

# Tema 2 - Maze Solver

- Responsabili: Giorgiana Vlăsceanu, Sergiu Dudă, Alexandru Popa
- Deadline soft (fără penalizări): **1.12.2017** ora **23:59**; Deadline hard: **08.12.2017** ora **23:59**
- Data publicării: 19.11.2017
- Data ultimei actualizări: 27.11.2017, 22:00
- Istoric modificări:
  - 19.11.2017:
    - Actualizat data publicării
  - 24.11.2017:
    - skel/test.data fixed
    - Adăugat clafiricări legate de directia de plecare
    - Adăugat exemplu pentru input și output
  - 25.11.2017:
    - Prelungire deadline
  - 27.11.2017:
    - Adăugat tester online și offline
    - Rezolvat problema la citirea indicilor starting\_row si starting\_col
    - Rezolvat afișarea numelor testelor

## Obiectiv

Tema are ca scop exersarea lucrului cu noțiunile de Verilog folosite pentru proiectarea circuitelor secvențiale.

## Descriere și cerințe

Implementați în Verilog un circuit secvențial sincron care parcurge un labirint de dimensiune 64×64.

Inițial fiecare punct din labirint poate fi descris de stări notate cu 0 și respectiv 1. Culoarul labirintului este descris de punctele marcate cu 0, iar pereții labirintului sunt marcați cu 1. Fiecare punct parcurs prin labirint va fi marcat cu 2, descriind astfel traseul dintre punctul de start și ieșirea din labirint. Regula de parcurgere a labirintului este dată de algoritmul Wall Follower cu urmărirea peretelui drept.

## Implementare

Automatul cu stări finite care va modela comportamentul general trebuie implementat în modulul maze, având interfața:

```
module maze(  
input          clk,  
input [maze_width - 1:0] starting_col, starting_row,    // indicii  
punctului de start  
input          maze_in,          // oferă informații despre  
punctul de coordonate [row, col]  
output [maze_width - 1:0] row, col    // selectează un rând și o  
coloană din labirint  
output          maze_oe;          // output enable (activează  
citirea din labirint la rândul și coloana date) - semnal sincron  
output          maze_we;          // write enable (activează  
scrierea în labirint la rândul și coloana date) - semnal sincron  
output          done);          // ieșirea din labirint a fost  
gasită; semnalul rămâne activ
```

Descrierea semnalelor folosite de acest modul este următoarea:

- `clk` - semnal de sincronizare
- `starting_row`, `starting_col` - coordonatele punctului de start pentru labirint
- `row` - indexul rândului punctului care se dorește a fie citit/scriș
- `col` - indexul coloanei punctului care se dorește a fie citit/scriș
- `maze_in` - semnal care conține informația punctului de coordonate [row,col]
- `maze_oe` - când este activ returnează pe firul `maze_in` informația de la coordonatele [row,col]
- `maze_we` - când este activ marchează cu 2 în labirint poziția desemnată de indicii [row,col]
- `done` - indică faptul că s-a terminat parcurgerea labirintului; trebuie să fie activ după găsirea ieșirii din labirint

Punctul de start poate fi localizat oriunde în labirint.

Din punctul de start există o singură direcție de plecare (vezi exemplul de input).

Este permisă modificarea declarării ieșirilor în `output reg`.

### Exemplu de input pentru un labirint 8x8

[test.data](#)

```
starting_row=1  
starting_col=1  
1 1 1 1 1 1 1 1  
1 0 1 1 0 0 0 1  
1 0 1 1 0 1 0 1  
1 0 1 1 0 1 0 1  
1 0 0 0 0 1 0 1  
1 1 0 1 1 1 0 1  
1 1 0 1 0 0 0 1
```

```
1 1 1 1 0 1 1 1
```

## Exemplu de output pentru un labirint 8x8

test.out

```
1 1 1 1 1 1 1 1
1 2 1 1 2 2 2 1
1 2 1 1 2 1 2 1
1 2 1 1 2 1 2 1
1 2 2 2 2 1 2 1
1 1 2 1 1 1 2 1
1 1 2 1 2 2 2 1
1 1 1 1 2 1 1 1
```

## Algoritmul Wall Follower

Algoritmul Wall Follower este unul dintre cei mai cunoscuți algoritmi ce propun soluționarea unui labirint. Este recunoscut și după regula de dreapta sau de stânga.

În cazul în care labirintul este simplu conectat, urmărirea peretelui din dreapta, de exemplu, asigură că se ajunge la o ieșire diferită dacă există una, în caz contrar o să se întoarcă la intrarea de la care a pornit. Dacă labirintul nu este pur și simplu conectat, această metodă nu va atinge obiectivul. Mai multe despre algoritm în resurse.

## Bonus

Se va acorda bonus pentru implementarea într-un număr minim de cicluri. Modalitatea aleasă va fi motivată în cadrul fișierului de README.

## Observații

- Nu este permisă cache-uirea labirintului (citirea și salvarea acestuia în cadrul modulului pentru procesare ulterioară).
- Operațiile de citire / scriere sunt făcute sincron; răspunsul operației vine în ciclul următor.
- Labirintul are o singură ieșire.

## Precizări

- Arhiva temei (de tip **zip**) trebuie să cuprindă în rădăcina sa (**fără alte directoare**) doar:
  - fișierele sursă (extensia .v)
  - fișierul README
- Arhiva **nu** trebuie să conțină fișiere de test, fișiere specifice proiectelor etc.
- Fișierului README va conține minim:
  - numele și grupa
  - prezentarea generală a soluției alese (ex: diagrama automatului implementat)
  - explicarea porțiunilor complexe ale implementării (poate fi făcută și în comentarii)
  - alte detalii relevante
- Vmchecker ne permite să revenim la orice soluție încărcată de voi; cereți revenirea la cea mai convenabilă soluție trimisă (punctaj teste automate + depunctare întârziere) printr-un [mail](#) responsabilului de temă
- Nu se oferă suport pe forum/mail/office hours pentru teme care nu abordează automatul secvențial și combinațional și nu folosesc atribuirile conform recomandărilor. Este obligatoriu la sesiuni de consultații din cadrul office hours să aveți diagrama de stări a automatului implementat.
- Tema trebuie realizată individual; folosirea de porțiuni de cod de la alți colegi sau de pe Internet (cu excepția site-ului de curs) poate fi considerată copiere și va fi penalizată conform [regulamentului](#).

## Notare

- 10 pct: corectitudine
- +2 pct: implementare într-un număr minim de cicluri
- -10 pct: cache-uirea labirintului
- -10 pct: folosirea construcțiilor nesintetizabile din Verilog (while, repeat, for cu număr variabil de iterații etc.)
- -1 pct: lipsa fișierului README.
- -0.5 pct: pentru fiecare zi de întârziere; tema poate fi trimisă cu maxim 7 zile întârziere față de termenul specificat în enunț (față de deadline-ul soft).
- -0.2 pct: folosirea incorectă a atribuirilor continue (assign), blocante (=) și non-blocante (<=).
- -0.2 pct: indentare haotică
- -0.2 pct: lipsa comentariilor **utile**
- -0.1 pct: comentarii inutile (ex. wire x; // semnalul x)
- -0.2 pct: diverse alte probleme constatate în implementare (per problemă)

Dacă tema primește 0 pe *vmchecker*, se pot acorda maxim 2 puncte pe ideea implementării, la latitudinea asistentului. Ideea și motivele pentru care nu funcționează trebuie documentate temeinic în README și/sau comentarii. Temele care au erori de compilare vor fi notate cu 0 puncte.

## Resurse

- [Schelet](#)
- [Tester offline](#)

- [PDF temă](#)
- [Wikipedia - Maze Solving Algorithm](#)
- [Ghidul studentului la AC](#)
- [Utilizarea vmchecker](#)
- [Debugging folosind Xilinx ISE](#)

From:

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

Permanent link:

<https://elf.cs.pub.ro/ac/wiki/teme/tema2>

Last update: **2017/11/27 20:00**

