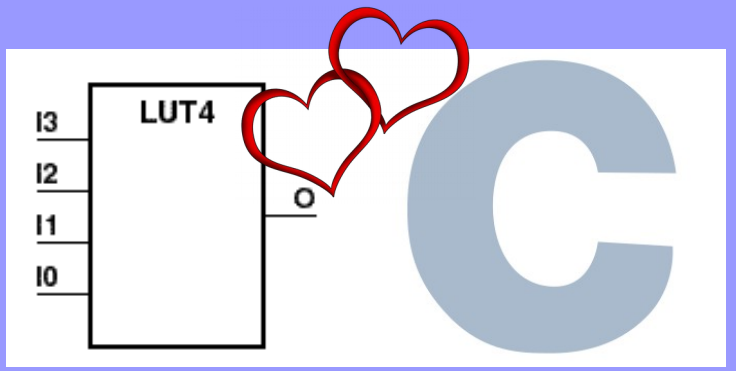
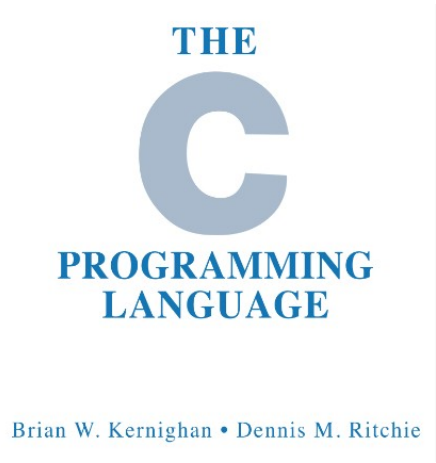
CflexHDL



Design digital circuits in C

Simulate really fast *with a regular compiler*

Language benefits



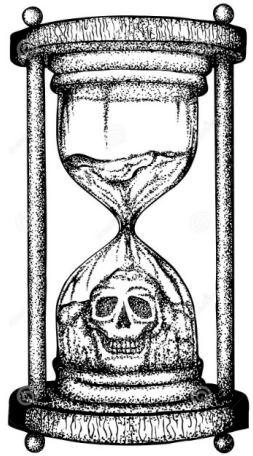
- C is much more known than

  Verilog/VHDL
- Helps to break the "scary" barrier to enter circuit design by embedded software developers
- Automatic FSMs from control flow ("while loops")
- Comfortable debugging (tools new and upcoming like gdbwave)

Simulation speed benefits

- Running compiled C code is even faster than fast HDL simulators
- Simulator runs at **220MHz** clock speed (LED glow demo)
- The 3D-like video demo surpasses realtime, even with a 15 years old CPU: **126FPS - 38M** pixel/s
- Original demo: 0.82 FPS (**156X** slower) - *Silice's "make verilator"*

Development time benefits



- Compiling C is much faster than doing synthesis (seconds instead of minutes)
- **240X** to **1028X** faster than common synthesis toolchains (led glow demo)
- Notable exception: Yosys + NextPNR (just 67X slower)

Led glow demo

```
void led_glow(uint1_t& led)
{
  union {
    uint25_t counter;
    struct { uint25_t :16; uint25_t mid:8; uint25_t msb:1; }
        counter_bits;
  };

  union {
    uint9_t PWM;
    struct { uint8_t lsb:8; uint8_t msb:1; } PWM_bits;
  };

  uint8_t intensity;

  counter = 0;
  while(always())
  {
    intensity = counter_bits.msb ? counter_bits.mid : ~counter_bits.mid;
    PWM = PWM_bits.lsb + intensity;

    led = PWM_bits.msb;
    counter = counter + 1;
  }
}
```

Argument *led* represents the output pin

Bit range access using C Bitfields, and a union for aliasing

When not translated to verilog, the *always()* function defers execution to the simulator

Sets bit output and updates register. The function never returns

Synthesis Workflow

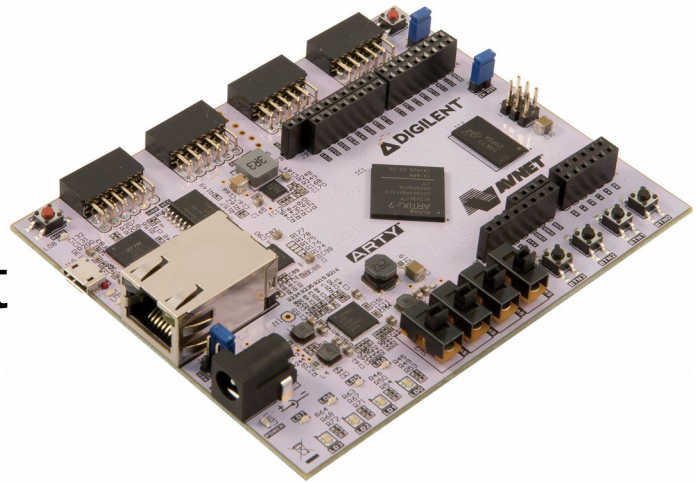
The C files are translated to verilog and integrated into existing toolchains

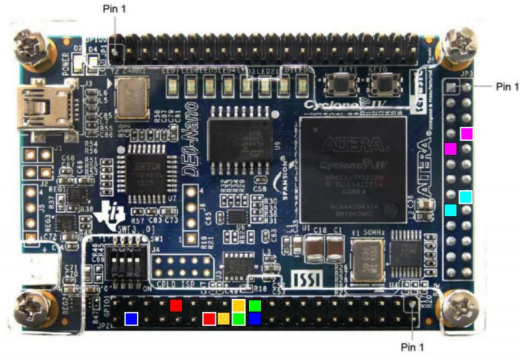
Development innovations

- The tool is novel in itself, but also required the development of non-existing modules such as:
  - **CflexParser**: the first C++ parser/generator for python. Based on clang indexer's bindings (tool to be released stand-alone)
  - Adding previously unsupported boards to Silice projects
  - Graphic display on the simulator: running the original C code or the Verilator output using a same codebase
  - Integration of Yosys+NextPNR into LiteX framework: **mainlined**

Ease of running & testing

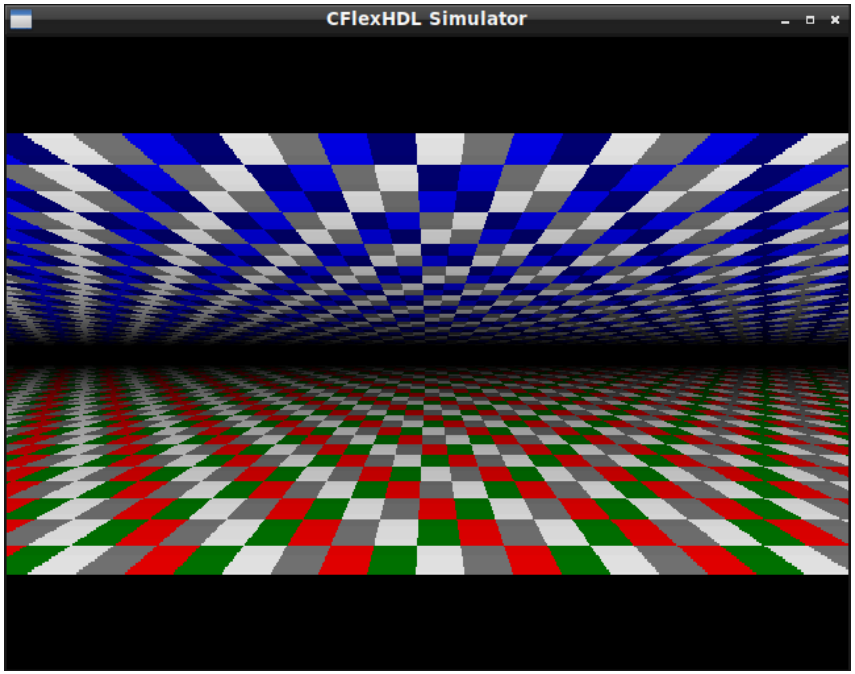
- Compile project (for simulation) or synth the bitstream and load, using just **make**
- Currently supporting:
  - Digilent Arty A7 board (Xilinx Artix-7)
  - Terasic DE0-Nano (Intel Cyclone IV)
  - All boards supported by LiteX, easy to add





3D-like video generation demo

- A VGA demo* was manually ported to C
- CflexHDL runs the C logic using any of:
  - A simulator that opens a window, after compiling the C sources **in 1.3 seconds**
  - A FPGA board with analog or digital video output (automatically generates and loads the bitstream)
  - Verilator output on the simulator window (same graphics but at lower FPS)



** https://github.com/sylefeb/Silice/blob/master/projects/vga_demo/vga_flyover3d.ice*

Example C & generated code (FSM)

```
cur_inv_y = 0;
offs_y = 0;
u = 0;
v = 0;
lum = 0;

while (always())
{
    // prepare next
    pos_u = pos_u + 1024;
    pos_v = pos_v + 1;

    // wait for sync
    while (pix_vblank == 1)
        wait_clk();

}
```

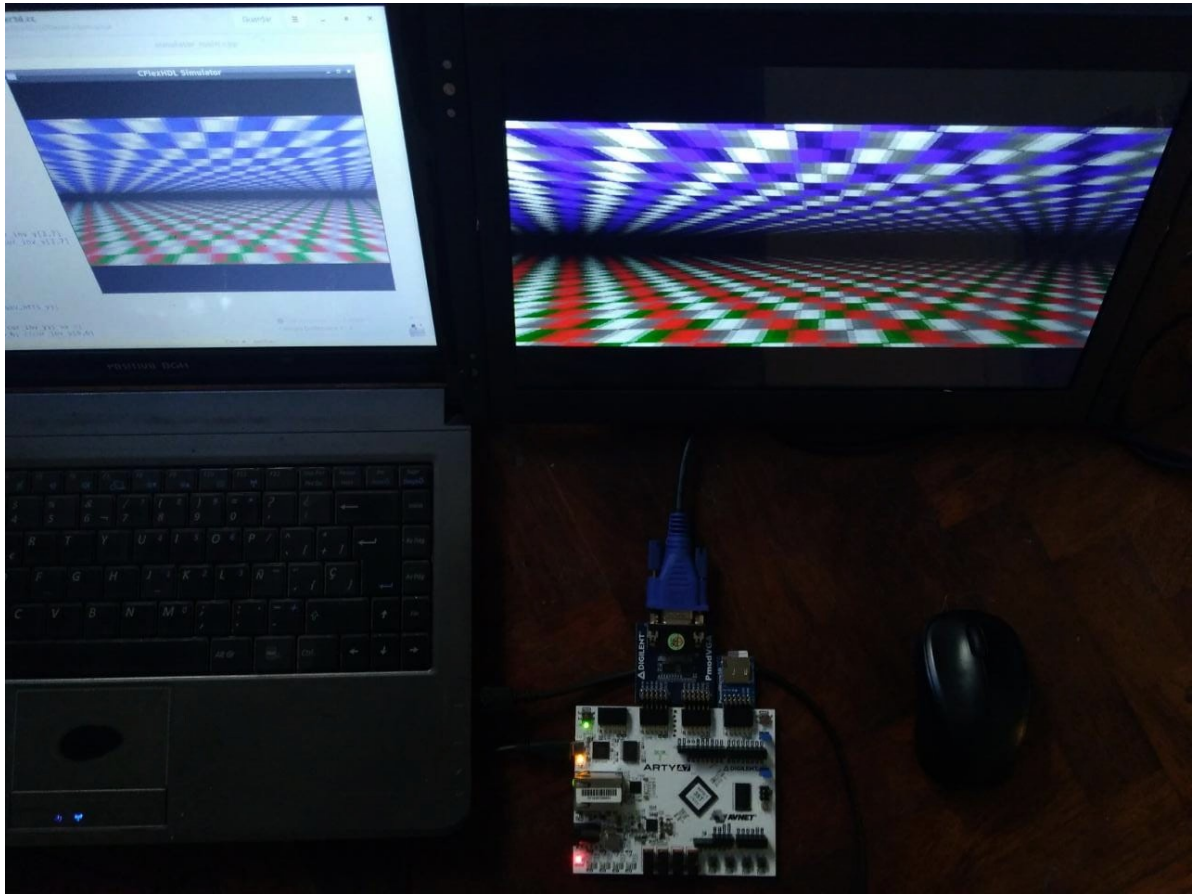
```
// _always_pre
(* full_case *)
case (_q_index)
0: begin
// _top
_d_cur_inv_y = 0;
_t_offs_y = 0;
_t_u = 0;
_t_v = 0;
_d_lum = 0;
_d_index = 1;
end
1: begin
if (1) begin
_d_index = 3;
end else begin
_d_index = 2;
end
end
4: begin
_d_pos_u = _q_pos_u+1024;
_d_pos_v = _q_pos_v+1;
_d_index = 5;
end
5: begin
if (in_pix_vblank==1) begin
_d_index = 5;
end else begin
_d_index = 1;
end
end
6: begin // end of frame_display
end
default: begin
_d_index = {3{1'bx}};
 end
endcase
// _always_post
end
```

It works! (TM)

$ cd demos/vga

$ make load run

- Makes bitstream,
- loads it,
- compiles the C source,
- and runs it in the simulator window



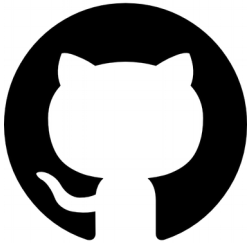
Dependencies

- Sylvain Lefebvre's Silice HDL: https://github.com/sylefeb/Silice
- Clang's indexer bindings: https://pypi.org/project/clang/
- Enjoy Digital's LiteX: https://github.com/enjoy-digital/litex
- Open FPGA Loader: https://github.com/trabucayre/openFPGALoader
- SDL 2: https://www.libsdl.org
- GNU gcc or Clang
- Python 3.x

Planned improvements

- Remove depedency with Silice HDL (needs to add FSM logic generation)
- Adopt some C++ syntax like templates to access bit fields
- Port a C soft-float library to support floating point in the FPGA
- Integrate the soon to be released **CflexTypes** library, currently providing:
  - Integer, fixed point and floating point types of variable widths
  - Instrumented templates with operator overloading, for benchmarking resource usage at simulation time and effect of varying bit widths (already tested)

Thanks!



https://github.com/suarezvictor/CflexHDL/