**Documentatie**

**FEATURE-URI IMPLEMENTATE**

**Despre ce este acest sistem?**

-> Realizarea unui sistem mai intelligent de semaforizare

**De ce ne dorim să implementăm acest sistem?**

-> Vrem să eficientizăm (d.p.d.v. al timpului) flow-ul traficului

**Cum?**

-> Dacă detectăm faptul că avem mai puține mașini pe o bandă a unui sens de drum, nu are rost să ținem foarte mult culoarea verde a semaforului respectiv; la fel, dacă știm că avem mai multe mașini pe un sens de drum, atunci trebuie să lăsăm culoarea verde pe o durată mai lungă de timp astfel încât să ne îndreptăm spre o fluidizare a traficului

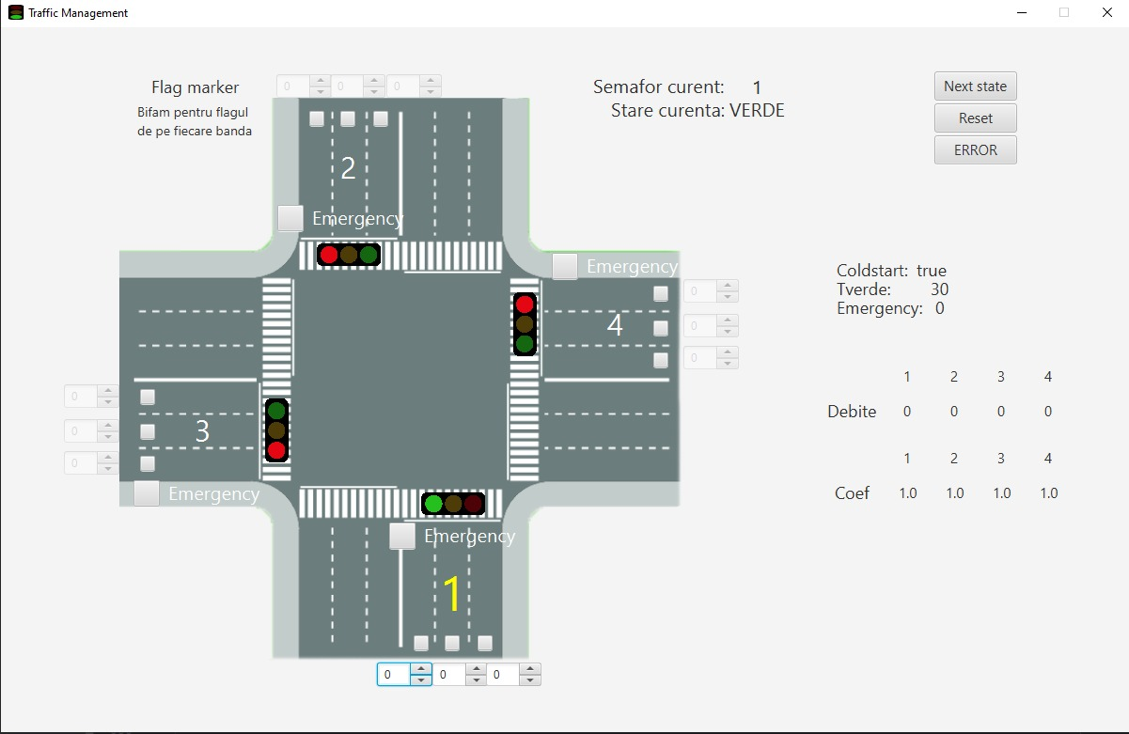
-> Timpul de verde al unui semafor este unul variabil: poate fi între Tmin și Tmax

-> În acest proiect regăsim și cazuri speciale de urgență; spre exemplu, în cazul unei ambulanțe, se va trimite un semnal din partea din care aceasta vine, iar culoarea semaforului va fi verde constant până când aceasta trece. După aceea, semnalul emitator va fi dezactivat. Totul revine la normal după trecerea acesteia.

-> Avem la bază și ideea de debit: pentru fiecare bandă/sens de mers vom avea un senzor (counter asincron) care adună (odată ce detectează o mașină, valoarea îi crește cu 1, iar în momentul respectiv va ști că pe acolo a trecut o mașină, fie că a făcut stânga, în față sau dreapta). Inițial, când se face verde, toți acești senzori vor fi puși pe 0/enable (se resetează fiecare counter de la semafor), iar cât timp este verde, ele tot adună. În momentul în care semaforul își schimbă culoarea în galben (dăm disable la countere/senzorii vor avea valoarea 0), valorile celor 3 counteri (1 counter pentru fiecare bandă a unui sens de mers) vor fi adunate într-un registru pentru semaforul actual, lucru care rezultă într-un debit specific pentru acel sens de mers.

-> La momentul de timp în care semaforul are culoarea galben, dăm disable la countere

-> Undeva mai în spatele intersecției, dar din nou pe fiecare bandă a sensului de mers, va exista câte un senzor care detectează dacă în dreptul lui este o mașină. Cu ce ne ajută acest lucru? Ne vor ajuta la determinarea unui factor de importanță, folosit pentru formula aflării timpului de verde necesar pentru acel semafor. Astfel vom detecta pe care bandă a sensului de mers avem o coloană mai lungă decât ar trebui (putem avea pe o bandă, pe două, pe toate 3, pe 2 din ele, pe niciuna etc)



A diagram of a diagram

Description automatically generated

**Schema Hardware**

Pentru a detalia această idee, vom ilustra un design care combină arhitectura Datapath cu modificări specifice pentru a se integra într-un sistem de management al traficului. Imaginea va prezenta circuite, unități de procesare, și alte componente tipice ale arhitecturii Datapath, dar și elemente unice care indică adaptarea pentru gestionarea traficului.

Vor fi reprezentate și conexiunile de date, poate sub forma unor linii sau cabluri care leagă diferitele componente. În plus, designul poate include simboluri sau reprezentări grafice ale unui sistem de management al traficului, cum ar fi semafoare, panouri de control, sau reprezentări abstracte ale fluxului de date în contextul gestionării traficului. Aceste elemente vor fi integrate într-o manieră care sugerează o interacțiune armonioasă și eficientă între arhitectura Datapath și cerințele unui sistem de management al traficului.

A blue and yellow diagram

Description automatically generated

**Datapath Arhitecture (**<https://en.wikipedia.org/wiki/Datapath> [Introduction of ALU and Data Path - GeeksforGeeks](https://www.geeksforgeeks.org/introduction-of-alu-and-data-path/)**)**

**Ce este o Cale de Date (Datapath) mai exact?**

Să presupunem că CPU-ul trebuie să efectueze orice acțiune de procesare a datelor, cum ar fi copierea datelor din memorie într-un registru și invers, mutarea conținutului registrelor de la un registru la altul sau adunarea a două numere în ALU. Prin urmare, de fiecare dată când are loc o acțiune de procesare a datelor în CPU, datele implicate în acea operație urmează o anumită cale, sau cale de date.

Căile de date sunt alcătuite din diverse componente funcționale, cum ar fi multiplexoare sau unitățile logice aritmetice (ALU). O cale de date este necesară pentru a efectua operațiuni de procesare a datelor.

O cale de date (datapath) reprezintă ALU, setul de registre și BUS-urile interne ale CPU-ului care permit fluxul de date între ele.

*O cale de date (datapath) a microarhitecturii organizată în jurul unui singur BUS*

O imagine care conține diagramă, Plan, Desen tehnic, schematic

Descriere generată automat

Cel mai simplu design pentru un CPU utilizează un singur BUS intern comun. Adunarea eficientă necesită o structură cu trei BUS-uri interne puțin mai complicată. Multe CPU-uri relativ simple au un fișier de registre cu 2 citiri și 1 scriere, conectat la cele 2 intrări și 1 ieșire ale ALU-ului.

La sfarsitul anilor '90, s-a intensificat cercetarea în domeniul căilor de date reconfigurabile - căi de date care pot fi reutilizate în timpul execuției, folosind fabrici programabile / [programmable fabric](https://en.wikipedia.org/wiki/Field-programmable_gate_array) - deoarece astfel de proiecte pot permite procesare mai eficientă și economii semnificative de energie.

ALU

În informatică, o unitate aritmetică și logică (ALU) este un circuit digital combinațional care efectuează operații aritmetice și operații pe biți pe numere binare întregi. Acesta este în contrast cu o unitate cu virgulă mobilă (FPU), care operează pe numere cu virgulă mobilă. Este un bloc de construcție fundamental al multor tipuri de circuite de calcul, inclusiv unitatea centrală de procesare (CPU) a calculatoarelor, FPUs și unitățile de procesare grafică (GPUs).

Intrările într-o ALU sunt datele asupra cărora se operează, numite operanzi, și un cod care indică operația ce urmează să fie efectuată; ieșirea ALU este rezultatul operației efectuate. În multe proiecte, ALU are, de asemenea, intrări sau ieșiri de stare, sau ambele, care transmit informații despre o operație anterioară sau operația curentă, respectiv, între ALU și registrele de stare externe.

A diagram of a computer

Description automatically generated

Intrări și Ieșiri: Liniile etichetate ca 'input' și 'output' sunt liniile de date care transportă informații binare către și dinspre ALU.Are doua input-uri.

Semnale de Control: Etichetele 'SelOP' sunt operații de selecție care determină ce operație ar trebui să execute ALU.

Operații: Etichetele precum 'Add', 'Sub', 'Mul', 'Div', 'And', 'Or', etc., reprezintă diferitele operații pe care ALU le poate efectua, cum ar fi adunarea, scăderea, înmulțirea, împărțirea, AND pe biți, OR pe biți, etc.

Carry, Zero, Negative, Overflow: Acestea sunt flaguri tipice pe care ALU le-ar putea seta în funcție de rezultatul operațiilor sale. De exemplu, steagul 'Carry' este setat dacă o operație aritmetică rezultă într-un carry out; 'Zero' este setat dacă rezultatul este zero; 'Negative' dacă rezultatul este negativ; și 'Overflow' dacă operația aritmetică duce la un overflow.

A diagram of a computer scheme

Description automatically generated

Semafoarele : descriem logica combinationala care verifica semnalele de la semafoare, care contine counterele si semnalele, facem asta pentru fiecare semafor. Fiecare scoate la output debitul total (prin cele 2 addere), tot timpul iesirile sunt calculate imediat si scoatem debitul total al lor, avem niste intrari care comunica in mod direct cu senzorii externi.

A diagram of a circuit

Description automatically generated

A diagram of a computer program

Description automatically generated

Selectam care semafoare sa fie rosii/galbene/verzi si avem index controller care ne permite sa controlam la ce semafor facem referire.

A computer screen shot of a diagram

Description automatically generated

Aici comparam flagurile si scoatem rezultatul dintre flagurile pe care trebuie sa le compare, cold start(bit-ul de cold) ne indica daca suntem intr o situatie de cold sau nu.

**Diferite secvente de Assembly**

; Inițializarea variabilelor

LI R1, 40 ; Tmax := 40

LI R2, 20 ; Tmin := 20

LI R3, 1 ; Kinit := 1

LI R4, 0 ; index := 0

LI R5, 1 ; cold := 1

LI R6, 30 ; Tmed := 30

LI R7, 20 ; Tdif := 20

LI R8, 30 ; Tverde := 30

LI R9, 0 ; Debit[0] := 0

LI R10, 0 ; Debit[1] := 0

LI R11, 0 ; Debit[2] := 0

LI R12, 0 ; Debit[3] := 0

LI R13, 0 ; Presupunem că emergency este stocat în R13

; Intrarea în bucla principală

main\_loop:

; Apel de funcție - set\_semafor\_verde(index)

MOVE R0, R4 ; Copiem index în R0 ca argument

CALL set\_semafor\_verde

; Apel de funcție - set\_enabled\_counter(index)

CALL set\_enabled\_counter

; Apel de funcție - reset\_counter(index)

CALL reset\_counter

; Bucată de cod pentru decrementarea și verificarea lui Tverde

CMP R8, #0

BEQ check\_emergency

SUB R8, R8, #1

BR main\_loop

; Presupunem că:

; R4 = index, R9-R12 = Debit[0-3], R1-R3 = Tmin, Tmax, Kinit

; R6 = Tmed, R7 = Tdif, R8 = Tverde, R5 = cold

; R13 = Suma\_debite, R14, R15, R16 = valori temporare pentru calcul

; Eticheta Galben

galben:

; Apel de funcție - set\_disabled\_counter(index)

MOVE R0, R4

CALL set\_disabled\_counter

; Calculul Debit[index]

; Presupunem că counter[index] este accesat printr-o subrutină

MOVE R0, R4

CALL get\_counter

MOVE R14, R0 ; Salvează rezultatul în R14

ADD R4, R4, #1

CALL get\_counter

ADD R14, R14, R0

ADD R4, R4, #1

CALL get\_counter

ADD R9, R14, R0 ; Presupunem că Debit[index] este în R9

SUB R4, R4, #2 ; Restabilește indexul

; Calculul Suma\_debite

ADD R13, R9, R10

ADD R13, R13, R11

ADD R13, R13, R12

; Calculul K[index+1]

ADD R4, R4, #1

; Presupunem că cmp\_importanta este o subrutină

MOVE R0, R4

CALL cmp\_importanta

MOVE R14, R0

ADD R4, R4, #1

CALL cmp\_importanta

ADD R14, R14, R0

ADD R4, R4, #1

CALL cmp\_importanta

ADD R14, R14, R0

ADD R14, R14, R3 ; Adaugă Kinit

; Presupunem că K[index+1] este stocat în R14

; Condiții și calculul lui Tverde

CMP R5, #1

BEQ set\_Tverde\_to\_Tmed

DIV R8, R9, R13 ; Tverde = Debit[index+1] / Suma\_debite

MUL R8, R8, R7 ; Tverde = Tverde \* Tdif

MUL R8, R8, R14 ; Tverde = K[index+1] \* Tverde

ADD R8, R8, R2 ; Tverde = Tmin + Tverde

BR check\_Tverde\_limits

set\_Tverde\_to\_Tmed:

MOVE R8, R6

; Continuă cu verificarea limitelor

check\_Tverde\_limits:

CMP R8, R2

BLT set\_Tverde\_to\_Tmin

CMP R8, R1

BGT set\_Tverde\_to\_Tmax

BR continue\_processing

set\_Tverde\_to\_Tmin:

MOVE R8, R2

BR check\_emergency2

set\_Tverde\_to\_Tmax:

MOVE R8, R1

; Continuă cu procesarea ulterioară

; Restul codului, inclusiv check\_emergency2 și alte etichete

; ...