Exercise 14 (Navigate Unit):

The ***navigate*** unit is poorly named, but I already committed to the name so am not going to change it. It does not really navigate the maze but rather performs high level functions of either changing heading or performing a move. It receives its directives from ***cmd_proc*** unit *(discussed later).*

During a change in heading this unit simply asserts **moving** and lets the ***PID*** do its job. When the ***PID*** unit declares we are **at_hdng** it returns to IDLE.

During a move this unit is mainly responsible for controlling forward speed. This involves:

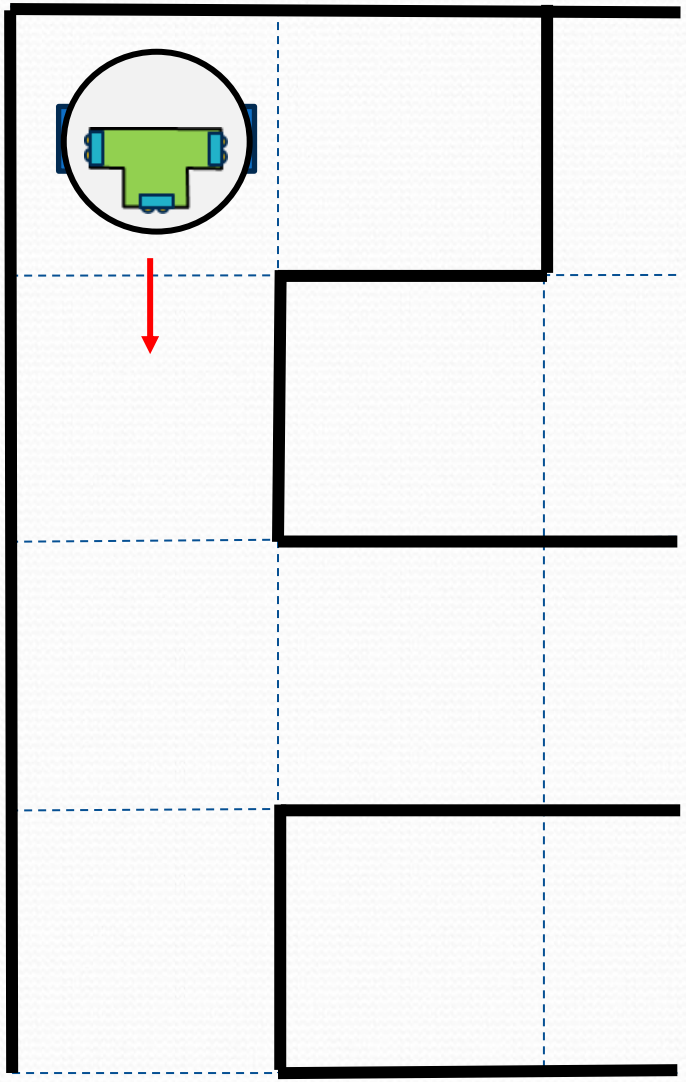
- Ramping the motors up to MAX_FRWRD *(acceleration)*
- Looking for the terminating condition (either obstacle in front or a lft/right opening in the maze)
- Ramping down the motor speed *(deceleration).* Deceleration can be normal, or fast depending on the terminating condition.

Exercise 14 (Navigate Unit):

Imagine the MazeRunner is located in the position shown heading south and is being instructed to stop at the first left opening. It would stop immediately, because the left IR sensor would see the left is already open.

For this reason, when the MazeRunner is to stop at a left or right opening it must look for a rise in the signal. In this case it must look for a rise of **lft_opn** as its terminating condition.

You must code rising edge detectors on both **lft_opn** and **rght_opn** inside ***navigate***. So you should have internal signals called **lft_opn_rise** & **rght_opn_rise**



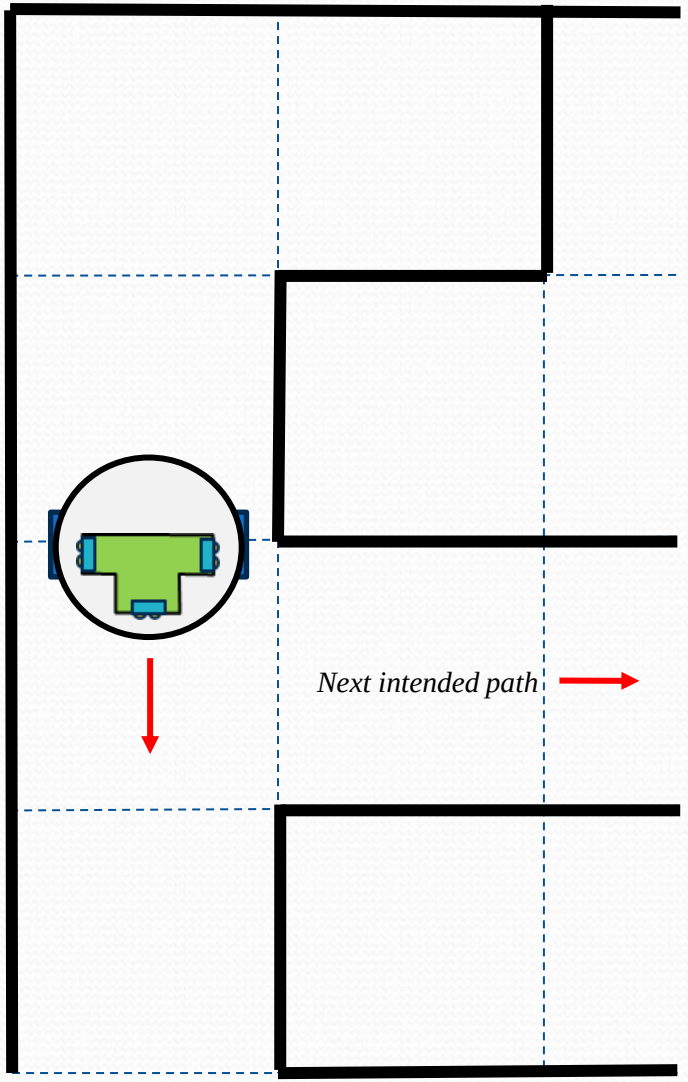
Exercise 14 (Navigate Unit):

When the MazeRunner gets to a left opening and it sees **lft_opn_rise** it will enter the deceleration state.

The image shown depicts the moment when **lft_opn_rise** would occur. From this point the MazeRunner should stop within ½ a square of travel so it is centered in the square and can make a turn to the east.

This is the condition considered "normal" deceleration and the **frwrd_spd** register is decremented at 2x the acceleration rate *(**dec_frwrd** signal asserted)*.

Contrast that to the scenario shown in the next slide where deceleration occurs at the "fast" rate or 8x the acceleration rate.



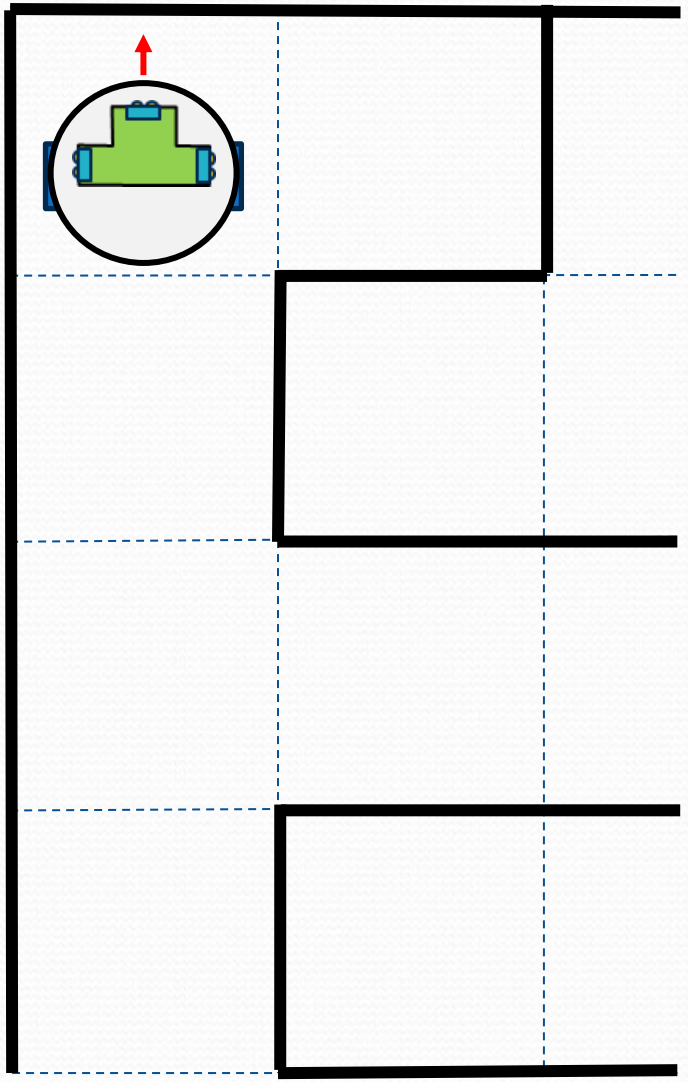
Exercise 14 (Navigate Unit)
(need to hit the brakes fast):

This scenario could depict the MazeRunner commanded to stop at a left opening, but a left opening was not found.

Obviously if the forward looking IR sensor determines the path is no longer open (~**frwrd_opn**) then the MazeRunner needs to stop.

Given the distance at which the center IR can make the call, the MazeRunner has very little distance in which to stop. It needs to hit the brakes hard.

When ~**frwrd_opn** is the cause for entering deceleration state we need to enter a state where we are decelerating hard (8x normal). The SM should be asserting **dec_frwrd_fast**.



Exercise 14 (Navigate Unit) (supporting circuits):

A shell (**navigate_shell.sv**) is provided that has the implementation of **frwrd_reg** completed. You primary job is to complete the SM, but there are a few ancillary things as well.

- Rising edge detectors on **lft_opn**/**right_opn**

- Rate of inc/dec of **frwrd_spd** is controlled by magnitude of **frwrd_inc**. **frwrd_inc** should be controlled by a parameter **FAST_SIM** and should be 6'h18 if **FAST_SIM** and 6'h02 otherwise

- You need to decode the current magnitude of **frwrd_spd** to generate **en_fusion**. **en_fusion** should be asserted if **frwrd_spd** is greater than ½ **MAX_FRWRD**. This allows the IR sensors to only affect navigation if we know the MazeRunner is moving forward.

The functionality of the SM is discussed next.

Exercise 14 (Navigate Unit) (SM functionality):

Navigate SM functionality:

- Sit in IDLE till told to turn to a new heading (**strt_hdng** asserted) or start a move (**strt_mv** asserted).
- If performing a heading change simply assert **moving** and wait until **at_hdng** signal *(from **PID** unit)* is asserted. Assert **mv_cmplt** on way back to IDLE.
- If performing a move first initialize forward speed (**init_frwrd**) and then ramp up to speed. In this state assert **moving** and **inc_frwrd**.
    - Stay in the accelerate state because the **frwrd_spd** register will saturate to **MAX_FRWRD** on its own. So effectively this is both accelerate and "cruise" state.
    - Leave the accelerate state when IR readings indicate stopping condition *(either obstacle in front (~frwrd_opn) or desired opening lft/right) (more on next slide)*
- There are two decelerate states, they differ only in whether one is asserting **dec_frwrd** or **dec_frwrd_fast**. The SM stays in these state until the **frwrd_spd** register is zero. Even though we are in process of stopping in these states, **moving** should still be asserted. These states return to IDLE when completed *(and assert **mv_cmplt** on the way)*

SM output signals:
moving
init_frwrd
inc_frwrd
dec_frwrd
dec_frwrd_fast
mv_cmplt

Exercise 14 (Navigate Unit) (move termination):

A heading change move is complete when **at_hdng** is asserted.

A forward movement command is complete when:

- There is a wall in front of the MazeRunner (~frwrd_opn). Need to brake fast in this scenario
- There is an opening to the left (**lft_opn_rise**) and the signal **stp_lft** is asserted.
- There is an opening to the right (**rght_opn_rise**) and the signal **stp_rght** is asserted.

stp_lft & **stp_rght** originate either from ***cmd_proc*** or from the maze solver. They are signals that instruct ***navigate*** to stop the MazeRunner at either or left or right opening.

At termination of either a heading change or a forward move the signal **mv_cmplt** should be asserted to let either ***cmd_proc*** or the maze solver know that the move has been completed.

Exercise 14 (Navigate Unit) (what you need to do):

- Study the provided **frwrd_spd** register implementation in **navigate_shell.sv**
- Rename **navigate_shell.sv** to **navigate.sv** and flesh it out.
    - Add edge detectors on **lft_opn/right_opn**
    - Implement **frwrd_inc** based on parameter FAST_SIM
    - Implement **en_fusion** signal
    - Implement the SM
        - Study requirements
        - Declare states and SM outputs
        - Implement state registers
        - Implement state transition and output logic
- A testbench **navigate_tb.sv** has been started. Look it over and **flesh it out to be more complete**.
- Submit:
    - **navigate.sv**
    - Completed **navigate_tb.sv**
    - Proof it passed the self-checking testbench (*transcript window*)