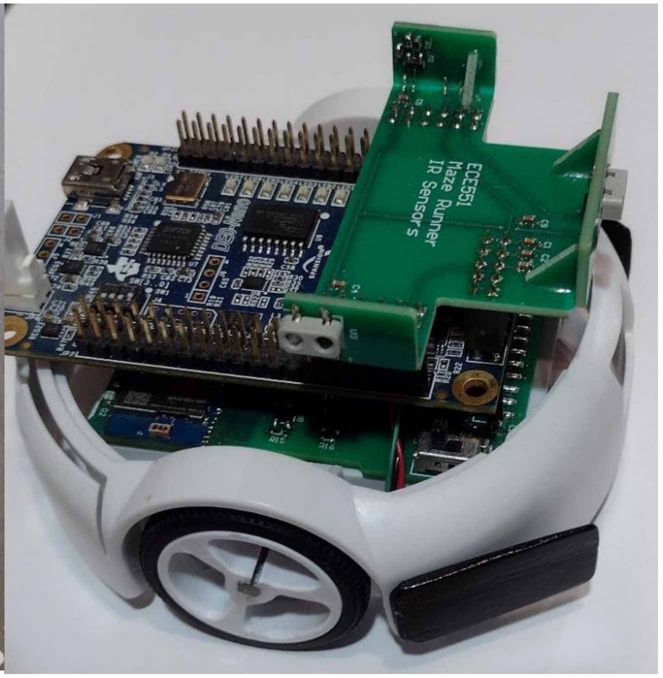
ECE 551

Project Spec

Fall ‘23

MazeRunner



Grading Criteria: (Project is 28% of final grade)

- Project Grading Criteria:
  - Quantitative Element 12.5%

    (yes this could result in extra credit)

$$Quantitative = \left(\frac{EricTommys\_ProjectArea}{YourSynthesizedArea}\right)$$

Note: The design has to be functionally correct for this to apply

  - Project Demo (87.5%)
    - ✓ Code Review (12.5%)
    - ✓ Testbench Method/Completeness (15%)
    - ✓ Synthesis Script review (7.5%)
    - ✓ Post-synthesis Test run results (10%)
    - ✓ Results when placed in EricHarish Testbench (22.5%)
    - ✓ Run of MazeRunner in maze (12.5%)
    - ✓ Temmates judgement of your contribution (7.5%)

Extra Credit Opportunity:

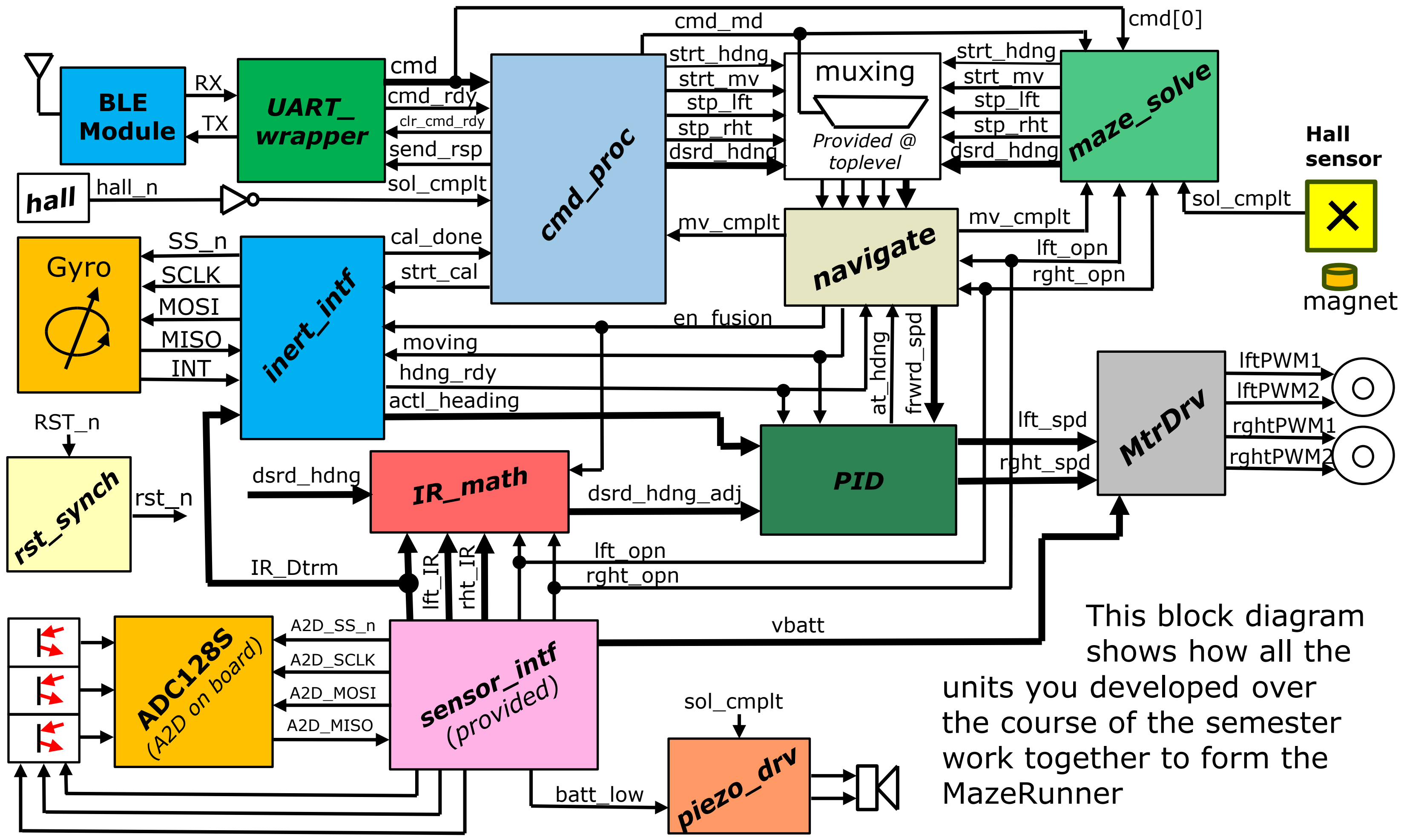
Appendix C of ModelSim tutorial instructs you how to run code coverage (NOTE: this only works on vsim on the linux machines)

- Run code coverage on a single test and get 1% extra credit
- Run code coverage across your test suite and get a cumulative number and get 2% extra credit.
- Run code coverage across your test suite and give **concrete** example of how you used the results to improve your test suite and get 3% extra credit.

Project Due Date

- Project Demo Involves:
  - ✓ Code Review
  - ✓ Testbench Method/Completeness
  - ✓ Synthesis Script & Results review
  - ✓ Post-synthesis Test run results
  - ✓ Results when placed in EricHarish testbench
  - ✓ Results when mapped to FPGA and run in maze

Block Diagram of MazeRunner

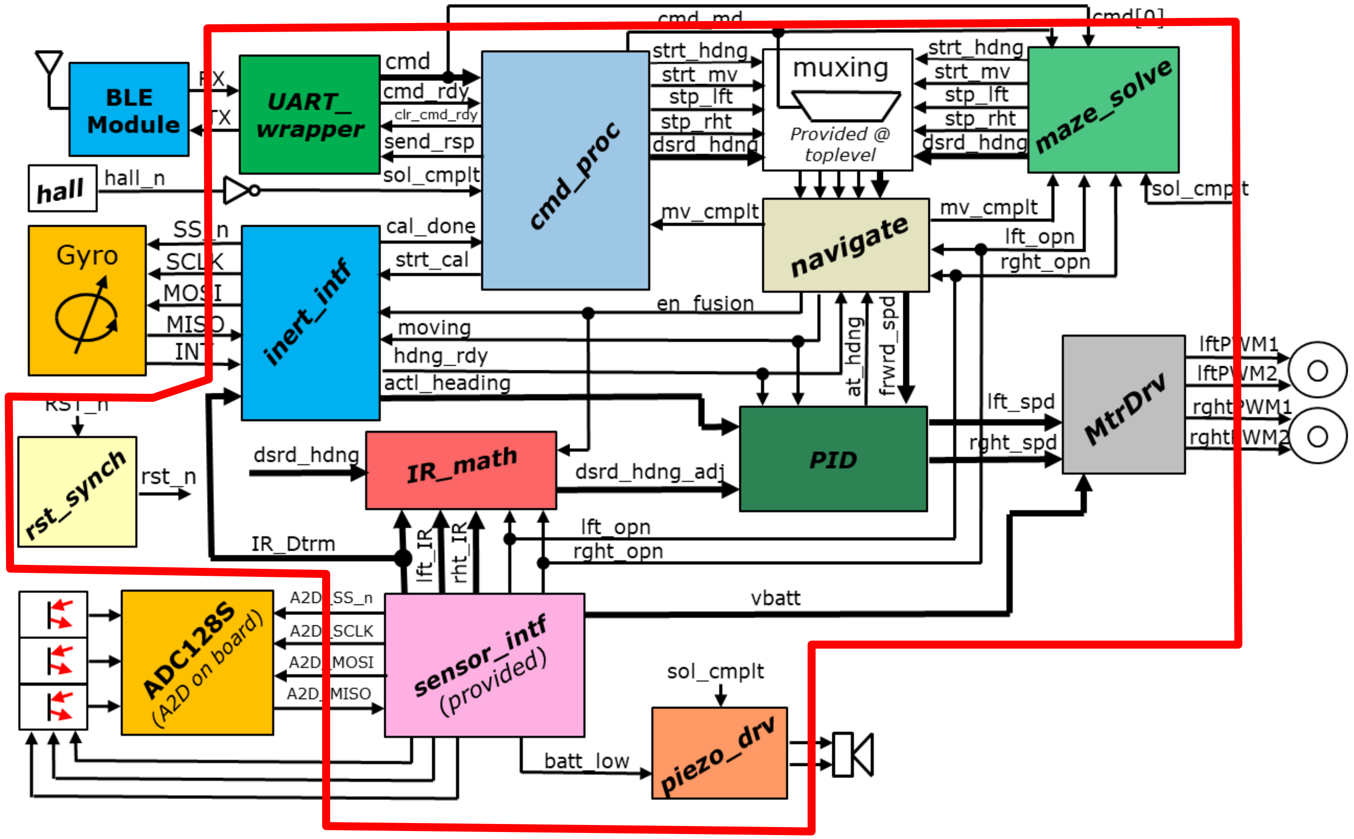


This block diagram shows how all the units you developed over the course of the semester work together to form the MazeRunner

Toplevel Digital of our Design

The blocks outlined in red are pure digital blocks, and will be coded with the intent of being synthesized via Synopsys to our standard cell library. For practical purposes we will also map that logic to a DE0 FPGA board so we can run the demos.

You Must have a block called **MazeRunner.sv** which is top level of what will be the synthesized DUT. A shell is provided to ensure we all have the same interface.



MazeRunner Interface

Signal Name:	Dir:	Description:
clk	in	Clock input (50MHz)
RST_n	in	Raw active low input from push button. Synched to form **rst_n** which resets all units.
INRT_MISO	in	Monarch In Serf Out from SPI bus of MEMs gyro
INRT_SS_n	out	Active low serf select to gyro
INRT_SCLK	out	SPI bus clock (to gyro)
INRT_MOSI	out	Monarch Out Serf In (to gyro)
INRT_INT	in	New set of reading ready from gyro
RX, TX	in	UART data in from Bluetooth module (cmd), and pos ack response (0xA5)
IR_lft_en, IR_cntr_en, IR_right_en	out	Enables emitter on the 3 IR sensors
hall_n	in	Input from hall effect sensor. Indicates magnet found
piezo, piezo_n	out	Drives to piezo buzzer for charge fanfare and low battery tone
lftPWM1,lftPWM2, rghtPWM1,rghtPWM2	out	PWM signals to control motor speed and direction
A2D_MISO	in	Monarch In Serf Out from SPI bus of MEMs gyro
A2D_SS_n	out	Active low serf select to gyro
A2D_SCLK	out	SPI bus clock (to gyro)
A2D_MOSI	out	Monarch Out Serf In (to gyro)

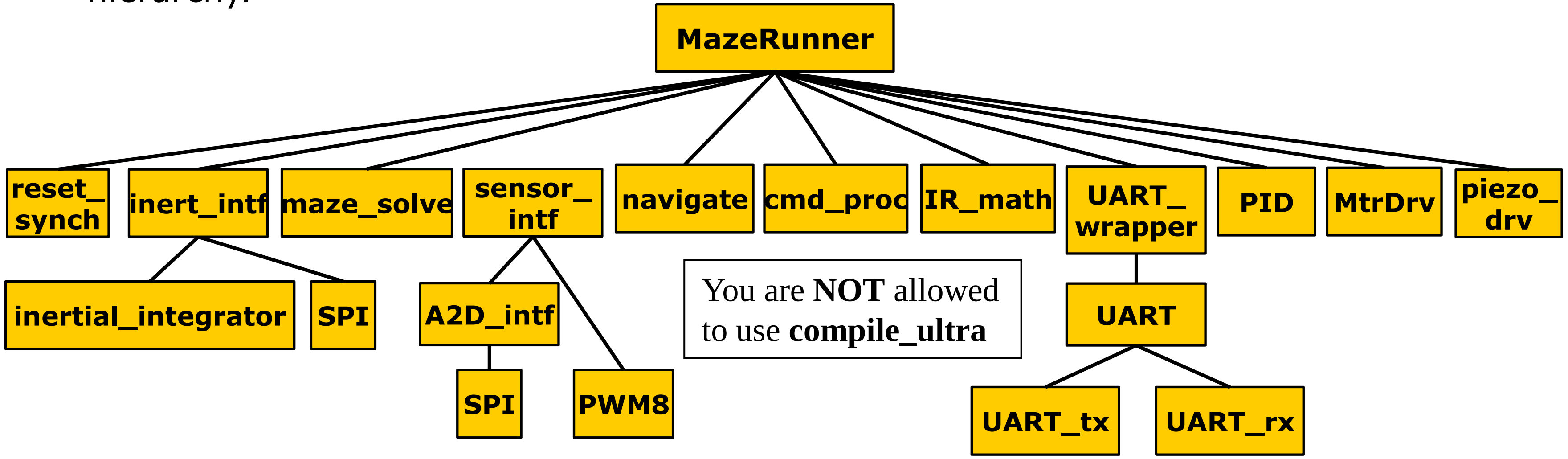
Provided Modules & Files: (available on website under: Project)

File Name:	Description:
MazeRunner_tb.sv	**Optional** testbench template file.
RunnerPhysics.sv	Very helpful model of the physics of the MazeRunner and the maze itself. Use this in conjunction with your testbench to validate behavior.
MazeRunner.sv	**Requried** to use as toplevel. Connects blocks and ensures we all have same interface signal names.
SPI_iNEMO4.sv	Final model of NEMO inertial sensor (gyro) intended for use along with RunnerPhysics.
inertial_integrator.sv	Sits inside ***inertial_interface***. Calculates actual heading.
sensor_intf.sv	Mainly an interface to the IR sensors.
DutyScaleROM.sv	Used in scaling PWM duty cycle for battery sag.
A2D_intf.sv & PWM8.sv	Interface to A2D converter on board. Uses your SPI_mnrch. Children of **sensor_intf.sv**
ADC128S_FC.sv & SPI_ADC128S.sv	Model of A2D converter. Child of **RunnerPhysics.sv**

These files exist in: **MazeRunnerProvided.zip**

Synthesis:

- You have to be able to synthesize your design at the **MazeRunner** level of hierarchy.



You are **NOT** allowed to use **compile_ultra**

- Your synthesis script should write out a gate level netlist of follower (**MazeRunner.vg**).
- You should be able to demonstrate at least one of your tests running on this post synthesis netlist successfully.
- Timing (400MHz) (2.5ns) is mildly challenging. Your main objective is to minimize area.

Synthesis Constraints:

Contraint:	Value:
Clock frequency	363MHz (yes, I know the project spec speaks of 50MHz, but that is for the FPGA mapped version. The standard cell mapped version needs to hit 400MHz. 2.75ns period
Input delay	0.6ns after clock rise for all inputs
Output delay	0.5ns prior to next clock rise for all outputs
Drive strength of inputs	Equivalent to a size 2, 2-input NAND gate from our library
Output load	0.1pF on all outputs
Wireload model	16000
Max transition time	0.125ns
Clock uncertainty	0.125ns

NOTE: Area should be taken after all hierarchy in the design has been smashed.

Eric's Area = 16591

Project Demos (Held in “The Dungeon” B555)

- Project Demos will be held in B555 over the course of 4 days.
  - Sat Dec 9th. 2.5% Extra credit for demoing very early
  - Mon Dec 11th 1.5% Extra credit for demoing early
  - Tues Dec 12th 0.75% Extra credit for demoing bit early
  - Weds Dec 13th Normal due time
  - Thurs Dec 14th 1% Penalty for demoing late

  - Project Demo (87.5%) (12.5% is based on your synthesized area)
    - ✓ Code Review (12.5%)
    - ✓ Testbench Method/Completeness (15%)
    - ✓ Synthesis Script review (7.5%)
    - ✓ Post-synthesis Test run results (10%)
      - ❑ You need to demonstrate post synthesis simulation of a short test at the “MazeRunner” level of hierarchy. Should perform calibration & heading change
    - ✓ Results when placed in EricHarish Testbench (22.5%)
    - ✓ Run of MazeRunner in the maze (12.5%)
    - ✓ Temmates judgement of your contribution (7.5%)