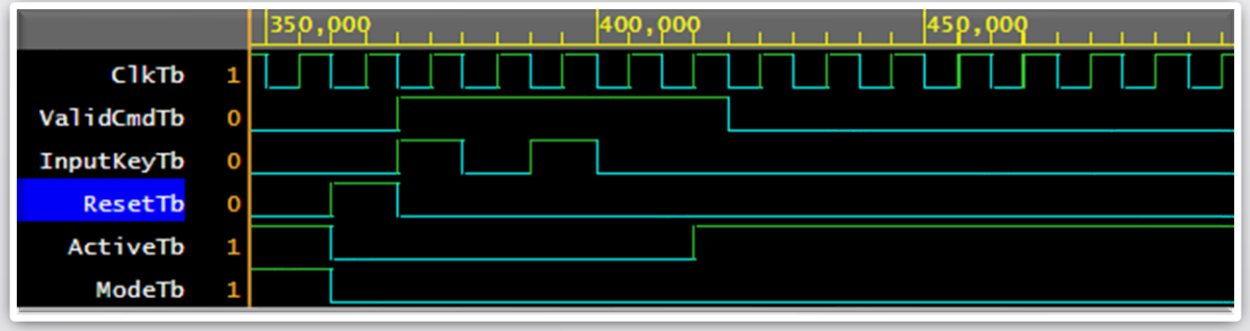


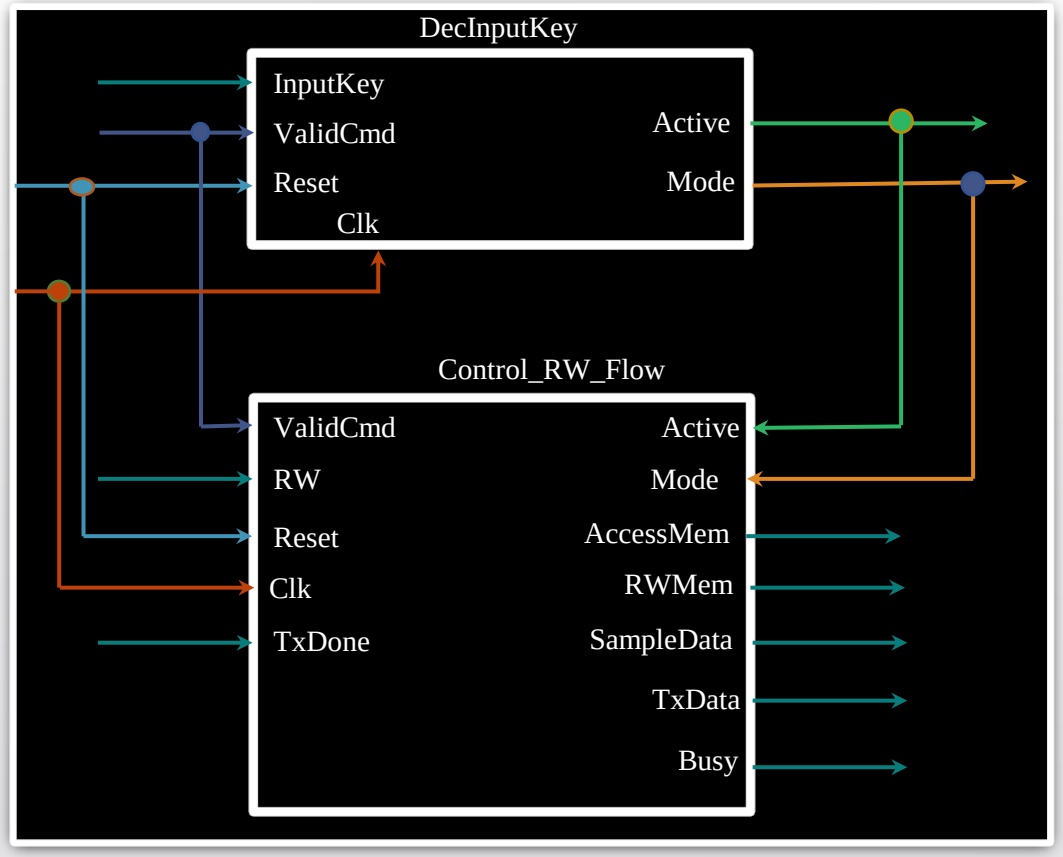
Unitate de control

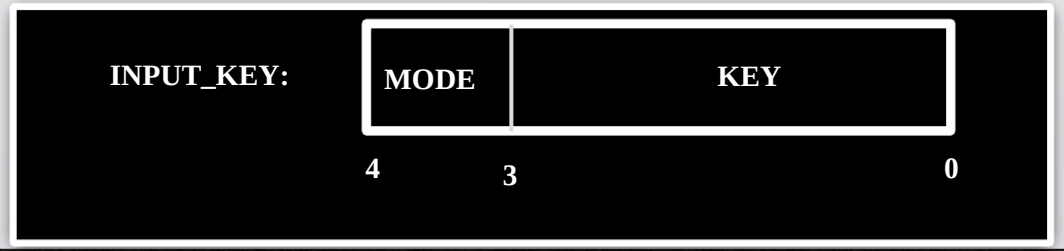
Exemplu de funcționare DecInputKey. MODE = 1:



Exemplu de funcționare DecInputKey. MODE = 0:





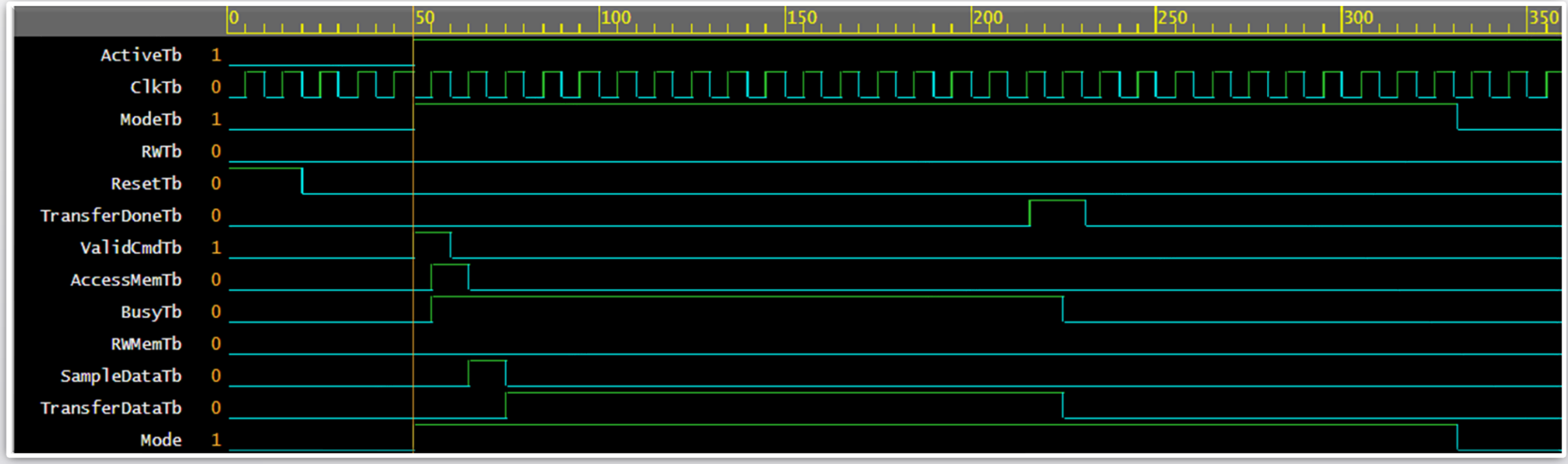


Unitate de control Read/Write FLOW (funcționare)

Comanda de READ din memorie. MODE = 1 (Acces la memorie, datele generate de ALU vor fi citite din memorie).

- Dacă Reset = 1:
  - Controller-ul va fi resetat.
  - Semnalele AccesMem = RWMem = SampleData = TransferData = Busy = 0.
- Dacă Reset = 0:
  - Dacă ValidCmd = 0:
    - Controllerul nu va procesa nici o comandă.
  - Dacă **ValidCmd = 1 && Active = 1 && Mode = 1 && RW = 0:**
    - Controller-ul va iniția o secvență de read de tipul: **Idle -> ReadMemory -> Sample SerialTransceiver -> Start Transfer Serial Transceiver -> Wait Transfer Done -> Idle**
    - **Clock-ul 1 ReadMemory**:
      - Va interpreta valoarea logică returnată de: **ValidCmd = 1 && Active = 1 && Mode = 1 && RW = 0**
      - Controllerul va activa semnalele(AccessMem=1, RWMem=0) generând o comandă de citire către memoria prezentă cu datele generate de ALU. In acest moment, Controllerul va activa și bitul BUSY. Clock-ul 1 va fi singurul moment în care vom evalua semnalul ValidCmd. In continuarea flow-ului de citire din memorie, semnalul ValidCmd nu va mai fi luat în calcul.
      - Odată ce secvența de Read a început, Controller-ul nu mai poate procesa alt tip de request(R/W);
    - **Clock-ul 2 Sample SerialTransceiver**:
      - Va interpreta valoarea logică returnată de: **Active && Mode && TransferDone = 0**
      - Controllerul va activa semnalul SampleData pentru încărcarea datei citite din memorie în SerialTranceiver. Pe tot parcursul secventei de citire, semnalul BUSY va rămâne asertat.
    - **Clock-ul 3 Start Transfer SerialTransceiver**:
      - Va interpreta valoarea logică: **Active && !TransferDone**
      - Controller-ul va activa semnalul TransferData pentru pornirea unui nou transfer.
    - **Clock-ul 4 -> n Wait Transfer Done:**
      - Va interpreta valoarea logica**: Active && !TransferDone**
      - Unde n reprezintă numarul de biti transferați. Controllerul va rămâne într-o stare de așteptare până când transferul va fi incheiat (TransferDone=1). După terminarea transferului (TransferDone = 1) controller-ul va fi trimis în starea IDLE.

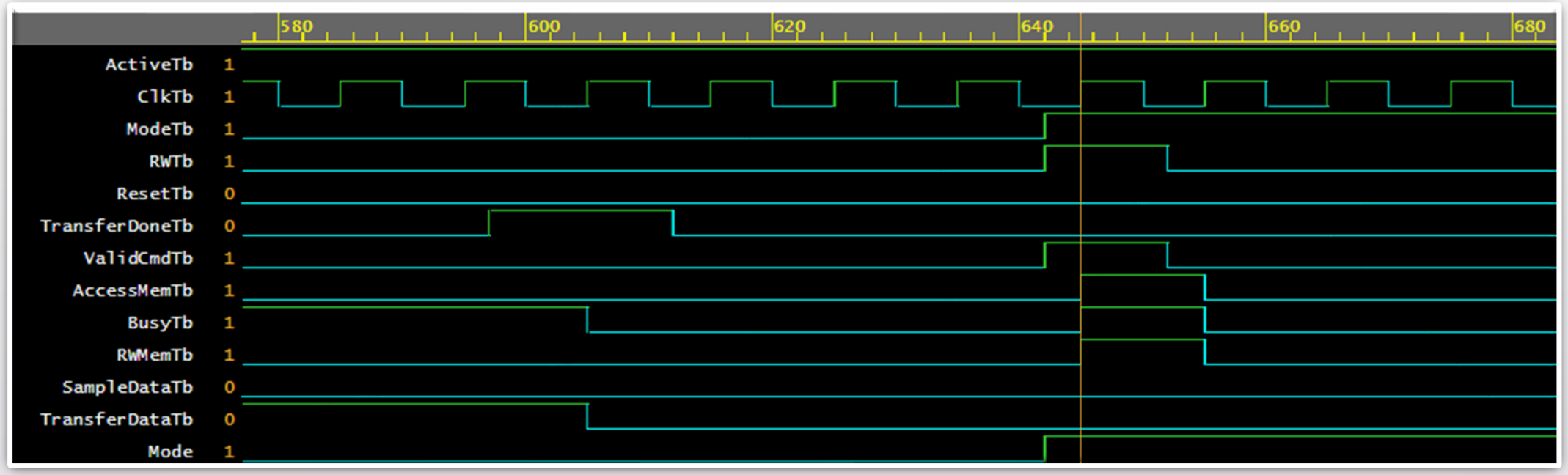
Exemplu transfer. MODE = 1. Comanda de READ din memorie.



Comanda de WRITE în memorie.
MODE = 1 (Acces la memorie, datele generate de ALU vor fi scrise in memorie).

- Dacă Reset = 0:
  - Dacă ValidCmd = 0:
    - Controllerul nu va procesa nici o comandă.
  - Dacă **ValidCmd = 1 && Active = 1 && Mode = 1 && RW = 1:**
    - Controller-ul va iniția o secvență de write, de tipul: **Idle -> WriteMemory -> Idle.**
    - Secvența de write poate fi și multiplă (scrieri succesive la fiecare front de ceas).
    - **Clock-ul 1 :**
      - **WriteMemory**:
        - Va interpreta valoarea logică returnată de: **ValidCmd = 1 && Active = 1 && Mode = 1 && RW = 1**
        - Controllerul va activa semnalele (AccessMem=1, RWMem=1) generând o comandă de scriere către memoria prezentă cu datele generate de ALU. In acest moment, controllerul va activa și bitul BUSY.
        - După finalizarea scrierii în memorie controller-ul va fi trimis in starea Idle.

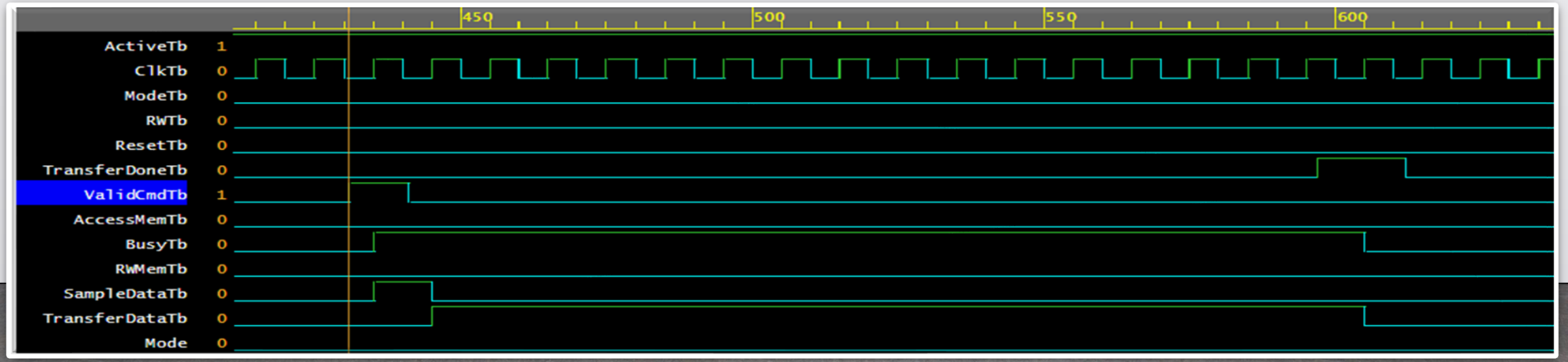
Exemplu transfer. MODE = 1. Comanda de WRITE în memorie.



- unde n reprezintă numărul de biti transferați. Controllerul va rămâne într-o stare de așteptare până când transferul va fi incheiat (TransferDone=1). In tot acest timp, controllerul nu va mai putea prelua comenzi de Read/Write de la utilizator.
- După terminarea transferului (TransferDone = 1) controller-ul va fi trimis într-o stare de Idle.

MODE = 0 (Acces la memorie, datele generate de ALU vor fi transferate pe interfața serială)

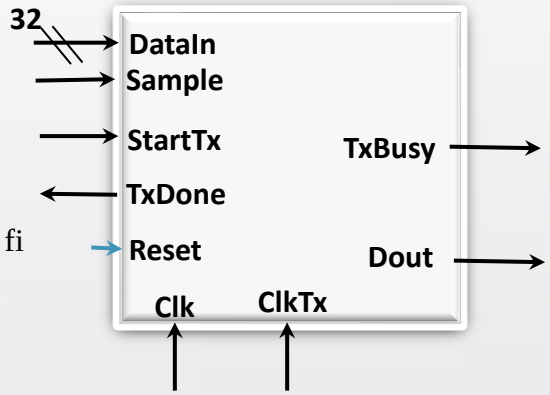
- Dacă Reset = 0:
  - Dacă **ValidCmd = 1 && Active = 1 && Mode = 0:**
    - Controller-ul va iniția o secvență de read : **Idle** -> **Sample SerialTransceiver** (cu data oferită de ALU) **-> Start Transfer Serial Transceiver** -> **Wait Transfer Done -> Idle**
    - **Clock-ul 1**: SampleSerial Transceiver: Va interpreta valoarea logică returnată de: **ValidCmd = 1 && Active = 1 && Mode = 0**
      - Controllerul va activa semnalul SampleData pentru încărcarea datei citite generate de ALU. Pe tot parcursul transferului, semnalul BUSY va rămâne asertat.
    - **Clock-ul 2**: Start Serial Transfer: Va interpreta valoarea logica returnată de: **Active && Mode = 0 && TransferDone = 0**
      - Controller-ul va activa semnalul TransferData pentru pornirea unui nou transfer.
    - **Clock-ul 3 -> n**: Wait TransferDone: Va interpreta valoarea logică: **Active && !TransferDone**
      - Controller-ul va activa semnalul TransferData pentru pornirea unui nou transfer.
      - n reprezintă numărul de biti transferați. Controllerul va rămâne într-o stare de așteptare până când transferul va fi incheiat (TransferDone=1). In tot acest timp, controllerul nu va mai putea prelua comenzi de Read/Write de la utilizator.
      - După terminarea transferului (TransferDone = 1) controller-ul va fi trimis în starea de Idle.



Serial Transceiver

Funcționare:

- Componenta va fi utilizată la transferul serial a datei eșantionate pe intrarea Din.
- Transferul se va realiza la frontul pozitiv al clock-ului ClkTx și la fel și Logica internă (evaluarea semnalelor DataIn, Sample, StartTx, TxDone)
- Se va implementa un transfer serial **PARAMETRIZABIL**. Un număr parametrizabil de biti transferați.



Interfața de intrare:

- **DataIn**:
  - Bus pentru datele de intrare.
- **Sample**:
  - Semnal de control pentru eșantionarea datelor de intrare primite pe bus-ul DataIn. Acest semnal va fi evaluat la frontul pozitiv de Clk. Nu pot fi active simultan Sample și StartTx.
- **StartTx**:
  - Semnal de intrare ce va comanda propagarea datelor stocate în registrul intern spre ieșirea serială Dout. Nu pot fi ambele semnale Sample si StartTx active.
- **Reset**:
  - Reset asincron. Odată activat, comandă sfârșitul unui transfer în curs, dar și resetarea bistabilelor interne.
- **Clk**:
  - Semnal de ceas pentru funcționare internă (Nu transfer serial). Evaluarea DataIn, Sample, StartTx, TxDone vor funcționa pe frontul pozitiv de ceas Clk.
- **ClkTx**:
  - Semnalul de clock pentru interfața serială pe care se va face transferul serial. Semnalele TxBusy, DataOut vor funcționa pe acest clock.

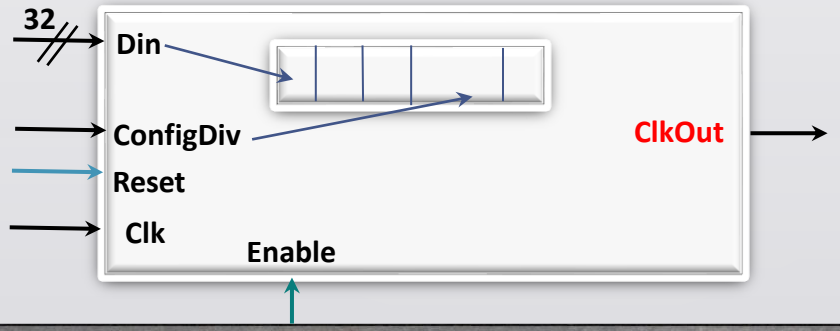
Interfața de ieșire:

- **TxBusy** :
  - Semnal activ pe parcursul unui transfer. Va fi asertat pe tot parcursul transferului începând cu bit-ul MSB. Când ultimul bit(LSB) a ajuns pe interfața serială semnalul TxBusy va fi activ până la următorul front pozitiv de ceas.
- **Dout**:
  - Serial Data Out . Datele stocate în registrul intern vor fi transferate la frontul pozitiv ClkTx. Transferul se va realiza începând cu MSB, LSB va fi ultimul bit transferat.

FrequencyDivider

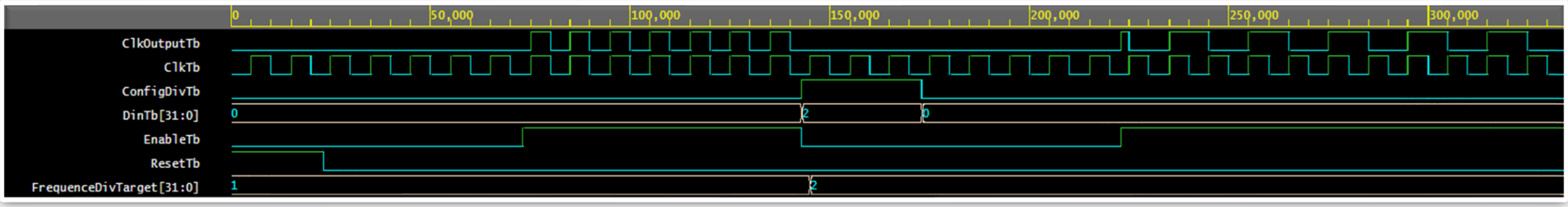
Mod de funcționare:

- Circuitul funcționează la frontul pozitiv al clock-ului Clk.
- Dacă Reset=1:
  - Intreg blocul logic se va reseta cât și registrul intern de configurare.
  - Frecvența de ieșire ClkOutput = Clk.
  - In tot acest timp ClkOut va fi 0 pe perioada resetului.
- Dacă Reset=0:
  - Dacă Enable=0:
    - Semnalul de iesire ClkOut = 0.
    - Dacă ConfigDiv = 0:
      - Semnalul ClkOut = T * Clk, unde ‘T’ este configurația memorată la ultima scriere în registru.
    - Dacă ConfigDiv = 1 :
      - Datele eșantionate la intrarea Din vor fi memorate în registrul intern de configurație.
  - Dacă Enable = 1:
    - Divizorul de frecvență va fi activ și va activa funcționarea semnalului ClkOut.

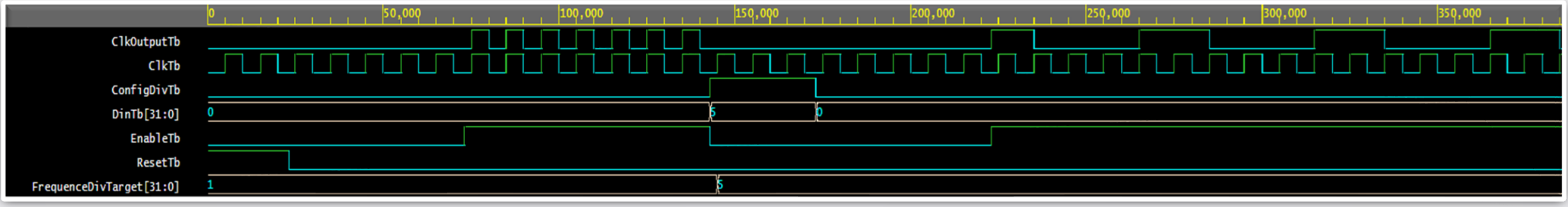


Exemple funcționare divizor de frecvență

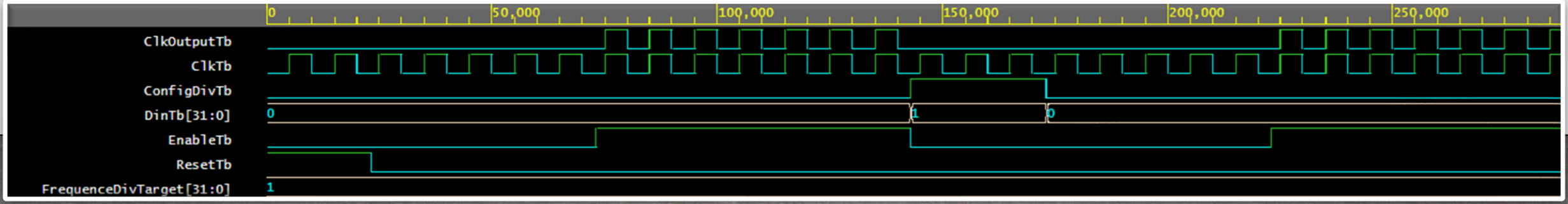
DIV = 2



DIV = 5



DIV = 1



Vă mulțumesc!