

# Laboratorul 5 - Debugging skills

În laboratoarele anterioare au fost prezentate elementele de limbaj Verilog și de descriere comportamentală a două tipuri de circuite: secvențiale și combinatoriale. Acest laborator se va concentra pe modul în care ne asigurăm că un circuit are funcționalitatea dorită.

În acest laborator vom corecta erorile dintr-un adder8 care i-am oferit o funcționalitate diferită decât a celorlalte module de tip adder implementate în laboratoarele anterioare.

## Descrierea funcțională a modului

### Sumar

Modulul adder8 va primi la intrare un pachet de date ce conține termenii adunării pe 8 biți, și va returna la ieșire suma tuturor termenilor. Modulul va avea un semnal de error, care va fi asertat în cazul în care se detectează vreo eroare (i.e. overflow).

### Interfete

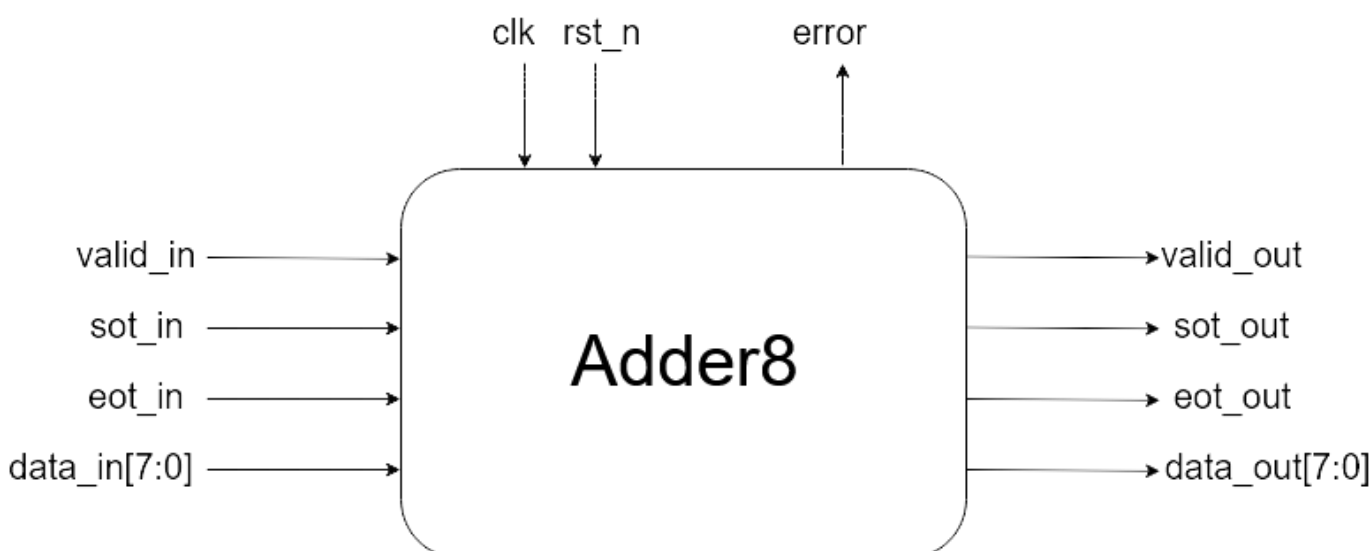


Fig. 1: Adder8

### Interfața de intrare

Port	Width (bits)	Directie	Valoare la reset	Descriere
------	--------------	----------	------------------	-----------

data_in	8	in	0	Folosit pentru a trimite termenii operatiei de adunare
valid_in	1	in	0	Valideaza un byte de date
sot_in	1	in	0	Marcheaza inceputul unei tranzactii. Va fi activ un singur ciclu de ceas
eot_in	1	in	0	Marcheaza sfarsitul unei tranzactii. Va fi activ un singur ciclu de ceas

Tab. 1: Interfata de Intrare

### Interfata de iesire

Port	Width (bits)	Directie	Valoare la reset	Descriere
data_out	8	out	0	Folosit pentru a trimite rezultatul operatiei de adunare.
valid_out	1	out	0	Valideaza un byte de date
sot_out	1	out	0	Marcheaza inceputul unei tranzactii. Va fi activ un singur ciclu de ceas
eot_out	1	out	0	Marcheaza sfarsitul unei tranzactii. Va fi activ un singur ciclu de ceas

Tab. 2: Interfata de Iesire

### Interfata de Status

Port	Width (bits)	Directie	Valoare la reset	Descriere
error	1	out	0	Folosit pentru a marca o eroare ce s-a intamplat in timpul realizarii operatiei de adunare.

Tab. 3: Interfata de Status

## Functionalitate

Modulul va primi unul sau mai multi bytes de date, va realiza operatia de adunare intre toti termenii, si apoi va pune rezultatul pe interfata de iesire. Rezultatul va fi reprezentat pe maximum 10 biti.

Deoarece latimea semnalului de iesire este de doar 8 biti, intreg rezultatul va fi pus pe interfata de iesire in doi cicli consecutivi (vezi exemplu de la capitolul Protocol de iesire).

In cazul in care rezultatul nu poate fi reprezentat pe 10 biti, semnalul de eroare va fi asertat, iar toti bitii sumei vor fi pusi pe 1. Semnalul de eroare trebuie asertat pe intreaga lungime a pachetului de iesire.

Tranzitiile RTL-ului (RTL = register-transfer level sau modul) si citirea semnalelor de intrare se va face pe frontul pozitiv al ciclului de ceas.

Resetul este activ pe 0, blocul va fi considerat in reset cand resetul este 0, si va functiona normal cand resetul este 1.

## Protocol

## Protocol de intrare

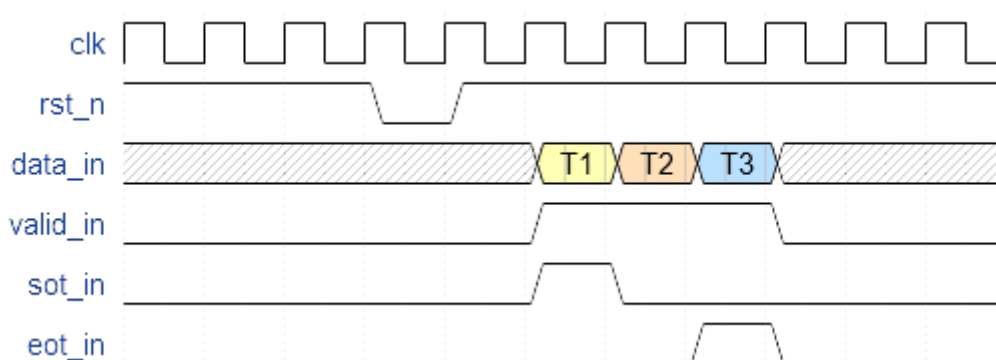


Fig. 1: O tranzactie de intrare

Blocul nu poate accepta tranzactii consecutive. Acesta are nevoie de un delay de minim un ciclu de ceas intre 2 intrari consecutive

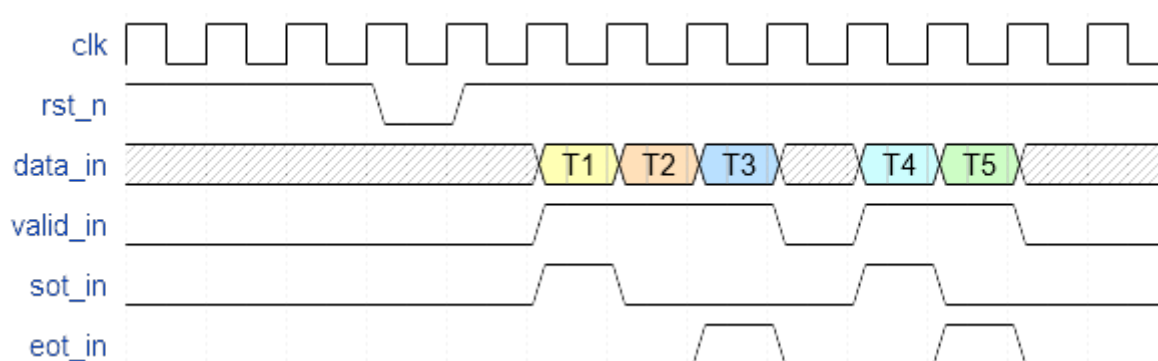


Fig. 1: Distanța între tranzactii consecutive

## Protocol de iesire

1) Pentru un rezultat pe maxim 8 biti

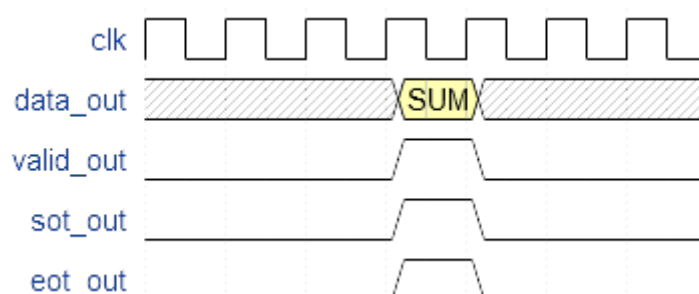


Fig. 1: Pentru un rezultat pe maxim 8 biti 2) Pentru un rezultat pe maxim 10 biti

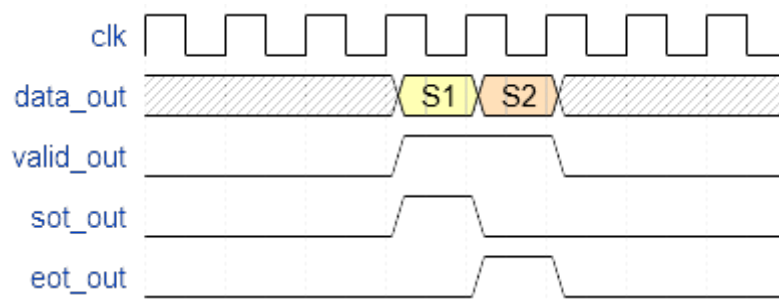


Fig. 1: Pentru un rezultat pe maxim 10 biti

Rezultatul operatiei va fi obtinut prin concatenarea S1 si S2;  
 $\text{Sum} = \{S1, S2\}'$

### 3) Pentru overflow

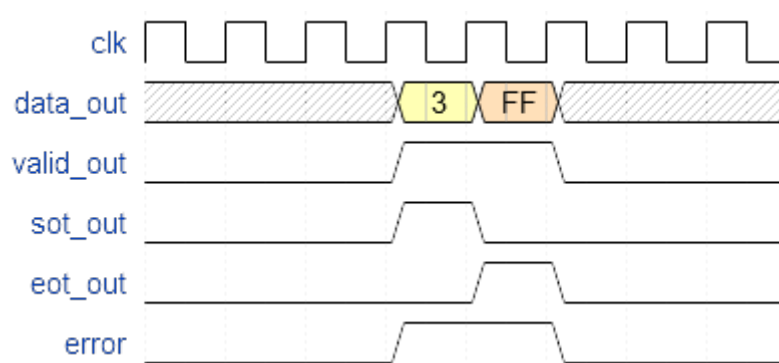


Fig. 1: Overflow

Observatie: Semnalul de eroare este 1 pe durata intregului transfer de date. Toti cei 10 biti ai iesirii sunt pusi pe 1.

## Exemplu de transfer

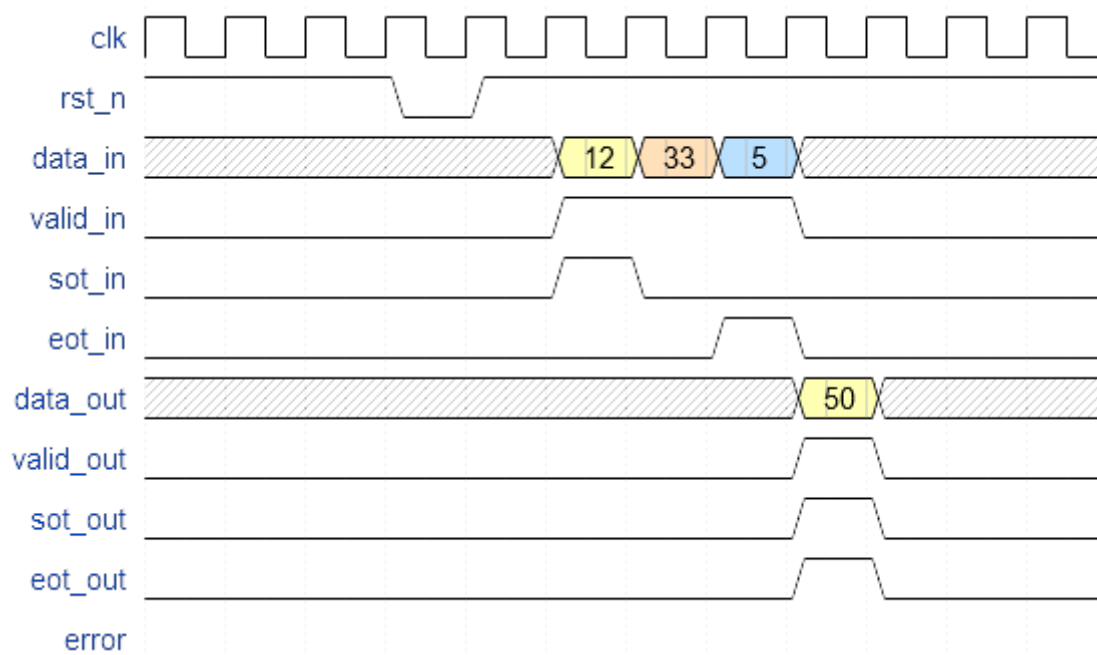


Fig. 1: Exemplu de transfer

## Diagrama de stari

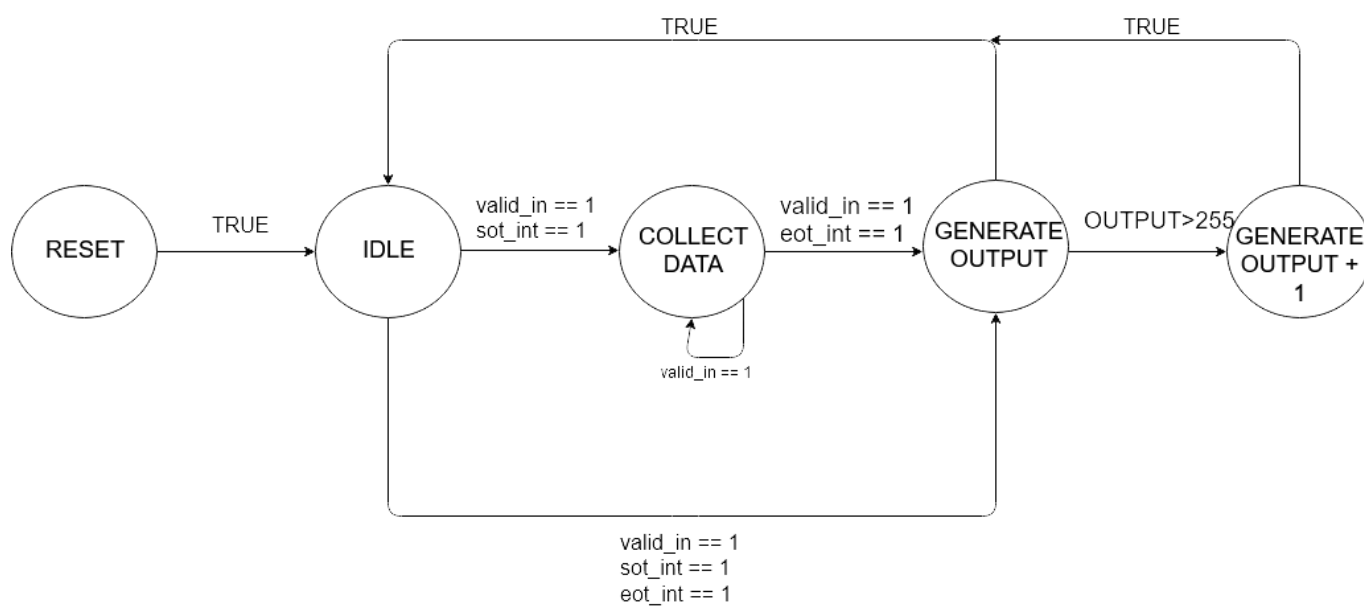


Fig. 1: Diagrama de stari

## Tehnici de debug

### Logs

Cel mai rapid mod de a identifica o eroare este prin a verifica log-ul de simulare. Acesta e salvat in fisierul **isim.txt** sau poate fi urmarit direct in consola din simulator.

Codul trebuie sa contina mesaje de debug pentru ca log-urile sa fie utile. Aceste mesaje pot fi scrise folosind functiile de sistem:

- `$display()`
- `$warning()` - nu e suportat de Xilinx ISE
- `$error()` - nu e suportat de Xilinx ISE

Un mesaj simplu precum `$display("[ADDER_STATE_STATUS]: Entering state IDLE")`, poate ajuta la debug-ul unei stari de **DEADLOCK** sau o tranzitie gresita.

## Waveforms

Diagramele cu forme de unda (waveforms) sunt unealta principala de debug folosita, complementand log-ul. Acestea iti arata modul in care s-au modificat semnalele si variabilele locale in timp, pe tot parcursul simularii.

Un caz tipic de debug este urmatorul:

1) Testul pica cu urmatorul mesaj de eroare:

```

Simulator is doing circuit initialization process.
Finished circuit initialization process.
[Test 0 PASSED]: Sum correctly checked!
Expected sum = 71, received 71
[ERROR]: error signal not asserted thorough out the entire transaction
[Test 1 FAILED]: Expected error, but did not receive error
Stopped at time : 770 ns : in File "H:/Users/Vlad/Desktop/AC/ex1_sol/average_test.v" Line 168
ISim>

```

Fig. 1: Console error 2) Datorita log-ului ne uitam pe semnalul de eroare din simulare (marcat cu visiniu in imaginea de mai jos)

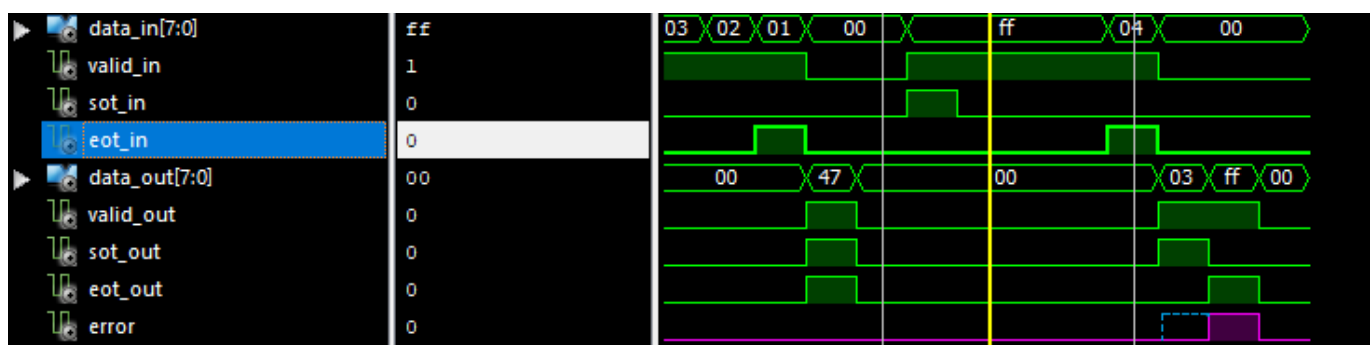


Fig. 1: Waves

Note: Pentru a intelege functionalitatea descrisa, cititi capitolul Descrierea functionala a modulului si urmariti scheletul de cod.

3) Din diagrama formelor de unde ne dam seama ca semnalul nu este asertat in starea in care trebuie. Vom adauga variabila **state** in diagrama.

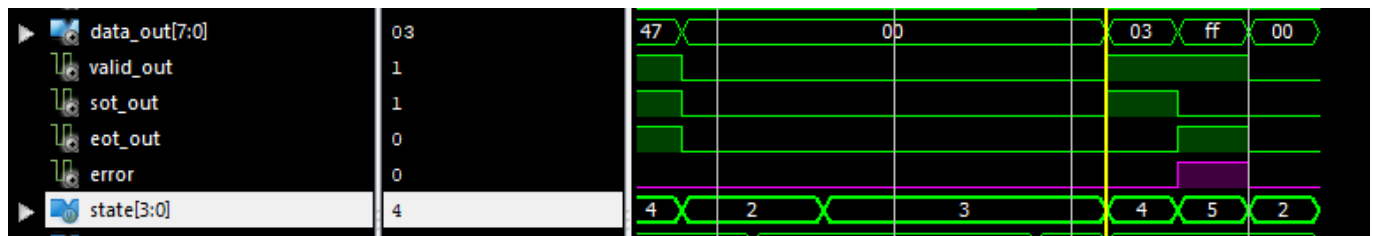


Fig. 1: Waves

4) Observam ca in starea **4 (GENERATE\_OUTPUT)** semnalul **error** nu se aserteaza asa cum este de asteptat, de aici ne ducem in cod si observam ca lipseste linia: **error\_reg = overflow;**

5) Corectam codul:

```
`GENERATE_OUTPUT: begin
    {data1, data2} = accumulator;

    error_reg = overflow;
    valid_out_reg = 1;
    sot_out_reg = 1;

    if (data1 == 0)
    begin
        eot_out_reg = 1;
        data_out_reg = data2;
        next_state = `IDLE;
    end
    else
    begin
        data_out_reg = data1;
        next_state = `GENERATE_OUTPUT + 1;
    end
end
```

Fig. 1: Cod corectat

6) Resimulam si obtinem comportamentul dorit

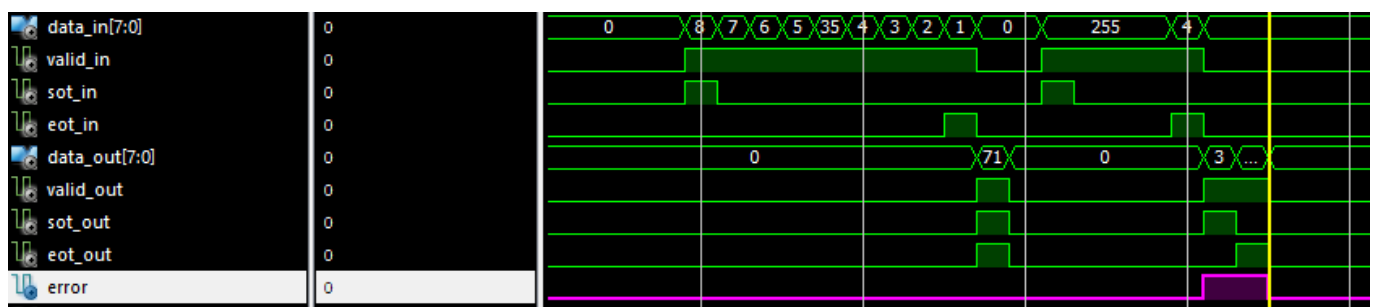


Fig. 1: Waves

## Breakpoints

In cazul in care niciuna din metodele de mai sus nu merg, se vor pune breakpoint-uri direct in cod, pentru a verifica starea simularii intr-o anumita zona din cod. Aceasta solutie nu este una eficienta, si

nu este viabila atunci cand codul devine mai mare. Pentru mai multe detalii urmariti [tutorialul de debugging](#).

## Teste

Niciuna din metodele de mai sus nu ar functiona daca nu ar exista teste care sa stimuleze modulul si sa monitorizeze activitatea de pe interfata de iesire.

Pentru fiecare functionalitate a blocului ar trebui scris un test. In cazul adder8 vom avea nevoie minim de:

- Un test care sa realizeze suma a N numere, iar rezultatul este pe 8 biti
- Un test care sa realizeze suma a N numere, iar rezultatul este pe 10 biti
- Un test care sa realizeze suma a N numere, iar rezultatul este pe mai mult de 10 biti
- Un test in care sa trimitem mai multe tranzactii consecutive
- Un test in care sa trimitem foarte multi termeni ( $N \geq 40$ )
- Un test in care sa trimitem un singur termen.

In cadrul scheletului de cod, in fisierul adder8\_test vom avea un task numit **drive** care se ocupa de stimularea corespunzatoare a modulului. Task-ul **monitor**, interpreteaza semnalele de output si verifica corectitudinea acestora.

## Exercitii

- **(9p)**. Rulati scheletul de cod si corectati erorile care apar. **(6×1.5p)**

- Hint: Testele se vor termina fie cu eroare, fie se vor bloca
- Hint: Intreg exercitiul ruleaza timp de 2.2us.

- **(1p)**. Pe modelul testelor din schelet, scrieti un nou test ce verifica o functionalitate a adder8 ce nu a fost explorata inca.

- **(Bragging rights) 4p** . Prin testul dezvoltat la exercitiul 2 se gaseste o eroare de functionare a adder8.

- Conditii de acordare a bonusului:
- Trimiteti un mail catre asistentul de laborator (cu [sergiu.a.duda@gmail.com](mailto:sergiu.a.duda@gmail.com) in CC) cu
  1. Descrierea erorii
  2. Pasii de reproducere (vectorul de test folosit)

Eroarea sa nu fi fost raportata inainte de un alt coleg (indiferent de grupa sau serie)



## Resurse

- [Schelet de cod](#)
- [Soluție laborator](#) (disponibilă începând cu 02.11.2019)
- [PDF laborator](#)

From:

<https://elf.cs.pub.ro/ac/wiki/> - **AC Wiki**

Permanent link:

<https://elf.cs.pub.ro/ac/wiki/lab/lab05>

Last update: **2019/12/13 14:10**

