Advant of FPGA  Implementation Platforms

| Day | Part 1 | Part2 | CROC | FURY | LIFE | Verilator TB | FPGA related comments about the puzzles solution | Branch Links |
|---|---|---|---|---|---|---|---|---|
| 1 | * | * | √ | | | | Not an fpga, rather a 2mmx2mm SOC design (pulp/croc) with puzzle solution hardware with a riscV running C code reading uart puzzle data and forwading to an obs memory mapped hardware solver. Full chip simulation in verilator, gds generated and timing closed on a tapeout referenced design. Fun. | https://github.com/ericpearson1313/croc/blob/aoc_day1/README.md#advant-of-code-day-1 |
| 2 | * | * | √ | | | √ | Fpga showcases implementing pattern matching to run in parrallel each cycle. Part 1 was done in croc and part 2 in a testbench. | https://github.com/ericpearson1313/croc/blob/aoc_day2/README.md#day-2-implemented-as-a-chip-for-the-2025-day-2-part-1-puzzle-solution-generation |
| 3 | * | * | | √ | | | FPGA masterd PCIe direct read of puzzle data from host memory. Also FPGA utilizing 4xLVDS outputs to generate simple HDMI wvga showing live (as of raster) fpga logic signals including PCIe monitors, git commit, and our puzzle solution. | https://github.com/ericpearson1313/FlatFury/blob/aoc_day3/README.md#advent-of-code-2025-day-3 |
| 4 | * | * | | | √ | | FPGA's block RAM total maximum bandwidth enables processing a 45x44cells array each cycle. Utillzed direct LUT programming for LAB carry path utilization and logic compaction. Unique addition of video rate cell counter for the puzzle, avoids having the counter in the torential generation loop. | https://github.com/ericpearson1313/fpga_life/tree/aoc25_day4?tab=readme-ov-file#aoc-day-4-part-1-and-2 |
| 5 | * | * | | | | √ | FPGAs are adept at systolic processing in this case comparing data against 200 programmable ranges as it steps past them. Part 2 I just did as quick linked list C code but looks like it could take adantage of an FPGAs shifting ability for the linked list range insertion. | https://github.com/ericpearson1313/croc/blob/aoc_day5/README.md#day-5---part-1 |
| 6 | * | * | | | | √ | FPGAs block rams are nicely useful as row buffers for moving window algorithms and FPGAs can using physical instances in parallel to solve 2 different problems (part 1 and part 2) in parallel both fed by the same input stream. | https://github.com/ericpearson1313/croc/blob/aoc_day5/README.md#day-6---part-1--2 |
| 7 | * | * | | | √ | | FPGAs displays nice christmas tree! Puzzle was stored in the FPGAs on-board flash and solved at video rate for direct HDMI display. | https://github.com/ericpearson1313/fpga_life/tree/aoc_day7#aoc-day-7-part-1-and-2 |
| 8 | * | * | | | | √ | FPGAs don't have always have a computational advantage. Day 8 solution is an example of a serial problem not(?) best solved with an FPGA. First did solutions with behavior verilog. However it does map to what should be a mezmerizing and colorful HDMI display when run on a FPGA. (to be completed) | https://github.com/ericpearson1313/croc/blob/aoc_day8/README.md#aoc-2025-day-8-part-2-puzzle-solution-generation-in-verilog |
| 9 | * | * | | | | √ | Multiple FPGA block RAMs configured as single write, multiple read is a nice area traqeoff for repeated 100k cycle searchs of coordinate pairs with pipelined comparison. I just did part2 with a pencil and paper. I looked complicated, FPGA or othewise. | https://github.com/ericpearson1313/croc/blob/aoc_day9/README.md#aoc-2025-day-9-puzzle-solution-generation-in-verilog |
| 10 | * | | | | | √ | FPGA eats up this problem with XOR array parallelism and adaptive search sizes. Uses AXI style stream input with handshake. Part 2 will need step through the solutions of variably constrained linear eqns (10x10), I don't have that handy, but will think on it. | https://github.com/ericpearson1313/croc/blob/aoc_day10/README.md#aoc-2025-day-10-puzzle-solution-generation-in-verilog |
| 11 | * | * | | √ | √ | √ | FPGA directly synthesized the puzzle text translated line per line to verilog. Showcases FPGA as both reconfigurable and capable of fully parallel designs giving an incredable 164ns latency to solve puzzle 1, and and 3 cycles totalling 547ns, for part 2. Implemented this design on 3 platforms. It was interesting to note the difference in compile time and performance between platforms. | https://github.com/ericpearson1313/croc/blob/aoc_day11/README.md#aoc-2025-day-11-puzzle-solution-generation-in-verilog<br>https://github.com/ericpearson1313/fpga_life/tree/aoc_day11?tab=readme-ov-file#aoc-day-11-part-2<br>https://github.com/ericpearson1313/FlatFury/blob/aoc_day11/README.md#advent-of-code-2025-day-11-parts-1--2 |
| 12 | * | | | | | √ | FPGAs processing of at-rate data per cycle processing and pruning data sets. | https://github.com/ericpearson1313/croc/blob/aoc_day12/README.md#aoc-2025-day-12-part-1-puzzle-solution-generation-in-verilog |

Legend

| | | |
|---|---|---|
| Synthesize-able Verilog | Behvioural Verilog | Other |

| | PLATFORMS | Links |
|---|---|---|
| CROC | The pulp croc riscV mini SOC is a great basis for experimentation and it tapeout referenced. The IIC-OSIC-TOOLs container has all the tools needed for the full flow. | https://github.com/pulp-platform/croc?tab=readme-ov-file#croc-system-on-chip |
| FURY | The LiteFury is a low cost FPGA board for learning PCIe. In an M.2 format it can connect to many platforms. It utilizes a 100K LE Artix-7 fpga. It hosted easily on a raspberry Pi5 running ubuntu, and uses 4x LVDS outputs to drive an FPGA. The puzzle solutions were branched off simplified branch of the vendor repo using an AXI PCIe core rather than the vendor provided DMA PCIe core, because. | https://github.com/RHSResearchLLC/NiteFury-and-LiteFury?tab=readme-ov-file#nitefury-and-litefury-xilinx-fpga-development-board-kit-in-m2-form-factor |
| LIFE | A custom board for the low cost MAX10 FPGA. The MAX10 has on board flash for configuration and data, 4x LVDS pairs each for HDMI/DVI outputs are connected allowing display of puzzle results. A push button allows stepping and running the algorithms. Forked from my conways game of life accellerator project. | |
| Verilator | A behavioral system verilog testbench can be used to read the puzzle file and drive the input signals and monitor and display output signals. A separate synthesizable system verilog puzzle solving module doing the compute. I/O between them was kept simple and area utilizations were kept within reasonable limits, but not place, route or timing closure was done. The handy IIC-OSIC-TOOL container was used in the croc repo above. | |