Ciuciu Anca-Maria 333AA

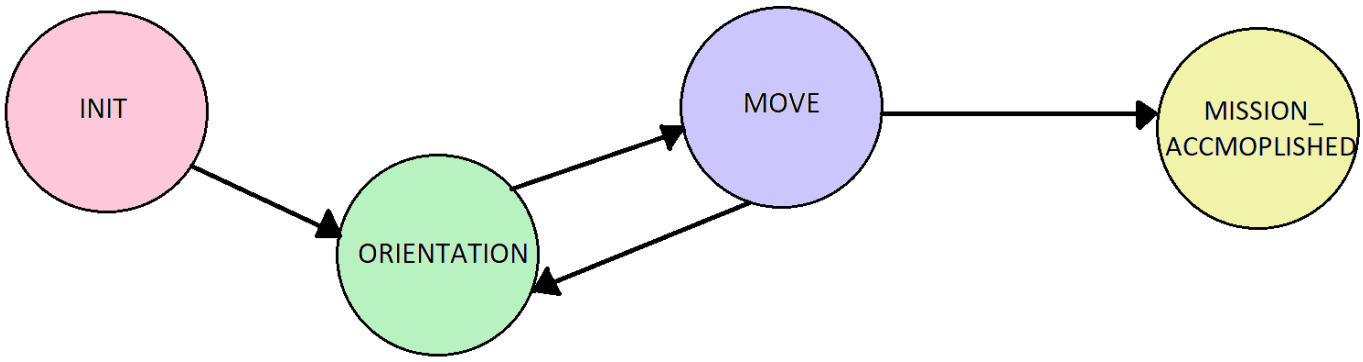
# Tema 2 – Maze

Obiectivul principal al acestei teme constă în rezolvarea unui labirint, compus din 1 și 0, cele două stări reprezentând peretele respectiv drumul liber. Totodată, pentru marcarea drumului, 0 va face tranziția in 2. Așadar, proiectarea soluției a fost posibilă prin intermediul unui state-machine alcătuit din 4 stări: *init*, *orientation*, *move* și *mission_accomplished*, orientarea realizându-se cu ajutorul marker-elor UP(0), LEFT(1), DOWN(2), RIGHT(3). Modulul de la care s-a început implementarea problemei este descris atfel:

```
module maze(
input                       clk,
input [maze_width - 1:0]    starting_col, starting_row,
input                       maze_in,
output [maze_width - 1:0]   row, col,
output                      maze_oe,
output                      maze_we,
output                      done);
```

- *clk* reprezintă semnalul de ceas
- *starting_row* și *starting_col* compun coordonatele inițiale pe care le vom avea in labirint, accesându-se în *init*
- cu ajutorul lui *maze_in* se interoghează matricea pentru a vedea daca drumul este blocat (1) sau nu (0)
- *row* și *col* au fost folosite pentru identificarea liniei si coloanei curente în matrice
- *maze_oe* activează citirea pentru a se putea accesa *maze_in* la următorul semnal de ceas iar *maze_we* activează scrierea pentru a marca celulele vizitate deja
- semnalul *done* este inițial 0, comutând pe 1 în momentul în care se găsește ieșirea din labirint în starea *mission_accomplished*

# Finite-state machine




Acest automat a fost implementat folosind două blocuri always@ cu ajutorul cărora s-a modelat partea de stare și respectiv, logica combinațională a acestuia. Așadar, elementele de stare ale acestui circuit s-au modelat pe frontul pozitiv al semnalului de ceas, *always @(posedge clk)* în care se verifică dacă misiunea a fost îndeplinită, iar în caz contrar, se va trece la următoarea stare din automat, adică la *next_state*, o variabilă de tip reg pe 3 biți (la fel ca *state*) deoarece automatul conține 4 stări. În continuare, partea combinațională a acestei rezolvări se concentrează pe desfășurarea logică a stărilor după cum urmează:

## 1. Starea INIT

Această stare presupune inițializarea liniei si coloanei curente cu *starting_row* și *starting_col*, două variabile ce reprezintă poziția inițială în care ne aflăm în labirint. Totodată , se va marca această celulă cu 2 prin intermediul comenzii de pe wite enable: *maze_we = 1*, și se va trece la starea următoare ce realizează o calibrare a entității prinsă în labirint.

## 2. Starea ORIENTATION

Având în vedere faptul că nu putem avea o perspectivă ortografică de sus asupra labirintului, se vor salva temporar coordonatele curente in variabilele reg pe 6 biți *last_row* și *last_col* în eventualitatea unei retrogradări. Totodată se va activa citirea pentru următoarea stare cu ajutorul comenzii: *maze_oe = 1.* Ar trebui menționat faptul că direcția inițială a entității este DOWN. Astfel, în cazul întâlnirii unei celule marcate cu 1 (zid), se va face o rotire de 90 de grade recalculându-se linia și coloana. Algoritmul folosit pentru ieșirea din labirint este Wall Follower Right astfel că, din direcția inițială DOWN, în această stare se va face tranziția spre RIGHT, urmând ca în starea următoare să se interogheze starea firului *maze_in.*

## 3. Starea MOVE

În această stare se va interoga *maze_in* astfel că, în cazul în care acesta este 1, adică drum blocat, va reveni la pozițiile anterioare. Totuși, dacă ”intuiția” a fost una productivă, și acea poziție este liberă, se analizează inerția entității descrisă de direcția acesteia, calculată pe baza relației dintre poziția anterioară și poziția curentă și efectuează o rotație de 90 de grade dreapta pentru a urmări peretele corespunzător, noua direcție fiind reținută în variabila de tip reg pe 2 biți *direction*.

De asemenea, în cazul în care entitatea depășește bordura matricei de 64x64, succesul este asigurat, ieșirea din labirint a fost în cele din urmă depistată – lucru constatat în următoarea stare.

## 4. Starea DONE

În această stare, variabila *done* se va face 1, semn că s-a terminat numărătoare pentru ciclii de ceas, ieșirea a fost gasită, misiunea a fost îndeplinită.

Inspirație pentru această temă:  Minecraft – Maze solving alg. – right hand rule

https://www.youtube.com/watch?v=vZ0h899Axr4

```
continue = true

while continue do
 turtle.turnRight()

 if turtle.detect() then
  turtle.turnLeft()
  a = turtle.forward()

  if a == false then
   turtle.turnLeft()
  end

 else
  turtle.forward()
 end

 local success, data = turtle.inspect()
 if data.name == "minecraft:grass" then
  continue = false
 end

end
```