CS2102 02
AntVengers!

黃稚存

國立清華大學
資訊工程學系

HW 07

Ant-Man Image: MarvelHeroes.com

Maze: Shutterstock

Ant-Man!

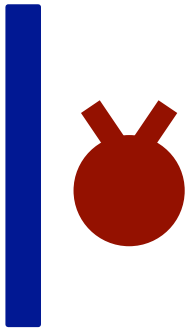
- You woke up and found yourself becoming an Ant-Man (or Wasp?!), being trapped in a dark Maze Universe!!
  - You lost the eyesight!
  - Instead, you got two antennae, left and right.
- You need to escape the Maze Universe (and save the world, of course)!
  - Using two antennae you can sense the Maze walls.

Image Source:
MarvelHeroes.com

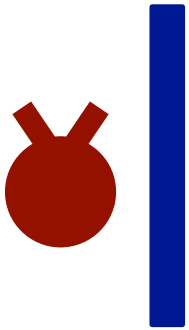
How The Antennae Work

2-bit Antenna Signal = `{ant_l, ant_r}` (or Simply {L, R})

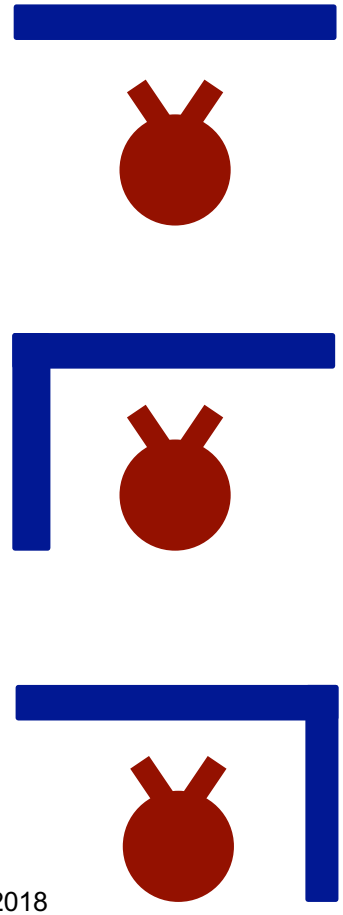
Case 1
Wall to the Left
LR = 10



Case 2
Wall to The Right
LR = 01



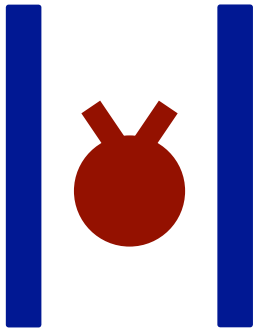
Case 3
Wall Ahead
LR = 11



Case 4
Open Space
LR = 00

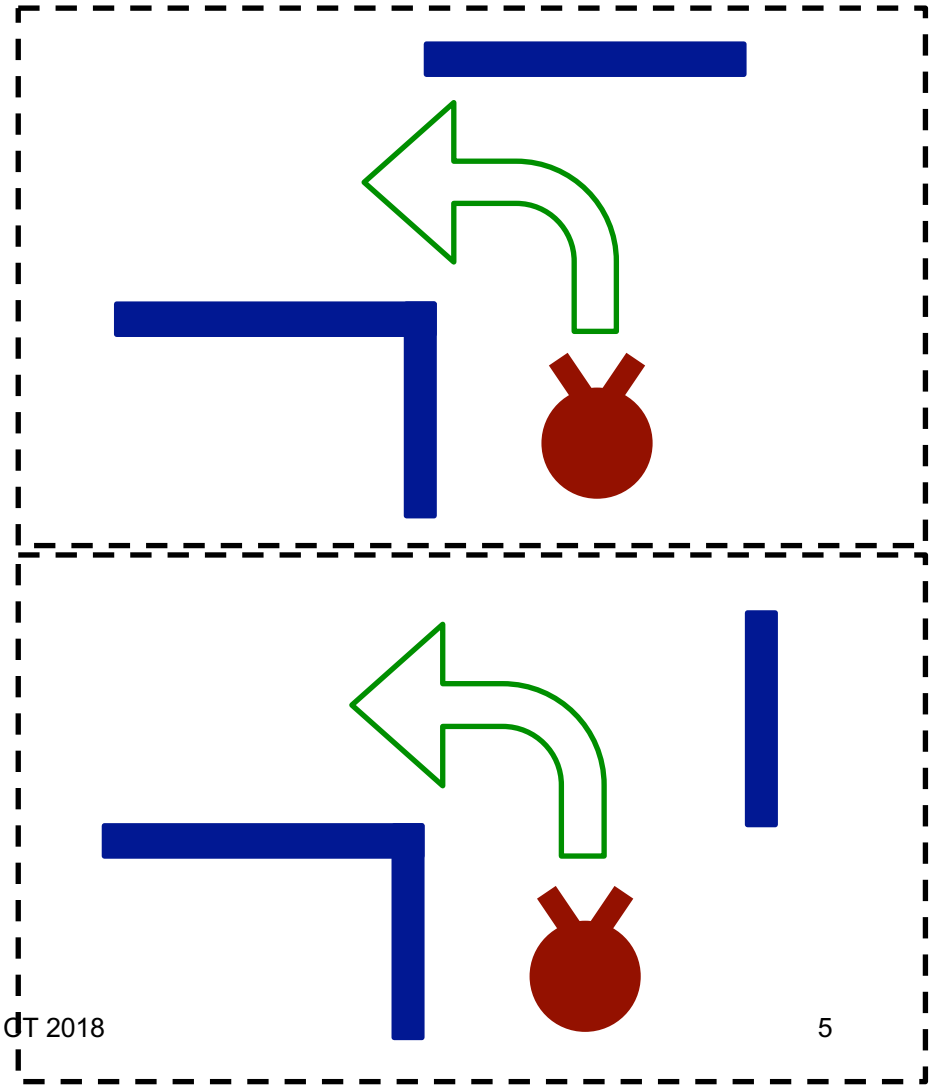
What Will Not Happen

Wall at Both Sides



There is no narrow corridor.

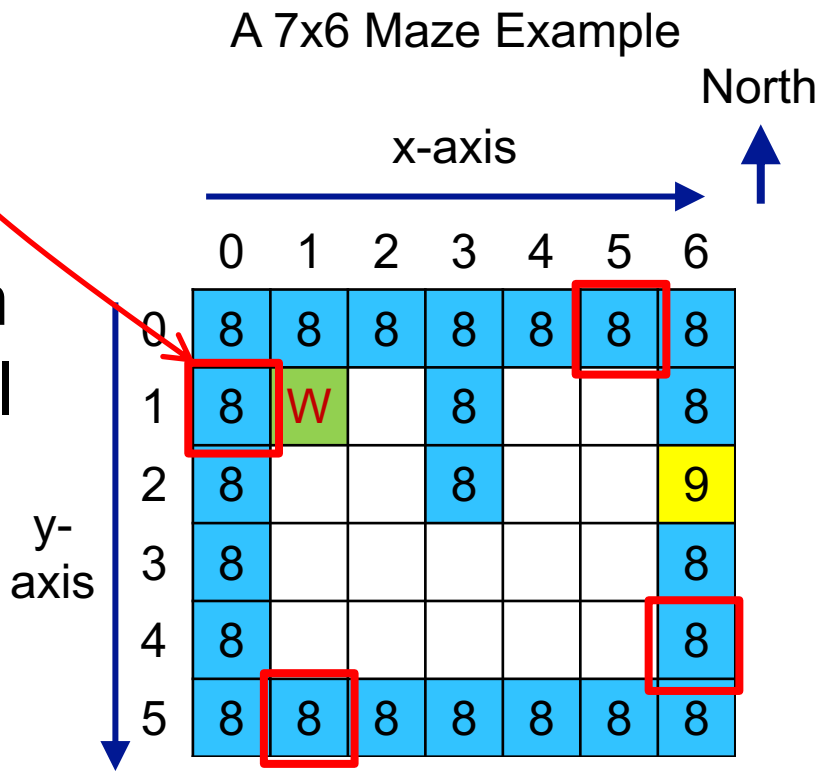
No Narrow Corner



Maze Universe

The exit will not appear at the corners (e.g., at (0,1), (5,0), (6,4), (1,5) in this case). Or it forms a narrow corner to prevent you from going out.

- Encoding
  - 8: Wall
  - 0: Empty space
  - 9: Exit
- You can be at any position initially, except upon a wall
- You can face to four different directions: N (north), E (east), S (south), W (west)



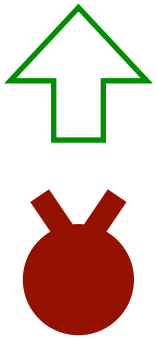
What Actions You Can Take

- You have four actions to take (2-bit `move`)
  - Standing still (halt)
  - Move forward (for 1-unit distance)
  - Turn left (in place, 90 degree)
  - Turn right (in place, 90 degree)
- One at a clock cycle (synchronous to rising clock edges)

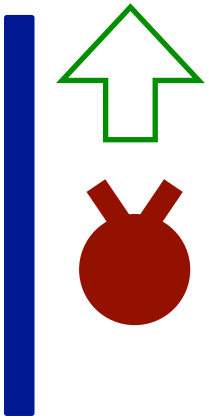
Your Strategies

Move Forward

Open Space in The Beginning

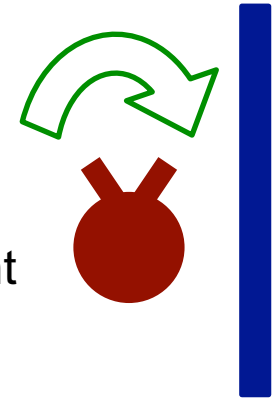


Wall to The Left

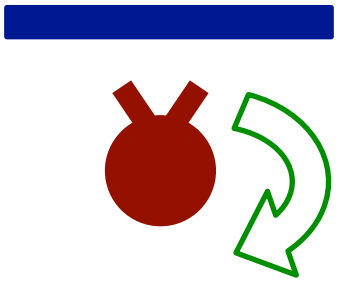


Turn Right

Wall to The Right

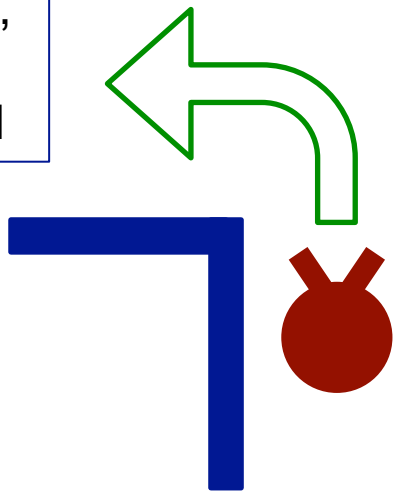


Wall Ahead

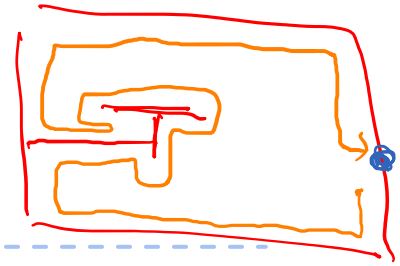


Move Forward, Turn Left, Move Forward

Left Corner

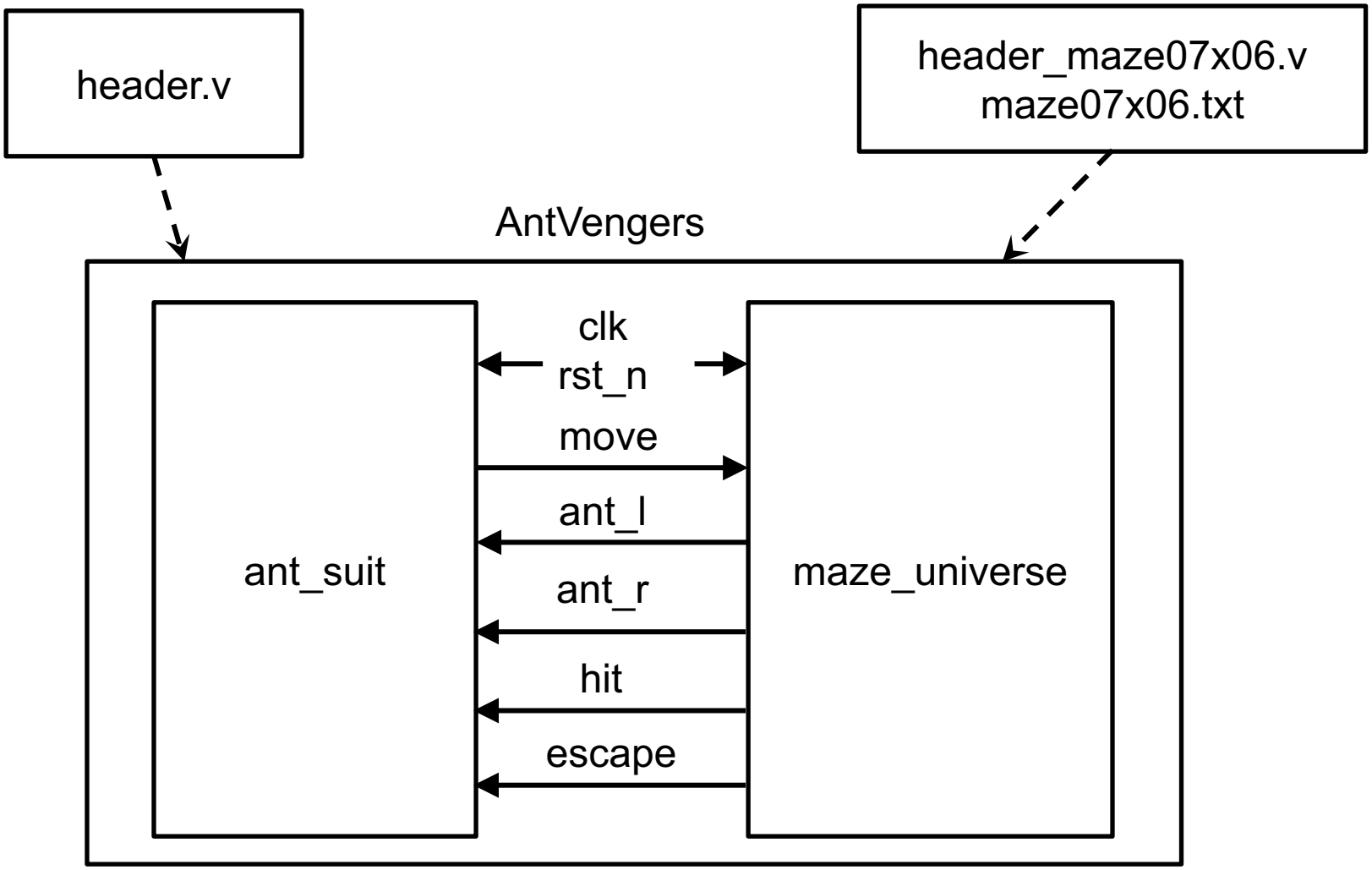


Assumptions



- There is no wall island in the maze
  - All walls are connected to prevent loops
- Keep the wall to your left
- Corridors are always wider enough
  - Wall ahead if both antennas detect something
- You will know when you bump into the wall
  - 1-bit `hit` signal
  - Your position/direction remains the same
- You will know when you escape the Maze
  - 1-bit `escape` signal

Building Blocks and IOs



Requirement

- Design an Ant-Man suit to get through the Maze Universe
  - Detail your design concept
- Create your own maze (>= 15x15)
- Read the source code
  - You may improve the Maze Universe (any bug inside?)
- Have fun!

Challenge?!

- What if there exists narrow corridors and/or narrow corners?
- What if you are not sure if there is any wall island in the maze

Challenge of Narrow Corridors and/or Narrow Corners

- Assume that it is possible to have narrow corridors and/or narrow corners in the maze
- How do you conquer them? Is it possible to solve the challenge with the present I/O signals?

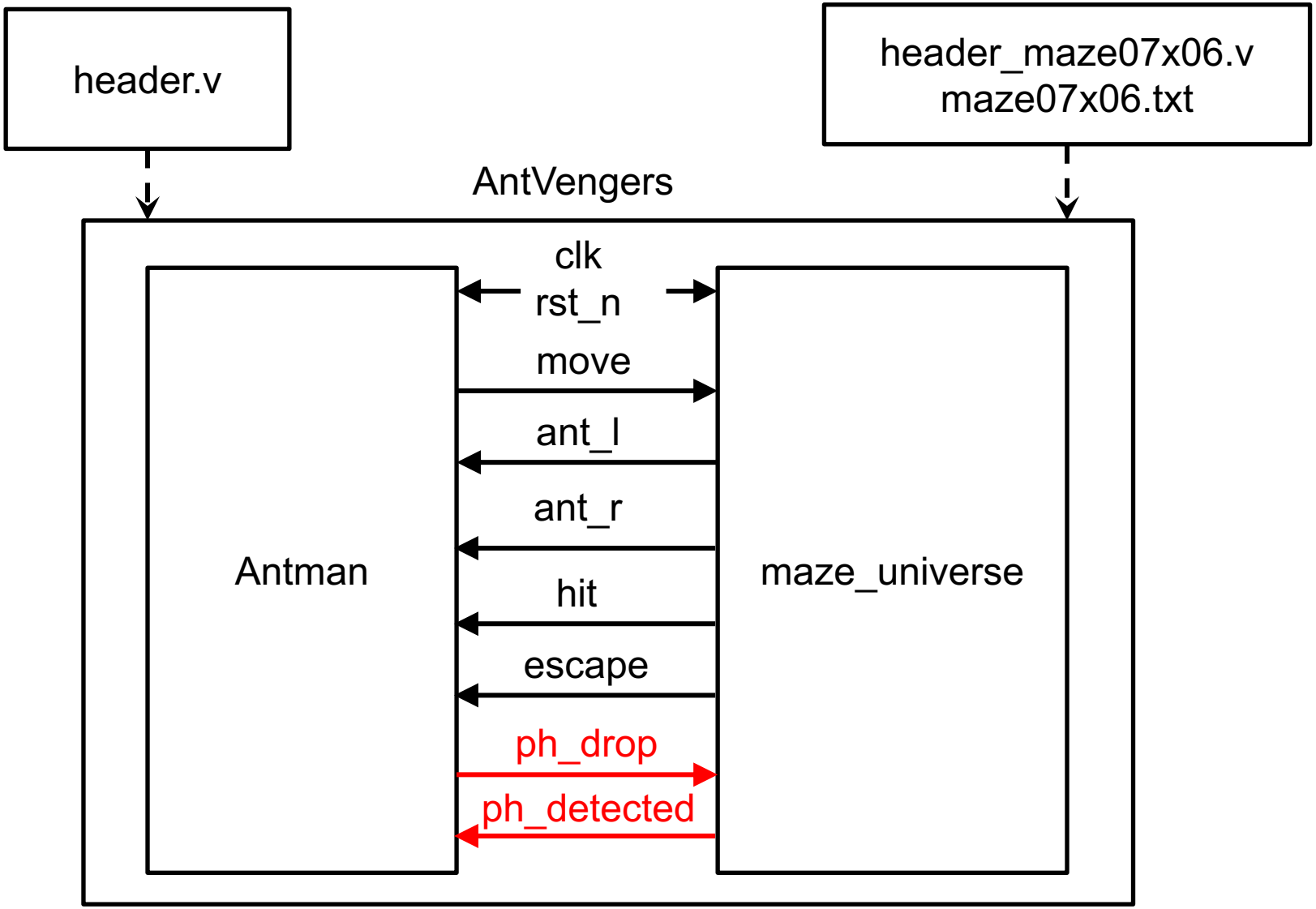
- Note: disable the wall checking by using NOCHECK mode

Challenge of Wall Islands

- Assume that it is possible to have wall island(s) in the maze
- How do you conquer the wall island? Possibly to learn from ant:
  - Deploy (drop) pheromone along the path
  - We implement 2-bit pheromone but only use 2'b00 (no pheromone) and 2'b01 (pheromone detected)
- You can even use different kinds of pheromone (with 2'b10 and 2'b11)
  - Use your imagination to extend the problem;
  - Then solve the problem as best as you can

- Note: turn on CHALLENGE mode

Modified IOs



Completeness of Challenge

- Propose your solution
  - Is it complete?
  - Discuss your assumptions
- Design the upgraded Ant-Man suit
  - You can turn on the challenge mode by including `challenge.v` for Verilog simulation or adding `+define+CHALLENGE` when invoking `ncverilog`
- Design your own maze
  - To test your Ant-Man suit (and beat others', if possible)
- Improve the specification?
  - You may modify and improve the maze universe for a better problem scenario (or solution)

00_README.txt

```
00_README.txt        : This README file.
ant.sh               : Shell script to simulate the  example.
ant_fsdb.sh          : Shell script to simulate the example
                       with fsdb output.
ant_challenge.sh     : Shell script to turn on the challenge
                       mode.
ant_nocheck.sh       : Shell script to disable the wall checking.
header.v             : Header file for AntVengers!
AntVengers.v         : AntVengers! test stimulus.
maze_universe.v      : Maze Universe that reacts to your
                       Ant-Hero suit.
ant_suit.v           : Ant-Man suit that you are going to
                       design and
                     : replace with.
header_maze*.v       : Header file for maze examples.
maze*.txt            : Maze examples.
```

ant.sh

```
#!/bin/sh
ncverilog \
  header.v \
  header_maze07x06.v \
  AntVengers.v maze_universe.v ant_suit.v \
  +debug=1 \
  +access+r
```

You may execute the shell script by
`$ sh ./ant.sh`

maze_universe.v

```
module maze_universe (
...
)
...
  initial begin
    if ($value$plusargs("debug=%d", debug)) begin
      $display(">>> Debug level = %d", debug);
    end else begin
      debug = 0;
    end
    if ($value$plusargs("fsdbfile=%s", fsdbfile)) begin
      if (debug >= 1)
        $display(">>> Dumping the wafeform to [%s]", fsdbfile);
      $fsdbDumpfile(fsdbfile);
      $fsdbDumpvars;
    end
```

You can select the debug mode by
`$ ncverilog +debug=1`

You can assign the file name by
`$ ncverilog +fsdbfile=whatever.fsdb`

ant_fsdb.sh

```
#!/bin/shncverilog \
   header.v \
   header_maze10x11.v \
   AntVengers.v \
   maze_universe.v \
   ant_suit.v \
   +fsdbfile=maze10x11.fsdb \
   +debug=1 \
   +access+r
```

ant_challenge.sh

```
#!/bin/shncverilog \
   header.v \
   header_maze07x06.v \
   AntVengers.v maze_universe.v ant_suit.v \
   +debug=2 \
   +define+CHALLENGE \
   +access+r
```

challenge.v

```
// turn on the challenge mode
`define CHALLENGE 1
```

maze_universe.v

```
module maze_universe (
  input wire clk,
  input wire rst_n,
  input wire [1:0] move,
// challenge mode
`ifdef CHALLENGE
  input wire [`PH_WIDTH - 1:0] ph_drop,
  output wire [`PH_WIDTH - 1:0] ph_detected,
`endif
  output reg ant_r = 0,
  output reg ant_l = 0,
  output reg hit = 0,
  output reg escape = 0
);
```

header_maze07x06.v

```
`define MAZE_WIDTH 7
`define MAZE_HEIGHT 6
`define INIT_X 1
`define INIT_Y 1
`define INIT_DIR `WEST
`define DEFAULT_MAZE "maze07x06.txt"
```

maze07x06.txt

```
8888888
8008008
8008009
8000008
8000008
8888888
```

header.v

```
`timescale 1ns/100ps
`define POS_WIDTH 8
`define MAZE_ELE_WIDTH 5
`define DIR_WIDTH 4
`define CYC     10
`define DELAY   1
`define ABORT   500
`define STRING  32
`define NORTH   4'b1000
`define EAST    4'b0100
`define SOUTH   4'b0010
`define WEST    4'b0001
```

```
`define HALT    2'b00
`define RIGHT   2'b01
`define LEFT    2'b10
`define FORWARD 2'b11
`define WALL    4'd8
`define EXIT    4'd9

`define PH_WIDTH  2
```

maze_universe.v

```
maze_description = `DEFAULT_MAZE;
fd = $fopen(maze_description, "r");
for (j = 0; j < `MAZE_HEIGHT; j = j + 1) begin
  for (i = 0; i < `MAZE_WIDTH; i = i + 1) begin
    status = $fscanf(fd, "%1d", maze[i][j]);
    if (maze[i][j] == `EXIT) begin
      exit_x = i;
      exit_y = j;
    end
  end
end
if (debug >= 1) begin
  display_maze_initial;
  display_maze;
end
if (debug == 3) display_maze_elements;
$fclose(fd);
```

ant_suit.v

```
module Antman (
  input wire clk,
  input wire rst_n,
  input wire ant_r,
  input wire ant_l,
  input wire hit,
  input wire escape,
// challenge mode
`ifdef CHALLENGE
  output reg [`PH_WIDTH - 1:0] ph_drop,
  input wire [`PH_WIDTH - 1:0] ph_detected,
`endif
  output reg [1:0] move
);
  // parameters: action
  parameter [1:0] halt       = `HALT;
  parameter [1:0] turn_right = `RIGHT;
  parameter [1:0] turn_left  = `LEFT;
  parameter [1:0] forward    = `FORWARD;
  parameter cyc = `CYC;
  parameter delay = `DELAY;
  initial begin
    #cyc;
    #cyc;
    @(posedge rst_n);
    standing_still;
    standing_still;
    moving_forward;
    turning_right;
    ...
  end
// challenge mode: deploy pheromone
`ifdef CHALLENGE
  always @* begin
    if (ph_detected == 0)
      ph_drop = 2'b1;
    else
      ph_drop = 0;
  end
`endif
  task moving_forward;
    begin
      @(posedge clk) move = forward;
    end
  endtask
  task turning_left;
    begin
      @(posedge clk) move = turn_left;
    end
  endtask
  ...
endmodule
```