Tema 2 - Maze

- Responsabili: Sergiu Dudă, George Flurche
- Deadline soft (fără penalizări): **05.12.2021**, ora **23:59**
- Deadline hard (cu penalizări): **12.12.2021**, ora **23:59**
- Data publicării: **22.11.2021**
- Data ultimei actualizări: 22.11.2021, 15:30
- Istoric modificări:
  - 22.11.2021
    - Publicare temă + checker offline
    - Update cu privire la modalitatea de notare
  - 24.11.2021
    - Adaugare exemplu input-output
  - 30.11.2021, 21:45
    - Imbunătățirea checker-ului online pentru a evita desincronizarea semnalelor de input
  - 01.12.2021, 13:26
    - Funcționalitatea de debug show_world funcțională; update doar checker offline

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 si 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 rămane 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/scris
- `col` - indexul coloanei punctului care se dorește a fie citit/scris
- `maze_in` - semnal care conține informația punctului de coordonate [row,col]
- `maze_oe` – cand este activ returneaza pe firul maze_in informatia de la coordonatele [row,col]
- `maze_we` – când este activ marchează cu 2 în labirint poziția desemnata de indicii [row,col]
- `done` - indică faptul că s-a terminat parcurgerea labirintului; trebuie să fie activ după găsirea ieșirii din labirint

Este permisă modificarea declarării ieșirilor în `output reg`.

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.

Exemplu input pentru un labirint 8x8

```
starting_row = 1
starting_col = 1

   0 1 2 3 4 5 6 7
  _______________
0 |1 1 1 1 1 1 1 1
1 |1 0 1 0 0 0 0 0
2 |1 0 1 0 1 1 1 1
3 |1 0 0 0 1 1 0 1
4 |1 1 0 1 1 0 0 1
5 |1 1 1 1 1 0 1 1
6 |1 1 0 0 0 0 1 1
7 |1 1 1 1 1 1 1 1
```

Exemplu output pentru un labirint 8x8

```
   0 1 2 3 4 5 6 7
  _______________
0 |1 1 1 1 1 1 1 1
1 |1 2 1 2 2 2 2 2
2 |1 2 1 2 1 1 1 1
3 |1 2 2 2 1 1 0 1
4 |1 1 2 1 1 0 0 1
5 |1 1 1 1 1 0 1 1
6 |1 1 0 0 0 0 1 1
7 |1 1 1 1 1 1 1 1
```

Ruta folosind algoritmul Wall Follower:

(1,1) → (2,1) → (3,1) → (3,2) → (4,2) → (3,2) → (3,3) → (2,3) → (1,3) → (1,4) → (1,5) → (1,6) → **(1,7)**

Bonus

Se va acorda bonus pentru implementarea într-un număr minim de ciclii.

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 citite / 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: descrierea de nivel înalt a algoritmului folosit);
  - 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 titularului de laborator.
- Tema trebuie realizată individual; folosirea de porțiuni de cod de la alți colegi sau de pe Internet (cu excepția site-ului de curs și a resurselor puse la dispoziție în conținutul temei) poate fi considerată **copiere** și va fi penalizată conform regulamentului.
- Aduceți-vă aminte de **recomandările** prezentate în Tema 0.

Notare

- 10 pct: corectitudine
- +2 pct: implementare într-un număr minim de ciclii
- -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, blocante și non-blocante
- -0.2 pct: indentare haotică
- -0.2 pct: lipsa comentariilor **utile**
- -0.1 pct: comentarii inutile
- -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

- Tester
- Wikipedia Maze-solving Algorithm [https://en.wikipedia.org/wiki/Maze-solving_algorithm#Wall_follower]
- Verilog 2000 Standard [https://sutherland-hdl.com/papers/2001-SNUG-presentation_Verilog-2000_standard.pdf]
- Ghidul studentului la AC [https://ocw.cs.pub.ro/courses/ac-is/studentguide]
- Utilizarea vmchecker [https://ocw.cs.pub.ro/courses/ac-is/tutoriale/6-vmchecker-utilizare]
- Debugging folosind Xilinx ISE [https://ocw.cs.pub.ro/courses/ac-is/tutoriale/3-ise-debug]