Controller Cache

de Efleih-Hassan Raian, Napau Dragos, Radu Bogdan

# Descriere Generala

In cadrul acestui proiect a fost realizat un cache controller complet functional, proiectat sa faca legatura intre procesor (simulat printr-un testbench) si memoria principala. Scopul arhitecturii este de a reduce timpul de acces la date, prin utilizarea unui sistem de cache eficient, asa cum este ilustrat in diagrama corespunzatoare.

A diagram of a computer

AI-generated content may be incorrect.

# Specificatii Tehnice

- Tipul cache-ului: 4-way set associative  
- Capacitate totala: 32 KB  
- Marimea unui block: 64 bytes (rezultand in 512 blockuri totale)  
- Dimensiunea unui cuvant: 4 bytes  
- Numar de seturi: 128  
- Politica de inlocuire: Least Recently Used (LRU)  
- Politica de scriere: Write-back cu write-allocate

# Componente Principale

- CPU/Testbench – trimite catre controller cereri de citire si scriere, alaturi de adrese.  
- FSM (Finite State Machine) – gestioneaza procesul de acces si control pe baza starilor sistemului si a semnalelor primite.  
- PA\_ADDRESS – moduleaza adresa fizica in componente (tag, index, offset).  
- CACHE MEMORY – pastreaza datele structurate in seturi si cai, folosind tag-uri si biti de validare.  
- MAIN MEMORY – se acceseaza in caz de cache miss, fie pentru citire, fie pentru write-back.

# Structura Adresei Fizice

A white rectangular object with black text

AI-generated content may be incorrect.

- Tag (19 biti) – identificatorul blocului, folosit la verificarea existentei datelor in cache.  
- Index (7 biti) – indica unul dintre cele 128 de seturi (2^7 = 128).  
- Block Offset (4 biti) – selecteaza un cuvant din cadrul unui block de 64 bytes (16 cuvinte de 4B).  
- Word Offset (2 biti) – adreseaza exact pozitia unui cuvant de 4 bytes.

# Organizarea Cache-ului

A close-up of a grid

AI-generated content may be incorrect.

Cache-ul urmeaza un model 4-way set associative, ceea ce inseamna ca:  
- Fiecare set contine 4 cai (ways) posibile pentru stocarea unui block.  
- Fiecare way include:  
 - Valid (1 bit) – semnaleaza daca linia contine date valide.  
 - Tag (19 biti) – identificatorul blocului.  
 - Data (64 bytes) – datele complete ale blocului.  
 - Dirty (1 bit) – indica daca linia a fost modificata.  
 - LRU (2 biti) – codifica ordinea de utilizare a fiecarei cai, pentru implementarea politicii LRU.  
In total, exista 128 seturi, fiecare avand 4 linii, deci 512 linii de cache in ansamblu.

# A diagram of a diagram AI-generated content may be incorrect.FSM – Fluxul Principal de Operatii

IDLE → VALID → TAG\_CHECK → READ\_HIT / WRITE\_HIT → UPDATE\_LRU → DONE  
→ MISS\_SELECT → DIRTY\_CHECK → WRITE\_BACK → ALLOCATE → READ\_ALLOC / WRITE\_ALLOC → UPDATE\_LRU → DONE

IDLE – asteapta o cerere de citire/scriere; la activare trece in VALID.

VALID – se verifica daca vreuna din cele 4 cai la indexul dat este valida. Daca niciuna nu este, se trece in MISS\_SELECT.

TAG CHECK – se compara tag-ul cautat cu cele existente. In caz de potrivire, se continua in READ\_HIT sau WRITE\_HIT.

READ HIT / WRITE HIT– se efectueaza operatia corespunzatoare si se marcheaza hit/miss. Urmeaza UPDATE\_LRU.

UPDATE LRU – actualizeaza LRU pentru a reflecta recenta utilizare.

MISS SELECT – alege fie o cale invalida, fie cea mai putin utilizata.

DIRTY CHECK – verifica daca linia selectata este dirty.

WRITE BACK – datele dirty sunt scrise in memoria principala.

ALLOCATE – datele noi sunt stocate in cache, linia este validata si dirty se seteaza pe 0.

READ ALLOC / WRITE ALLOC – se realizeaza operatia de citire/scriere dupa alocare. Urmeaza UPDATE\_LRU.

DONE – finalizeaza operatia si revine in starea IDLE.

# Testbench – Validare si Evaluare

**Structura:**

A screen shot of a computer

AI-generated content may be incorrect.- Initializarea semnalelor de control si date  
- Instantierea modulului cache\_controller  
- Mecanism de resetare  
- Bloc pentru contorizarea acceselor  
  
**Scenarii testate:**  
- READ HIT – citire dintr-un way valid.  
- WRITE HIT – scriere in cache pe o linie valida.  
- READ MISS – fortare miss prin invalidarea unei linii.  
- WRITE MISS – scriere pe un way invalid cu alocare noua.  
- WRITE MISS cu dirty – testeaza scrierea inapoi in memorie inainte de alocare.  
- READ MISS cu dirty – se verifica alocarea dupa curatarea liniei existente.