
Project F - FPGA Development

Project F is a little oasis where you can quench your thirst for FPGA knowledge and find accessible, open-source designs to learn from and build on. Our projects include the *Verilog Library*, *FPGA Graphics*, and *FPGA Maths* tutorial series.

The Project F website features fifty blog posts on a wide range of FPGA and Verilog topics.

Follow @WillFlux@mastodon.social for updates. Join the Project F Discussions and 1BitSquared Discord.



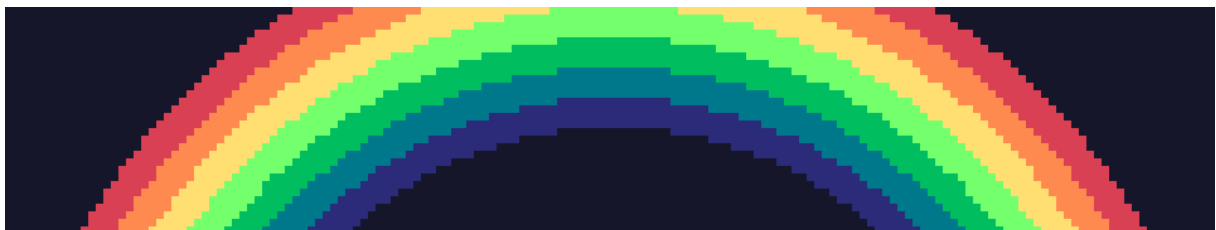
Verilog Library

The Project F Library includes handy Verilog designs for everyone. From framebuffer and video output to division and square root, ROM and RAM, and even circle drawing. You can freely build on these MIT licensed designs.

Visit the Library for the Verilog designs or get an overview from the Verilog Library blog post.

FPGA Graphics

In this series, we learn about graphics at the hardware level and get a feel for the power of FPGAs. We'll learn how screens work, play Pong, create starfields and sprites, paint Michelangelo's David, draw lines and triangles, and animate characters and shapes. Along the way, you'll experience a range of designs and techniques, from memory and finite state machines to crossing clock domains and translating C algorithms into Verilog.



- **Beginning FPGA Graphics:** Designs - Blog

-
- **Racing the Beam:** Designs - Blog
 - **FPGA Pong:** Designs - Blog
 - **Display Signals:** Blog (no designs in git)
 - **Hardware Sprites:** Designs - Blog
 - **Framebuffers:** Designs - Blog
 - **Lines and Triangles:** Designs - Blog
 - **2D Shapes:** Designs - Blog
 - **Animated Shapes:** Designs - Blog

Hello

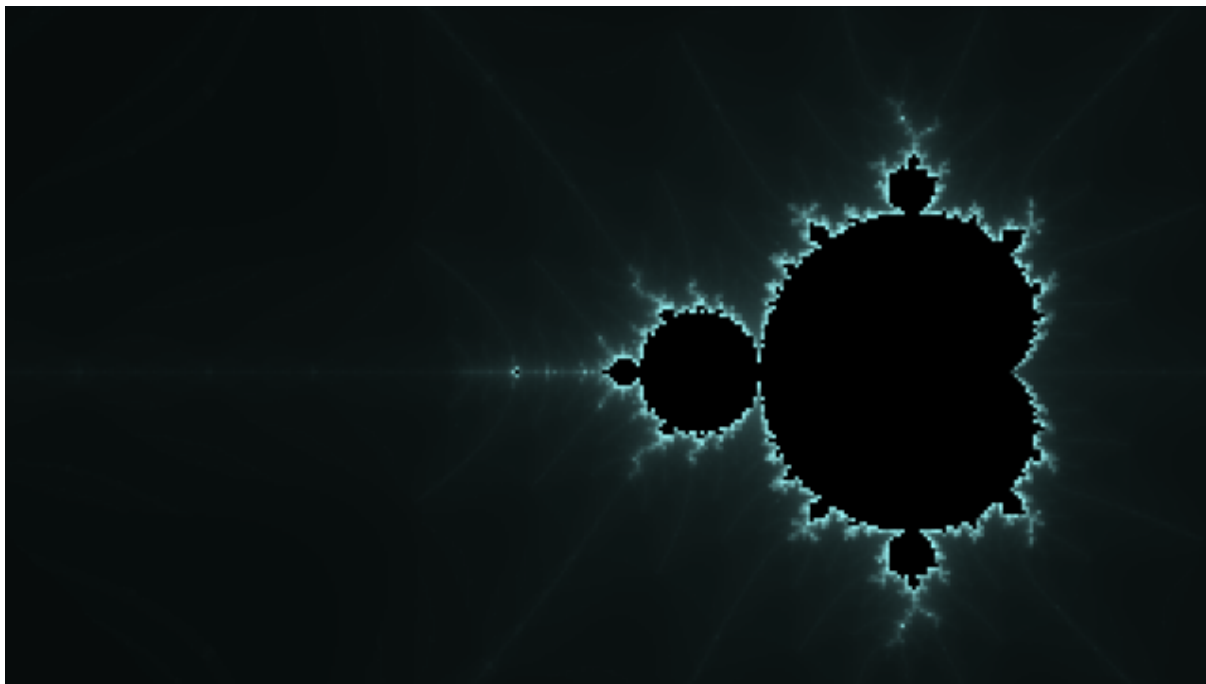
A three-part introduction to FPGA development with Verilog with dev boards:

- **Hello Arty:** Designs - Blog 1 - Blog 2 - Blog 3
- **Hello Nexys:** Designs - Blog 1 - Blog 2

Maths and Algorithms

Maths & Algorithms is our latest tutorial series:

- Numbers in Verilog - introduction to numbers in Verilog
- Vectors and Arrays - working with Verilog vectors and arrays
- Multiplication with DSPs - efficient FPGA multiplication
- Fixed-Point Numbers - precision without complexity
- Division in Verilog - divided we stand
- *More maths soon*



Demos and Effects

- **Ad Astra:** Designs - Blog - greetings with starfields and hardware sprites
- **Castle Drawing:** Designs - Blog - draw a castle and rainbow in 16 colours
- **Life on Screen:** Designs - Blog - Conway's Game of Life in logic
- **Mandelbrot Set:** Designs - Blog - Mandelbrot set with fixed-point maths
- **Rasterbars:** Designs - Blog - classic animated colour bars
- **Sine Scroller:** Designs - Blog - greet your viewers in style



Requirements

FPGA Architecture

Our designs seek to be vendor-neutral, but some functionality requires support for vendor primitives. We currently support two FPGA architectures:

- **XC7** - Xilinx 7 Series FPGAs, such as Spartan-7 and Artix-7
 - BUFG, MMCME2_BASE
 - HDMI support: OBUFDS, OSERDES2
- **iCE40** - Lattice iCE40 FPGAs, such as iCE40 UltraPlus
 - SB_IO, SB_PLL40_PAD, SB_SPRAM256KA

We also infer block ram (BRAM); see lib/memory.

Porting to other architectures should be straightforward.

SystemVerilog?

We use a few simple features of SystemVerilog to make Verilog more pleasant:

-
- `logic` type is safer and less work than using `wire` and `reg`
 - `always_comb` and `always_ff` to make intent clear and catch mistakes
 - `$clog2` to calculate vector widths (e.g. for addresses)
 - `enum` to make finite state machines simpler to work with
 - Matching names in module instances: `.clk_pix` instead of `.clk_pix(clk_pix)`

I believe these features are helpful, especially for beginners. All the SystemVerilog features are compatible with recent versions of Verilator, Yosys, Icarus Verilog, and Xilinx Vivado.

Thank You, Sponsors!

Thank you to all my sponsors for supporting Project F. Special thanks go to the following: David C. Norris, Didier Malenfant, LaDirth, matt venn, Paul Sajna, and STjurny for their recent generosity.