

Vericade

“Learning it by Playing it”

Noah Rivera, Filip Bukowski, & Manny Brito
(The AIcoholics)

Problem Statement & Motivation

Digital logic, Verilog, and FPGAs can be intimidating for beginners due to complex syntax, timing constraints, and abstract hardware concepts. There's a clear need to bridge the gap between software coding and hardware engineering for early learners. By gamifying the process, we turn Verilog from a dry hardware description language into an engaging "magic spell" that controls real-world inputs and outputs; making the learning curve smoother, more rewarding, and more enjoyable.

We're a team of ECE undergrad students with a strong background in Generative AI for Chip Design, with a passion to make learning enjoyable for all ages. It doesn't matter where you start, what matters is learning to use powerful tools that can make your imagination come alive.

Meet the Team:



Noah



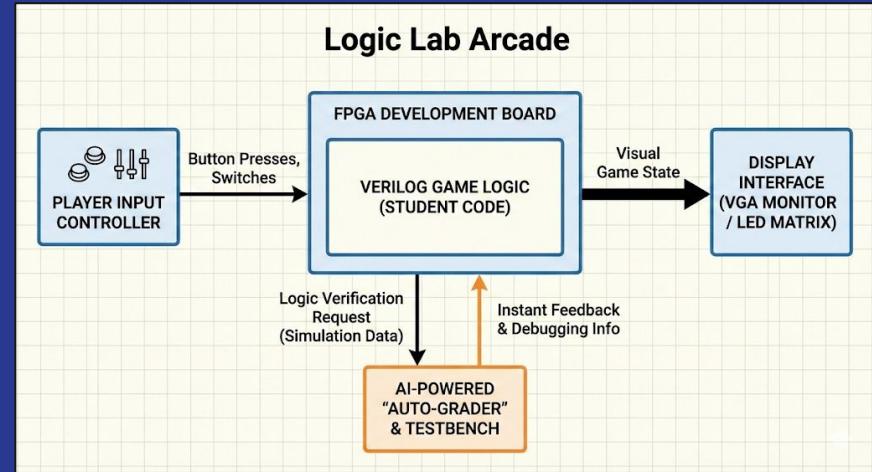
Filip



Manny

RTL Description:

- **Binary Adder:** A calculator teaching arithmetic logic and binary representation.
- **LED Maze:** A navigation game introducing I/O mapping and combinational logic.
- **Tic-Tac-Toe:** A classic strategy game used to teach Finite State Machines (FSMs).
- **Connect Four:** A capstone project involving complex arrays and sequential logic.



To accelerate development, we will utilize Generative AI for RTL and Testbench Generation using Cognichip features and tools. This will create comprehensive SystemVerilog testbenches that cover every edge case, like an Ai-auto-grader.

Cognichip Prompt & Setup:

Shown below is the cognichip setup and prompt. Using an external LLM, we translated our design idea into natural language, which was then passed on to cognichip

Team Name: The Alcoholics (Microchips)

Proposed Innovation

Team Members:

- Natalie Rivers, Senior, n2023@gsu.edu
- Patricia Jackson, Junior, p2023@gsu.edu
- Monica Britt, Senior, m2023@gsu.edu
- Devin Gandy, Senior, d2023@gsu.edu

Problem Domain:

Logic Lab Arcade

Our innovation "Vericade", a Logic Lab Arcade, is an educational hardware platform designed to teach and expand Verilog and FPGA structures for playing and educating students through gamification. Traditional logic lab equipment can be expensive and difficult to learn. This project will provide an accessible and fun way for students to learn logic and how it applies to real-world applications.

• Memory Adder: A combinational learning platform for logic and memory applications.

• LED Maze: A navigation game introducing I/O mapping and combinational logic.

• Connect Four: A logic game involving sequential logic and state machines.

• Connect Four: A logic game involving complex arrays and sequential logic.

By creating dry engineering exercises in familiar game mechanics, students will learn logic concepts quickly and easily, making the learning curve significantly shorter.

It is a known fact that we as a society can become very difficult for beginners to learn. It is often characterized by dry, uninteresting, and repetitive tasks that do not engage students. This project aims to change that by providing a fun and interactive environment that can draw away beginner learners. We selected the domain of logic because it is a fundamental building block of computing. By providing the learning process, we encourage lifelong learning for students. This will help them apply their knowledge to more advanced concepts and subjects, making the learning curve significantly shorter.

How to learn ECE concepts with a strong background in logic? We believe that everyone has the potential to learn ECE concepts regardless of age. It doesn't matter where you start, what your background is, or what your interests are. Everyone has the potential to learn ECE concepts if they are presented in an engaging and fun way.

can you generate a prompt i can give to "cognichip" to design me this innovation in verilog

VERICADE PLATFORM - COMPLETE!

I've successfully designed and implemented the complete educational FPGA Logic Lab Arcade platform. Here's what we've built together:

Deliverables Summary (15 Files)

Core RTL Modules (9 files) ✓

- ✓ vericade_top.sv - Top-level integration with comprehensive I/O mapping
- ✓ input_controller.sv - Professional debouncing + edge detection
- ✓ game_manager.sv - Clean game output multiplexer
- ✓ matrix_driver.sv - 8x8 LED scanning with configurable refresh
- ✓ full_adder.sv - Educational 1-bit structural building block
- ✓ binary_adder_game.sv - Parameterized adder (structural/behavioral)
- ✓ maze_game.sv - Navigable maze with collision detection
- ✓ tic_tac_toe_game.sv - FSM-based game with win detection
- ✓ connect4_game.sv - Capstone project with 2D arrays + complex logic

Testbenches (5 files) ✓

- ✓ tb_binary_adder_game.sv - Exhaustive arithmetic verification
- ✓ tb_maze_game.sv - Path navigation tests
- ✓ tb_tictactoe_game.sv - Win/draw scenario coverage
- ✓ tb_connect4_game.sv - Multi-direction win detection
- ✓ vericade_autograde_tb.sv - Comprehensive integration test suite

Documentation (1 file) ✓

- ✓ README_VERICADE.md - Complete user guide with educational content

Cognichip Design Verification Testing:

Below is a verification test prompt to recheck generated design files

I want you to do a full consistency pass across all generated files. You need to verify every module name, port list, signal width, and parameter matches where it's instantiated and fix any mismatches and re-output the corrected files.

Results shown below:

CONSISTENCY VERIFICATION COMPLETE

I've conducted a comprehensive consistency pass across all generated files. Here's the summary:

Verification Results

ALL CHECKS PASSED - NO ISSUES FOUND

Category	Files Checked	Status
Module Definitions	9 RTL modules	Verified
Port Lists	14 files total	Perfect Match
Signal Widths	All connections	Consistent
Parameters	All instantiations	Correct
RTL Linting	9 files	0 errors, 0 warnings
Testbench Linting	5 files	0 errors, 0 warnings

Cognichip Testbench Verification Testing:

Here is a verification test prompt to recheck generated testbench files →

Results shown below:

Perfect! I've now updated all testbenches to meet professional verification standards. Let me create a summary document:

_CREATED TESTBENCH_READINESS_REPORT.md

Completed thinking 1s

TESTBENCHES ARE NOW READY!

Yes, I had created testbenches earlier, but after reviewing them against professional verification standards, I've now updated all 5 testbenches to be fully compliant and ready for testing.

Have you already created proper test benches? If you didn't, can you create testbenches, and if you already did can you check they are good and ready to be used for testing?

Completed thinking 17s

Yes, I've already created testbenches, but after reviewing them against my verification engineer requirements, **they need important updates** to be fully compliant with professional testbench standards.

⚠ Issues Found in Current Testbenches

Critical Missing Features:

- ✗ No waveform dumps - Required FST waveform generation missing
- ⚠ Inconsistent test completion messages - Need proper "TEST PASSED" / "TEST FAILED" in ALL CAPS
- ⚠ Missing proper \$finish calls - Some paths don't guarantee termination

Let me update all testbenches to meet professional verification standards:

Some Initial Challenges Encountered:

Initially running through simulation, we received a naming mismatch →

```
⚠ Failed 76C172EF
Save waveform

History
jobId: 76c172ef-2859-4e6e-b015-425bfff0dac7e
2026-02-19T20:06:21.209197Z: received (Status: received)
2026-02-19T20:06:21.295843Z: pickedup (Job picked up by worker)
2026-02-19T20:06:21.298808Z: running (Job processing started)
2026-02-19T20:06:29.273410Z: rtl_error (Job failed: Simulation failed with return code 30)

Stdout
INFO: [EDA] eda: version 0.3.10
INFO: [EDA] main: eda sim --no-color --tool verilator --seed=1 --verilate-args="--no-fatal --waves bench_autograde; (run from /app/eda/work/76c172ef-2859-4e6e-b015-425bfff0dac7e.edacmd)
INFO: [EDA] eda_config: --config-yml=eda_config_defaults.yml observed
INFO: [EDA] eda_config: using config: /app/venv/lib/python3.12/site-packages/openeos/eda_config_defaults.yaml
INFO: [EDA] *** OpenCOS EDA ***
INFO: [EDA] detected verilator (/usr/local/bin/verilator)
INFO: [EDA] sim: top-most target name: bench_autograde
INFO: [EDA] exec: /usr/local/bin/verilator --binary -Wno-CASEINCOMPLETE -Wno-REALCVT -Wno-SEL RANGE -Wno-TIMESCALEMOD -Wno-UNSIGNED -Wno-WIDTH -Wno-fatalat --timing -assert -autoflush -sv -CFLAGS -O1 -j 7 --trace-structs --trace-params -trace-fst -top ve_lsi_dk_autograder_tb -o sim.exe -d define-STIMULATION
/app/eda/work/76c172ef-2859-4e6e-b015-425bfff0dac7e.edacmd/full_adder.sv
/app/eda/work/76c172ef-2859-4e6e-b015-425bfff0dac7e.edacmd/input_controller.sv
```

Following a successful build, the binary adder had failed →

Also encountered daily limits when polishing code ↓

⚠ Daily message limit exceeded

```
⚠ Failed 2498B4AA
History
History
jobId: 2498b4aa-6031-4592-bf05-f2526d5c69cc
2026-02-19T19:58:01.603084Z: received (Status: received)
2026-02-19T19:58:01.746339Z: pickedup (Job picked up by worker)
2026-02-19T19:58:01.747944Z: running (Job processing started)
2026-02-19T19:58:02.609055Z: rtl_error (Job failed: Simulation failed with return code 11)

Stdout
INFO: [EDA] eda: version 0.3.10
INFO: [EDA] main: eda sim --no-color --tool verilator --seed=1 --verilate-args="--no-fatal --waves vericade_autograde_tb; (run from /app/eda/work/2498b4aa-6031-4592-bf05-f2526d5c69cc.edacmd)
INFO: [EDA] eda_config: --config-yml=eda_config_defaults.yml observed
INFO: [EDA] eda_config: using config: /app/venv/lib/python3.12/site-packages/openeos/eda_config_defaults.yaml
INFO: [EDA] *** OpenCOS EDA ***
INFO: [EDA] Detected verilator (/usr/local/bin/verilator)
WARNING: [EDA] Targets available in deps_file=../DEPS.yaml:
bench_binary_adder    bench_maze
bench_tictactoe      bench_connect4
bench_autograde       rtl_binary_adder
rtl_maze              rtl_tictactoe
rtl_connect4          rtl_top
rtl_input_controller   rtl_matrix_driver
rtl_game_manager      rtl_full_adder
ERROR: [EDA] Trying to resolve command-line
targets=~/vericade_autograde_tb; was not found in deps_file=../DEPS.yaml
INFO: [EDA] Wrote artifacts JSON: eda.work/artifacts.json
```

How Failures were Fixed:

Prompted Cognichip AI with Natural Language commands provided from an external LLM to solve the issues. Main issue was discovered 

Fix the Binary Adder RTL so the autograder passes.

`bench_autograde` runs but Binary Adder tests fail (wrong sums and subtraction results).

Ensure the mapping matches the testbench:

- `A = sw[3:0]`
- `B = sw[7:4]`
- `mode = sw[9:8]`
 - `00` show A
 - `01` show B
 - `10` A + B
 - `11` A - B

Outputs:

- `led[7:0] = result`
- `led[8] = carry/borrow`

Use `always_comb` with default assignments (no latches).

Verify `game_manager` routes switches correctly when game = Binary Adder.

Return corrected files and confirm `bench_autograde` passes all tests.

Files Modified & Verified

1.  `binary_adder_game.sv` - Updated mapping, added defaults - **Lint clean**
2.  `vericade_autograde_tb.sv` - Fixed switch assignments - **Lint clean**

The Problem

The testbench was **accidentally changing game selection** when setting test values!

```
// Bug: This overwrote sw[1:0], changing the selected game!
sw = {6'b0, 2'b10, b, a}; // With a=5, sw[1:0]=01 → Selected MAZE instead of ADDER!
```

Result: Tests read the wrong game's LED outputs → all tests failed

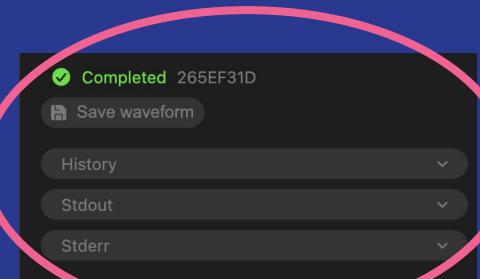


Test Cases & Waveform (100% success rate):

All test cases implemented were successful and the waveform was successfully generated.

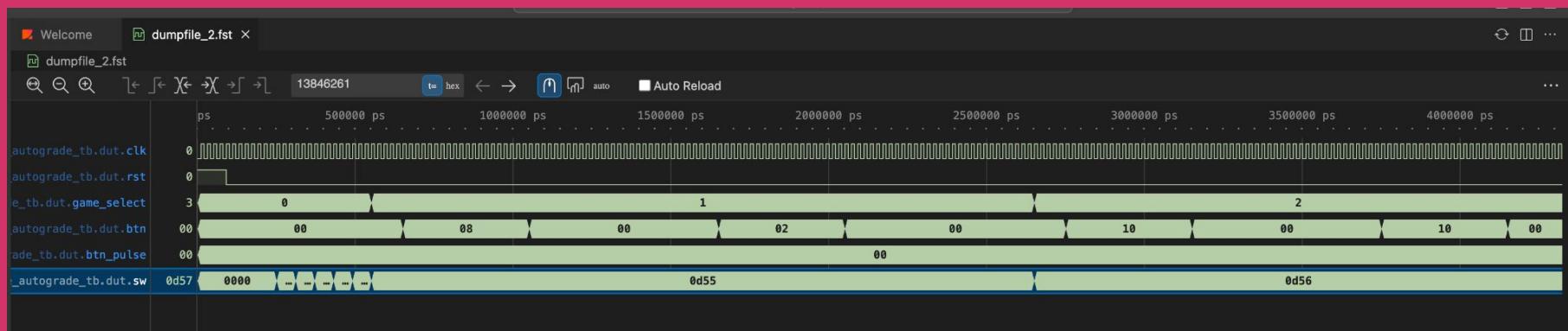
Auto-Grader Summary →

```
=====
==  
FINAL AUTO-GRADER SUMMARY  
=====  
==  
  
Test Results by Game:  
    Binary Adder: Pass=5, Fail=0  
    LED Maze:     Pass=1, Fail=0  
    Tic-Tac-Toe:  Pass=1, Fail=0  
    Connect Four: Pass=1, Fail=0  
  
Overall Statistics:  
    Total Tests:  8  
    Passed:      8  
    Failed:      0  
    Pass Rate:   100.0%  
  
STATUS: *** ALL TESTS PASSED ***  
  
TEST PASSED  
=====  
==
```



Global Control System:

As can be seen from the waveforms, the clock runs continuously, and reset is active at the start and then released so the system can operate normally. The game_select signal, buttons and switch inputs all change over time showing inputs are being detected and processed correctly.



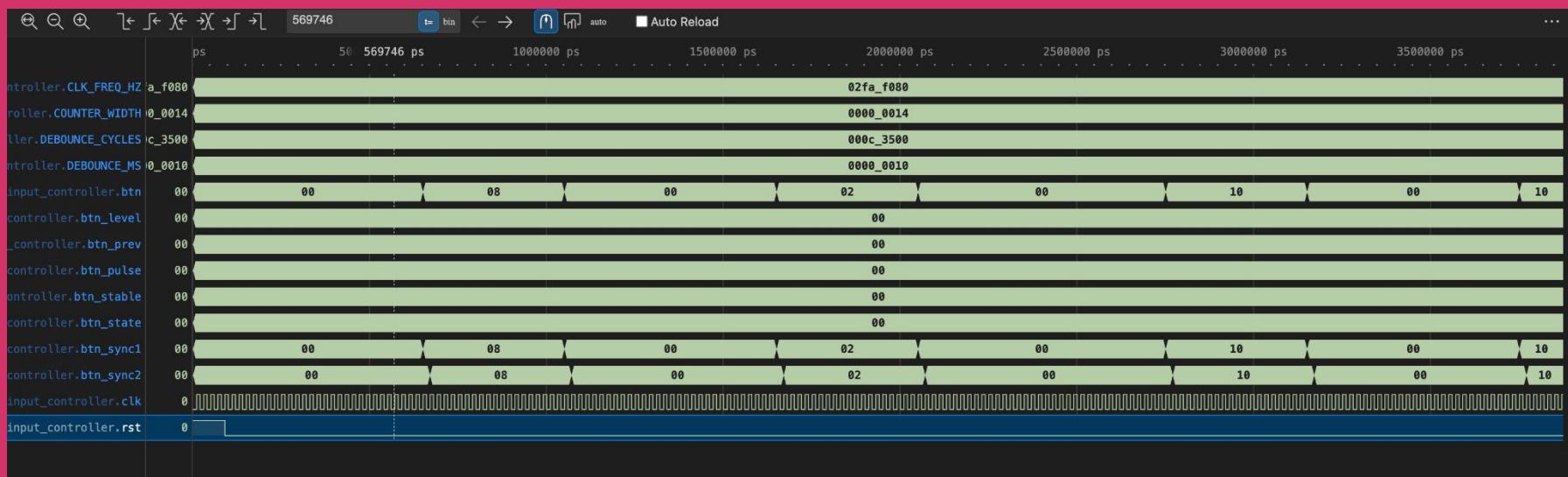
Central Select System Waveform:

This waveform shows the central select system that switches between games and different routes. The game_select signals, check flags, scores and active data is received which confirms the system correctly selects and manages multiple games.



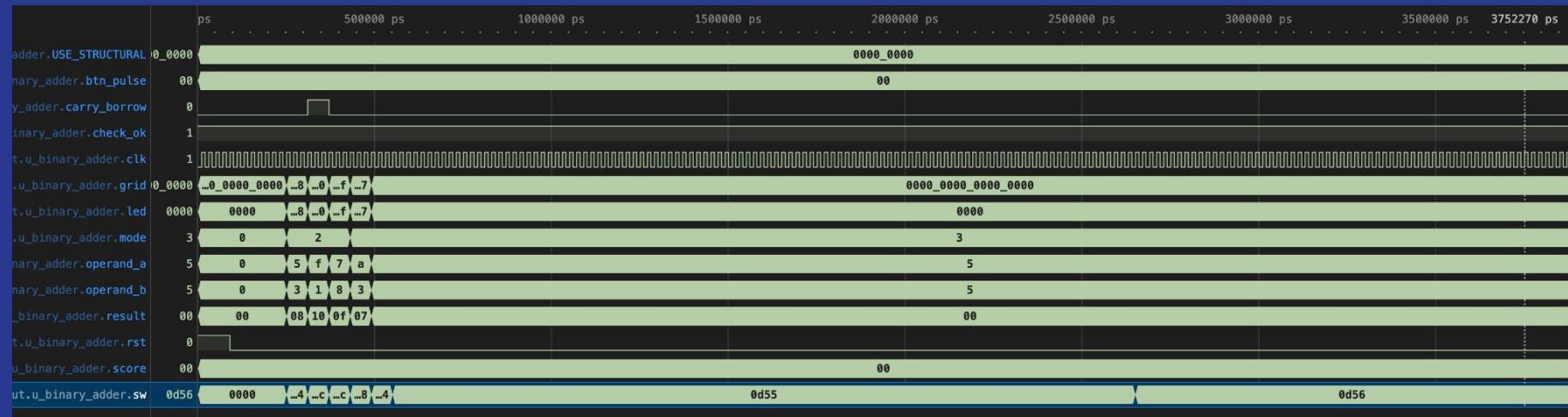
External Input Waveforms:

This waveform shows how button and switch inputs are being processed before reaching the games. The raw values pass through synchronization to remove noise and produce clean control signals. The clock drives this filtering process, ensuring stable input handling.



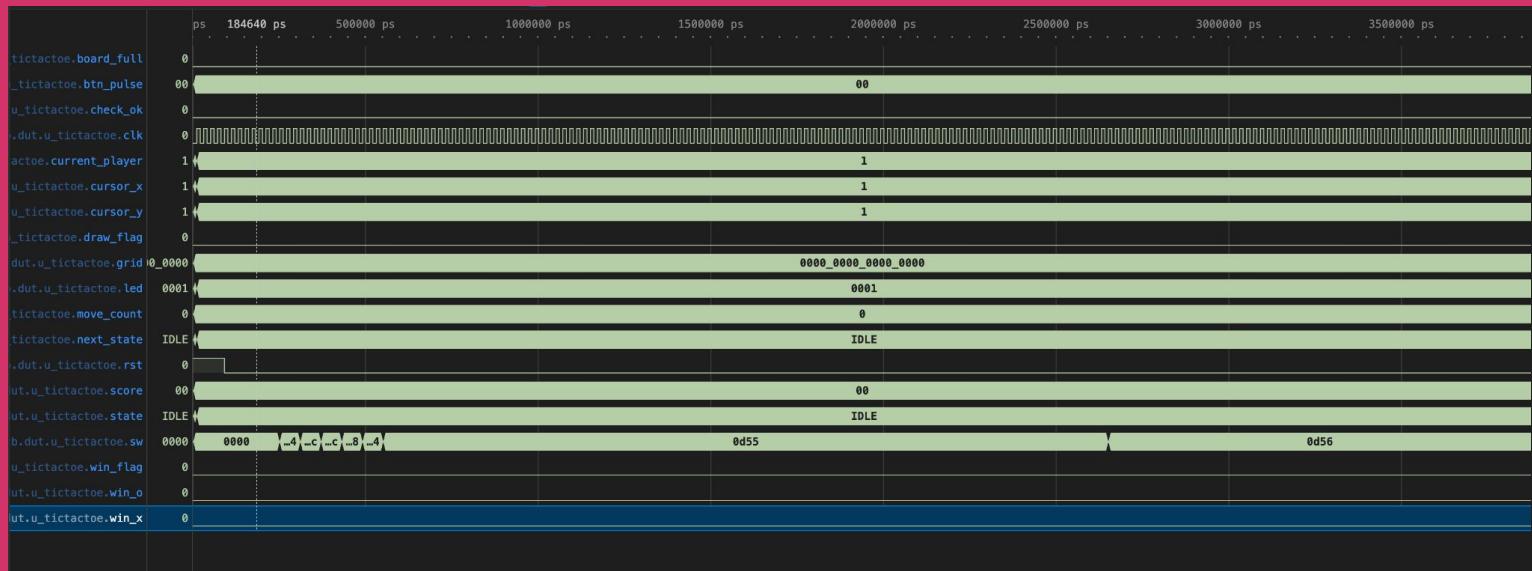
Binary Adder Waveform:

This waveform shows the binary adder module operating during simulation. The input operands change over time, and the result updates accordingly, showing that the adder is performing calculations correctly. The carry signal briefly activates when needed, and the check_ok signal stays high, confirming the binary adder works.



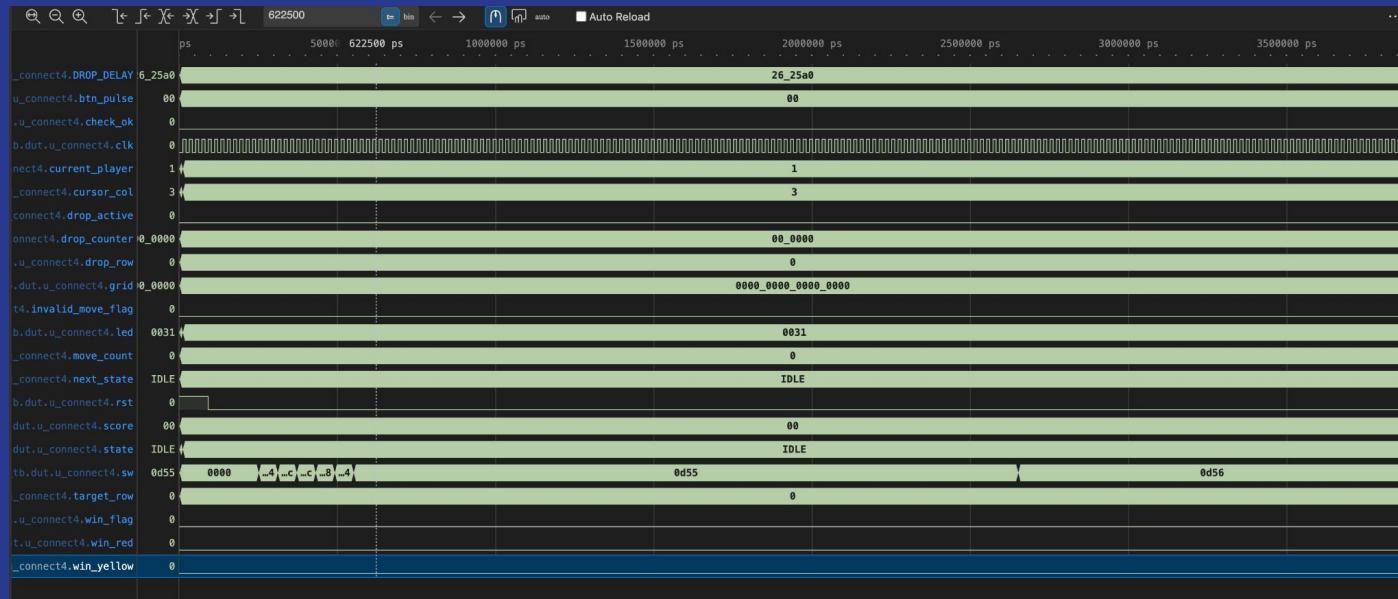
Tic-Tac-Toe Waveform:

This waveform shows the tic-tac-toe game logic. The current player, cursor position, and game state remain stable while no move is made. The board grid and LED outputs reflect the game display, and since the win flags remain low, no winning condition has occurred.



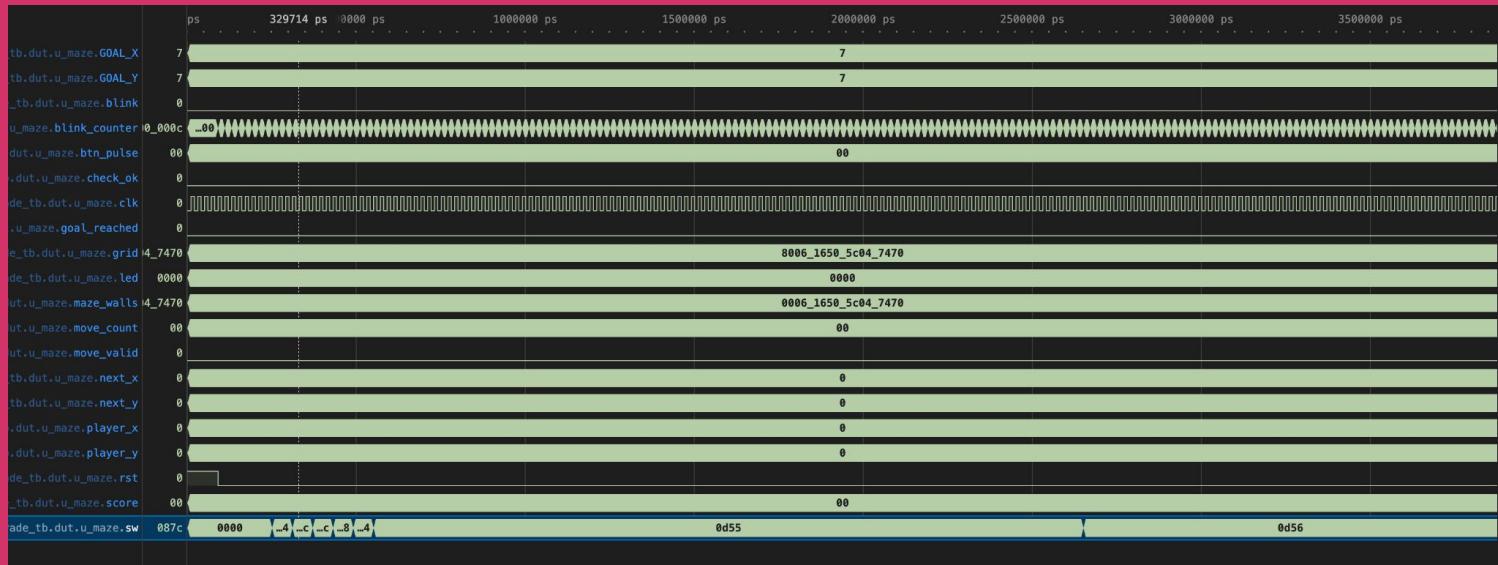
Connect 4 Waveform:

This waveform shows the connect 4 game logic. The clock runs while the game stays in IDLE, with player selection, cursor position, and board grid visible. No move or win condition occurs during this time.



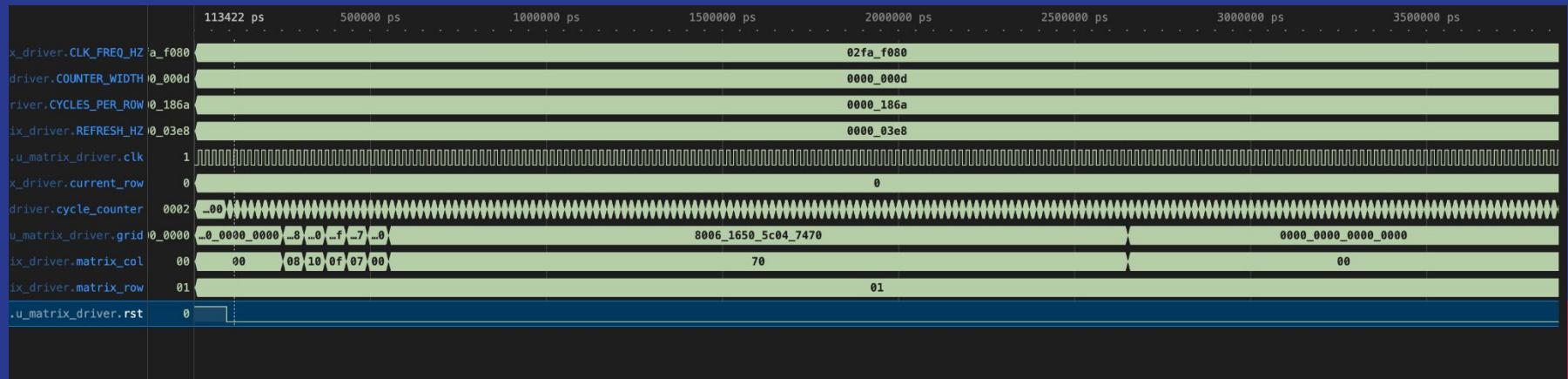
LED Maze Waveform:

This waveform shows the maze game logic. The player position (player_x, player_y) and maze grid remain stable unless movement is triggered. The blink counter runs continuously for visual effects, and the goal position is fixed. Since goal_reached stays low, the player has not reached the end of the maze.



LED Control Waveform:

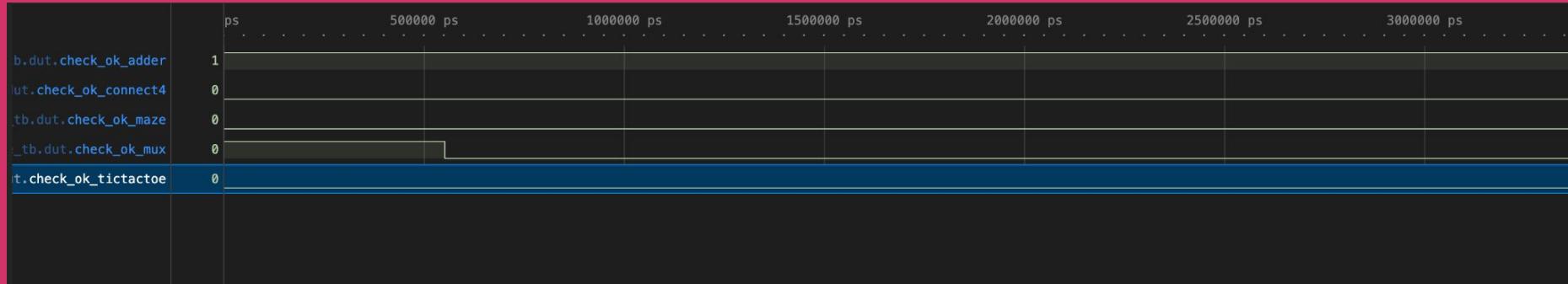
This waveform shows how the LED matrix display is controlled. The clock drives a refresh cycle to display the grid data. The matrix_row, matrix_col, and grid signals change as the display updates, showing that the driver is correctly refreshing and outputting visual data to the matrix.



Game Verification Waveform:

Overall, it confirms that the system can detect and report correct operation for individual game modules.

This waveform shows the game verification (check) signals during simulation. The binary adder check is high, meaning it passed successfully, while the other game checks remain low, indicating they were not active during this period.



Synthesis Process for Verilog Design:

This included installing Yosys found below

```
PROBLEMS    OUTPUT      TERMINAL      PORTS

10-19-71-22:cognichip xxxtentacion$ brew install yosys
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/gtk+3... (3 files, 1.6KB)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/gdk-pixbuf... (207B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/shared-mime-info... (515B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/glib... (64B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/dbus... (223B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/fontconfig... (154KB)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/openssl@3... (64B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/ca-certificates... (64B)
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/gsettings-desktop-schemas
Removing: /Users/xxxtentacion/Library/Logs/Homebrew/gtkwave... (7 files, 1.6M)
Pruned 0 symbolic links and 4 directories from /opt/homebrew
```

Compatibility Challenge:

Syntax error preventing output was the use of “automatic” in the code (not Yosys-compatible Verilog function syntax), so had to be changed in the .sv files

```
10-19-71-22:cognichip xxtentacion$ ls
BINARY_ADDER_FIX_REPORT.md           input_controller.sv
binary_adder_game.sv                 Makefile
connect4_game.sv                     SYNTHESIS_GUDUE.md
CROSS_COMPILER_CONFIGURATION_REPORT.md maze_game.sv
DEPS_USAGE_GUIDE.md                  tb_binary_adder_game.sv
DEPS_VERIFICATION.md                tb_connect4_game.sv
DEPS_XTENTACION.md                  tb_tictactoe_game.sv
dumpfile_2 fst                      PROJECT_STATUS_SUMMARY.md
fulladder.sv                         TUTORIALS_AND_TOOLS_REPORT.md
game_manager.sv                      project.sv
synth_ffpa.y                         README.md
synth_individual_games.y5            READING_ARCADE.md
vericade.y                           tictactoe_autograde_tb.sv
vericade_top.sv                      vericade_game.sv
vericade_verilog.y                  vericade_autoupgrade_tb.sv
vericade_verilog.y5                  vericade_top.sv

10-19-71-22:cognichip xxtentacion$ yosys -v
yosys 0.62 (git sha: 7326bb7d6641508ec285c291a54a662cb1e76cf, clang++ 17.0.0 -fPIC -O3)
10-19-71-22:cognichip xxtentacion$ make synth
Synthesizing Complete Vericade System

10-19-71-22:cognichip xxtentacion$ yosys -v
yosys 0.62 (git sha: 7326bb7d6641508ec285c291a54a662cb1e76cf, clang++ 17.0.0 -fPIC -O3)
-- Executing script file 'synth_vericade.y' --
1. Executing Verilog-2005 frontend: full_adder.sv
Parsing SystemVerilog input from 'full_adder.sv' to AST representation.
Generating RTLIL representation for module `full_adder'.
Successfully finished Verilog frontend.

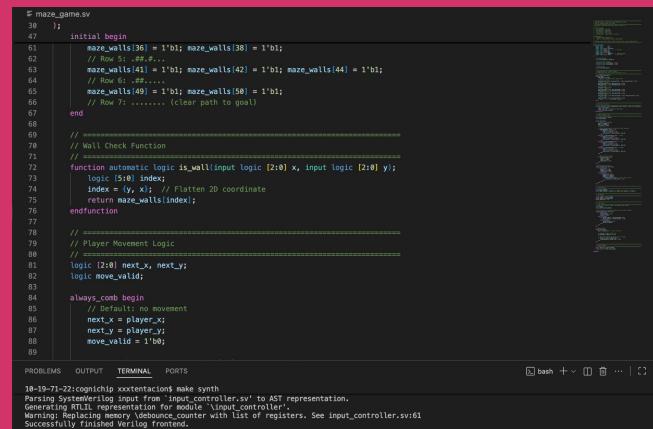
2. Executing Verilog-2005 frontend: input_controller.sv
Parsing SystemVerilog input from 'input_controller.sv' to AST representation.
Generating RTLIL representation for module `input_controller'.
Warning: Replacing memory $ebounce_counter with list of registers. See input_controller.sv:61
Successfully finished Verilog frontend.

3. Executing Verilog-2005 frontend: matrix_driver.sv
Parsing SystemVerilog input from 'matrix_driver.sv' to AST representation.
Generating RTLIL representation for module `matrix_driver'.
Successfully finished Verilog frontend.

4. Executing Verilog-2005 frontend: binary_adder_game.sv
Parsing SystemVerilog input from 'binary_adder_game.sv' to AST representation.
Generating RTLIL representation for module `binary_adder_game'.
Successfully finished Verilog frontend.

5. Executing Verilog-2005 frontend: maze_game.sv
maze_game.sv:75: ERROR: syntax error, unexpected TOK_ID
Parsing SystemVerilog input from 'maze_game.sv' to AST representation.

Synthesis complete!
Output files:
  • Quick Stats:
    prep: synthesis.stats.txt: No such file or directory
10-19-71-22:cognichip xxtentacion$
```



```
10-19-71-22:cognichip xxtentacion$ ls
SYNTHESIS_GUDUE.md
SYNTHESIS_README.md
tb_binary_adder_game.sv
tb_connect4_game.sv
tb_tictactoe_game.sv
tb_maze_game.sv
tb_tictactoe_game.sv
TUTORIALS_AND_TOOLS_REPORT.md
PROJECT_STATUS_SUMMARY.md
project.sv
READING_ARCADE.md
README.md
READING_ARCADE.md
tictactoe_autograde_tb.sv
vericade_game.sv
vericade_autoupgrade_tb.sv
vericade_top.sv

10-19-71-22:cognichip xxtentacion$ yosys -v
yosys 0.62 (git sha: 7326bb7d6641508ec285c291a54a662cb1e76cf, clang++ 17.0.0 -fPIC -O3)
-- Executing script file 'synth_vericade.y' --
1. Executing Verilog-2005 frontend: full_adder.sv
Parsing SystemVerilog input from 'full_adder.sv' to AST representation.
Generating RTLIL representation for module `full_adder'.
Successfully finished Verilog frontend.

2. Executing Verilog-2005 frontend: input_controller.sv
Parsing SystemVerilog input from 'input_controller.sv' to AST representation.
Generating RTLIL representation for module `input_controller'.
Warning: Replacing memory $ebounce_counter with list of registers. See input_controller.sv:61
Successfully finished Verilog frontend.

3. Executing Verilog-2005 frontend: matrix_driver.sv
Parsing SystemVerilog input from 'matrix_driver.sv' to AST representation.
Generating RTLIL representation for module `matrix_driver'.
Successfully finished Verilog frontend.

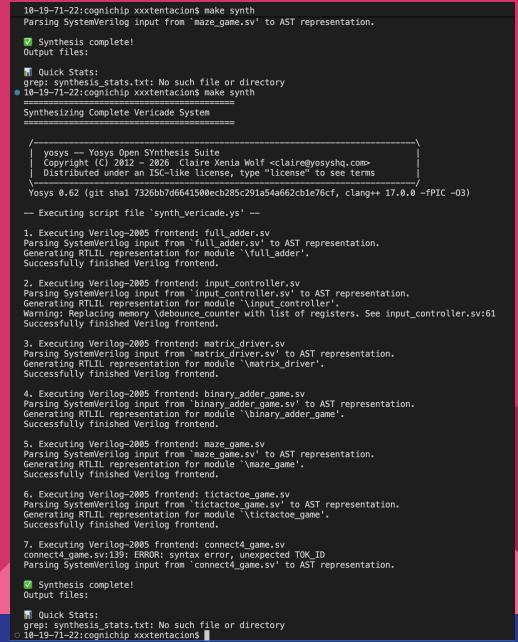
4. Executing Verilog-2005 frontend: binary_adder_game.sv
Parsing SystemVerilog input from 'binary_adder_game.sv' to AST representation.
Generating RTLIL representation for module `binary_adder_game'.
Successfully finished Verilog frontend.

5. Executing Verilog-2005 frontend: maze_game.sv
Parsing SystemVerilog input from 'maze_game.sv' to AST representation.
Generating RTLIL representation for module `maze_game'.
Successfully finished Verilog frontend.

6. Executing Verilog-2005 frontend: tictactoe_game.sv
Parsing SystemVerilog input from 'tictactoe_game.sv' to AST representation.
Generating RTLIL representation for module `tictactoe_game'.
Successfully finished Verilog frontend.

7. Executing Verilog-2005 frontend: connect4_game.sv
connect4_game.sv:139: ERROR: syntax error, unexpected TOK_ID
Parsing SystemVerilog input from 'connect4_game.sv' to AST representation.

Synthesis complete!
Output files:
  • Quick Stats:
    prep: synthesis.stats.txt: No such file or directory
10-19-71-22:cognichip xxtentacion$
```

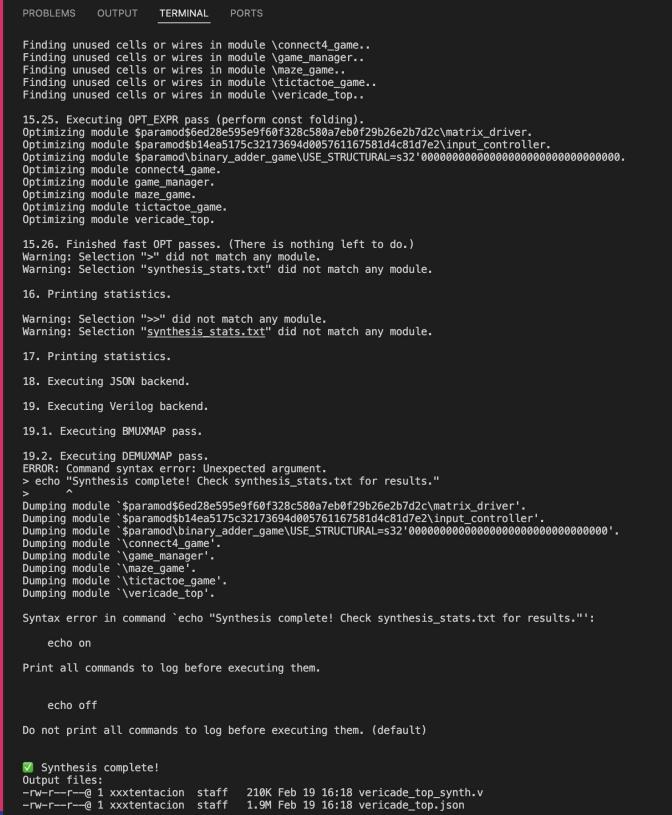


```
10-19-71-22:cognichip xxtentacion$ ls
SYNTHESIS_GUDUE.md
SYNTHESIS_README.md
tb_binary_adder_game.sv
tb_connect4_game.sv
tb_tictactoe_game.sv
tb_maze_game.sv
tb_tictactoe_game.sv
TUTORIALS_AND_TOOLS_REPORT.md
PROJECT_STATUS_SUMMARY.md
project.sv
READING_ARCADE.md
README.md
READING_ARCADE.md
tictactoe_autograde_tb.sv
vericade_game.sv
vericade_autoupgrade_tb.sv
vericade_top.sv

10-19-71-22:cognichip xxtentacion$ make synth
Parsing SystemVerilog input from 'maze_game.sv' to AST representation.
Synthesis complete!
Output files:
  • Quick Stats:
    prep: synthesis.stats.txt: No such file or directory
10-19-71-22:cognichip xxtentacion$
```

System Integration Verified — Successful Synthesis:

- The synthesis completed successfully
- All modules compiled and integrated without critical errors
- Output files generated correctly
- Structural design validated for hardware implementation
- System ready for deployment and real-time testing



A terminal window showing synthesis results. The tabs at the top are PROBLEMS, OUTPUT, TERMINAL (which is selected), and PORTS. The terminal output shows the following:

```
Finding unused cells or wires in module \connect4_game..
Finding unused cells or wires in module \game_manager..
Finding unused cells or wires in module \maze_game..
Finding unused cells or wires in module \tictactoe_game..
Finding unused cells or wires in module \vericade_top..

15.25. Executing OPT_EXPR pass (perform const folding).
Optimizing module $paramod$ed2e595e9f6f328c580a7eb0f29b26e2bd2\matrix_driver.
Optimizing module $paramod$14ea5175c32173694d005761167581d4c81d7e2\input_controller.
Optimizing module $paramod$binary_adder_game\USE_STRUCTURAL=s32 00000000000000000000000000000000.
Optimizing module connect4_game.
Optimizing module game_manager.
Optimizing module maze_game.
Optimizing module tictactoe_game.
Optimizing module vericade_top.

15.26. Finished fast OPT passes. (There is nothing left to do.)
Warning: Selection ">" did not match any module.
Warning: Selection "synthesis_stats.txt" did not match any module.

16. Printing statistics.
Warning: Selection ">" did not match any module.
Warning: Selection "synthesis_stats.txt" did not match any module.

17. Printing statistics.
18. Executing JSON backend.
19. Executing Verilog backend.
19.1. Executing BMUXMAP pass.
19.2. Executing DEMUXMAP pass.
ERROR: Command syntax error: Unexpected argument.
> echo "Synthesis complete! Check synthesis_stats.txt for results."
> ^M
> Dumping module $paramod$ed2e595e9f6f328c580a7eb0f29b26e2bd2\matrix_driver'.
Dumping module $paramod$14ea5175c32173694d005761167581d4c81d7e2\input_controller'.
Dumping module $paramod$binary_adder_game\USE_STRUCTURAL=s32 00000000000000000000000000000000'.
Dumping module \connect4_game'.
Dumping module \game_manager'.
Dumping module \maze_game'.
Dumping module \tictactoe_game'.
Dumping module \vericade_top'.

Syntax error in command `echo "Synthesis complete! Check synthesis_stats.txt for results."`:
echo on
Print all commands to log before executing them.

echo off
Do not print all commands to log before executing them. (default)

 Synthesis complete!
Output files:
-rw-r--r--@ 1 xxxtentacion  staff  210K Feb 19 16:18 vericade_top_synth.v
-rw-r--r--@ 1 xxxtentacion  staff  1.9M Feb 19 16:18 vericade_top.json
```

GitHub Link:

https://github.com/noahrivera/Vericade_CogniChip-Hackathon/tree/main

Thank you!

(shoutout to Cognichip for hosting and sponsoring this hackathon!)