## Space-Time Adventures on Novena

# Introducing: Balboa

Star Simpson
@starsandrobots

Andy Isaacson
@eqe

#balboaFPGA

## What This Talk is About

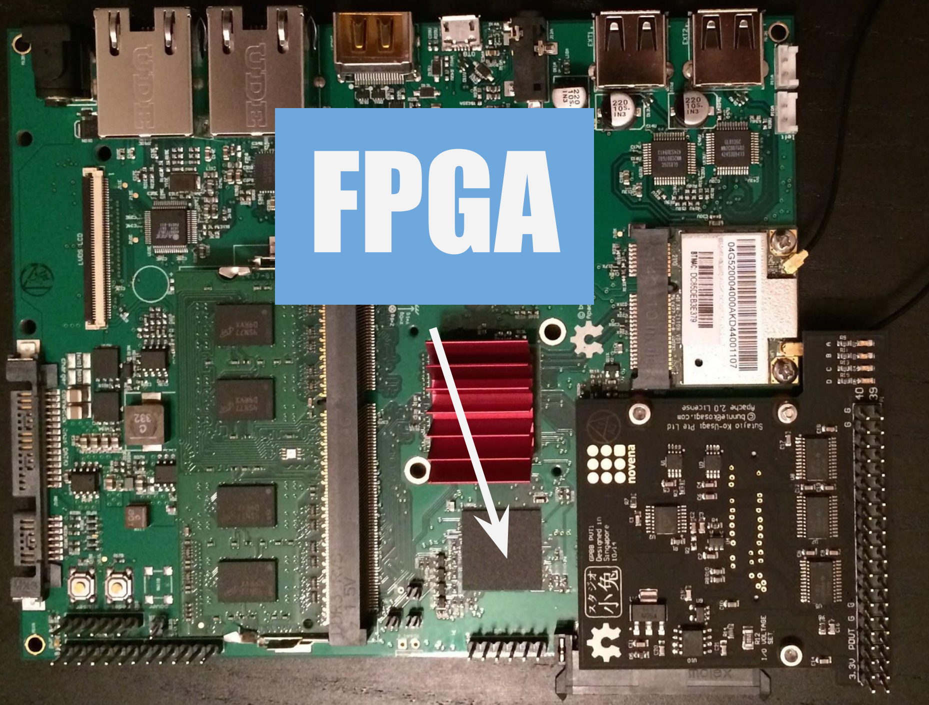
- What is Novena?
- What are FPGAs and why do you use one?
- What tools exist today?
- **Balboa** manifesto & our work done to date

## Background

●●●●

## Novena

## Novena



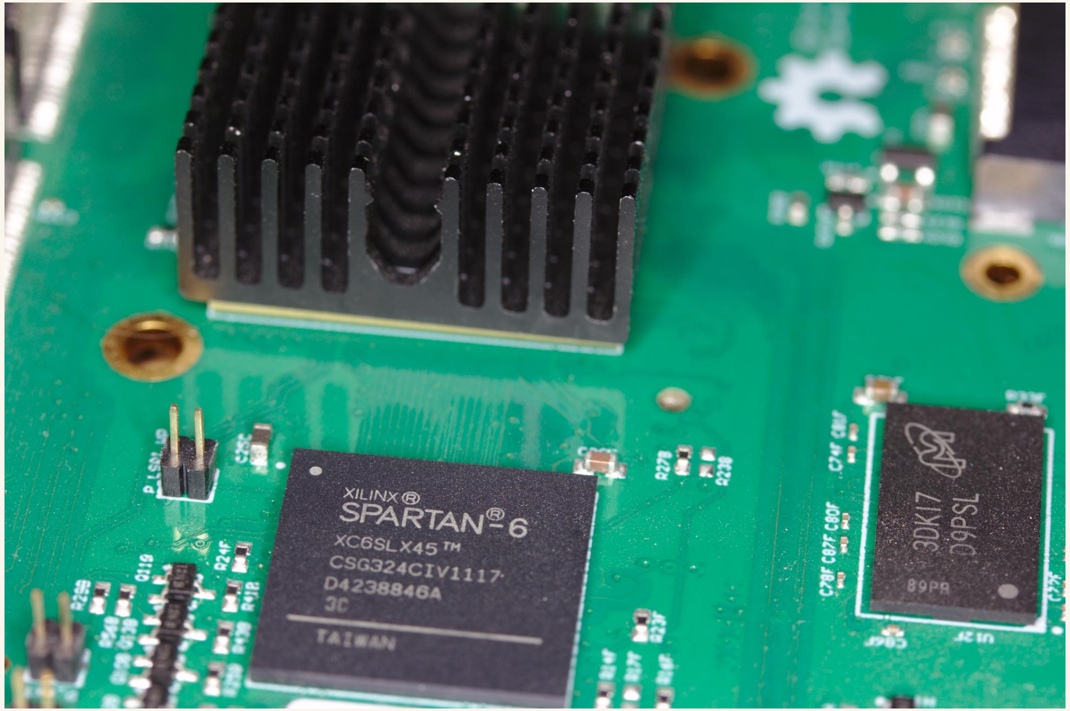
## What is an FPGA?

“Field Programmable Gate Array”

What are FPGAs good for?

## FPGA Terms

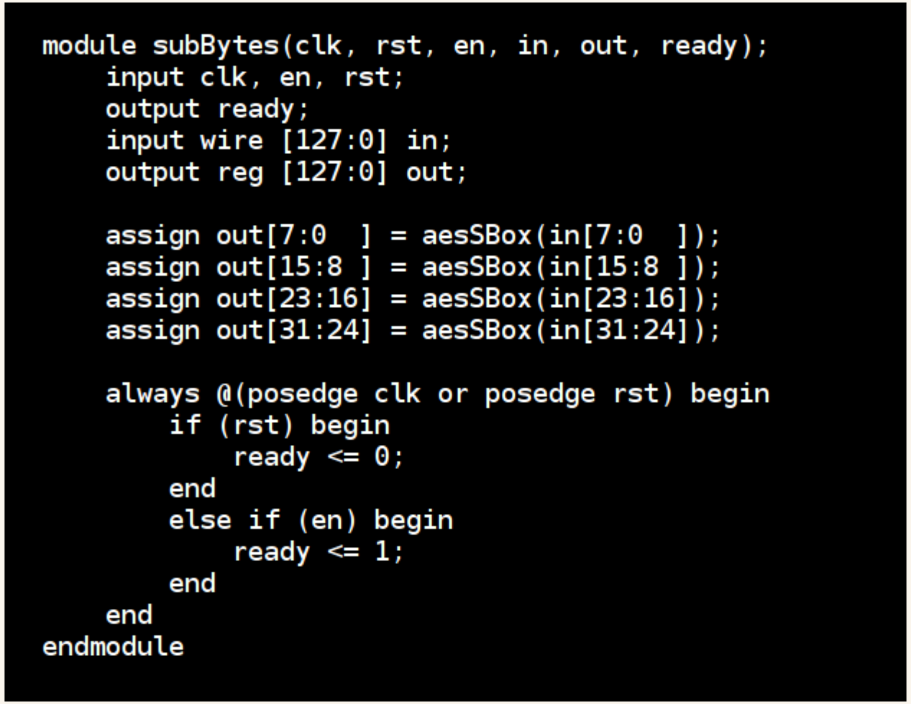
- LUTs
- Logic cells
- Slices
- Fabric
- Cores



## Hardware Description Languages (HDLs)

What is a hardware description language?

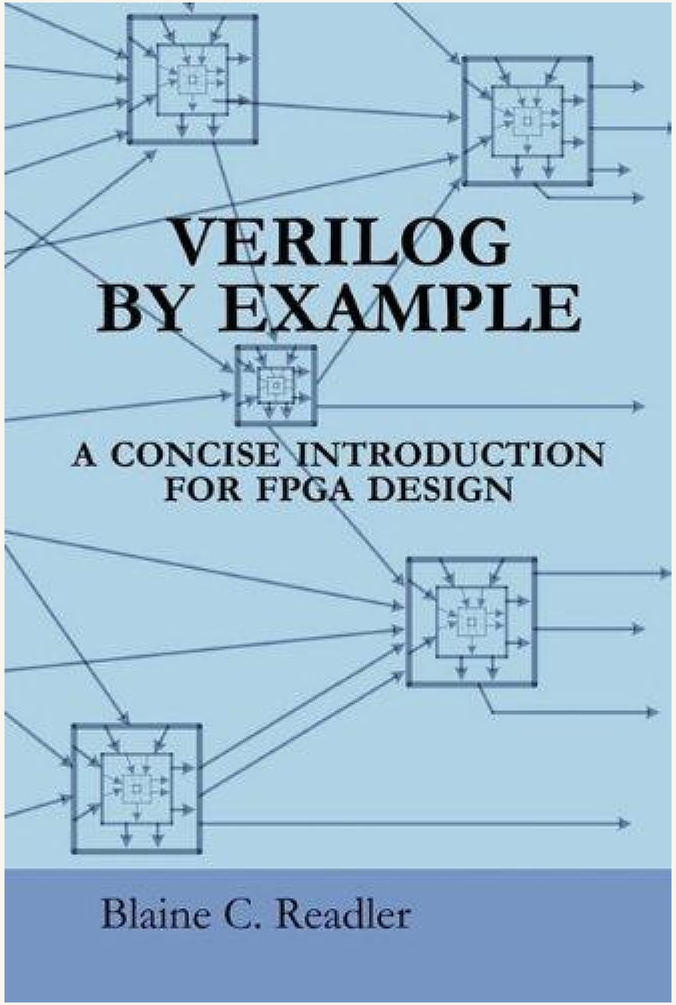
## Verilog



```
module subBytes(clk, rst, en, in, out, ready);
    input clk, en, rst;
    output ready;
    input wire [127:0] in;
    output reg [127:0] out;

    assign out[7:0  ] = aesSBox(in[7:0  ]);
    assign out[15:8 ] = aesSBox(in[15:8 ]);
    assign out[23:16] = aesSBox(in[23:16]);
    assign out[31:24] = aesSBox(in[31:24]);

    always @(posedge clk or posedge rst) begin
        if (rst) begin
            ready <= 0;
        end
        else if (en) begin
            ready <= 1;
        end
    end
endmodule
```

## Verilog Reference

*Verilog by Example*
a good place to start learning Verilog:


VERILOG BY EXAMPLE

A CONCISE INTRODUCTION FOR FPGA DESIGN

Blaine C. Readler

# VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity signed_adder is
  port
  (
    aclr : in   std_logic;
    clk  : in   std_logic;
    a    : in   std_logic_vector;
    b    : in   std_logic_vector;
    q    : out  std_logic_vector
  );
end signed_adder;

architecture signed_adder_arch of signed_adder is
  signal q_s : signed(a'high+1 downto 0); -- extra bit wide

begin -- architecture
  assert(a'length >= b'length)
    report "Port A must be the longer vector if different sizes!"
    severity FAILURE;
  q <= std_logic_vector(q_s);

  adding_proc:
  process (aclr, clk)
    begin
      if (aclr = '1') then
        q_s <= (others => '0');
      elsif rising_edge(clk) then
        q_s <= ('0'&signed(a)) + ('0'&signed(b));
      end if; -- clk'd
    end process;

end signed_adder_arch;
```

## How to Build for an FPGA

- synthesis
- place and route
- bitstream generation

## What is an FPGA Core?

- IP Cores

OpenCores

www.opencores.org

## Environment

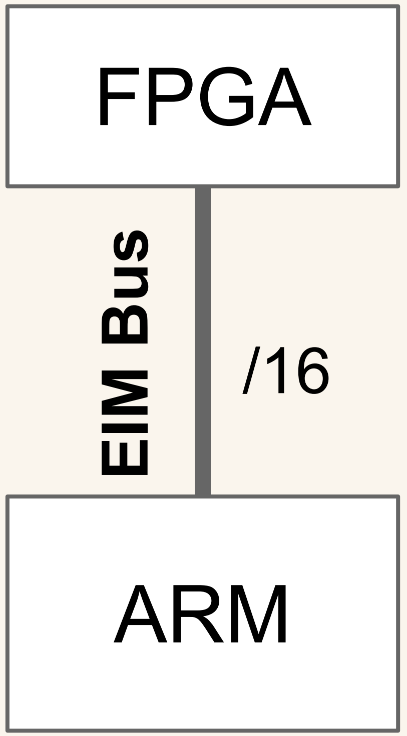
●●●●

# Novena

Spartan 6 LX45

- 43k logic cells
- 6.8k slices
- 401kb distributed RAM
- 58 DSP48A slices
- 2088kb block RAM

## Novena Hardware Platform



## Uses of the FPGA on Novena

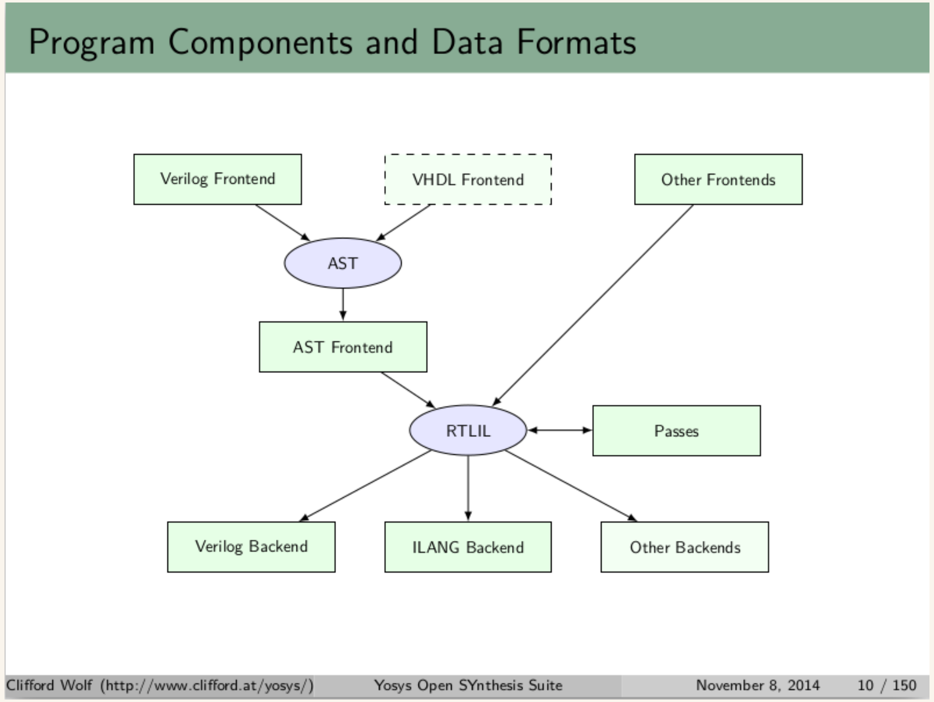
- bitcoin mining
- emulation
- crypto
- coprocessing
- high speed data acquisition
  - paired with 2x 8-bit 500++Msps ADC

## Current OSS Ecosystem for FPGAs

- Yosys
- Migen
- MyHDL
- Chisel

# Yosys

“Yosys is the first full-featured open source software for Verilog HDL synthesis.”


Program Components and Data Formats

Clifford Wolf (http://www.clifford.at/yosys/) Yosys Open SYnthesis Suite November 8, 2014 10 / 150

http://www.clifford.at/yosys/files/yosys_presentation.pdf

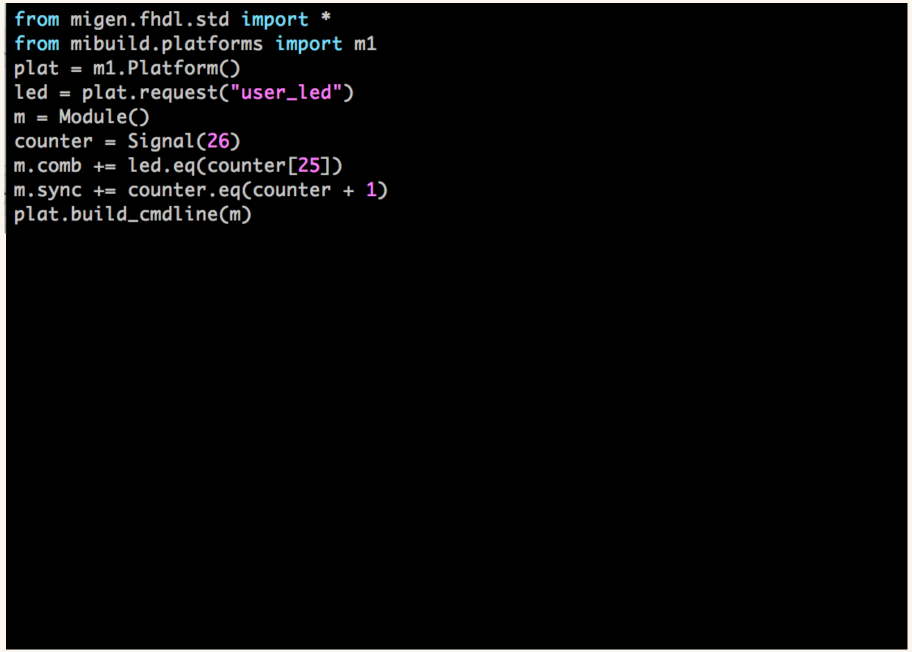
http://www.clifford.at/

## Migen

“Python toolbox for building hardware”

- High level Python description of circuit
- Run Python program to **generate** your HDL
- Used in real live projects - misoc, etc
- Outputs Verilog -or- VHDL

# Migen



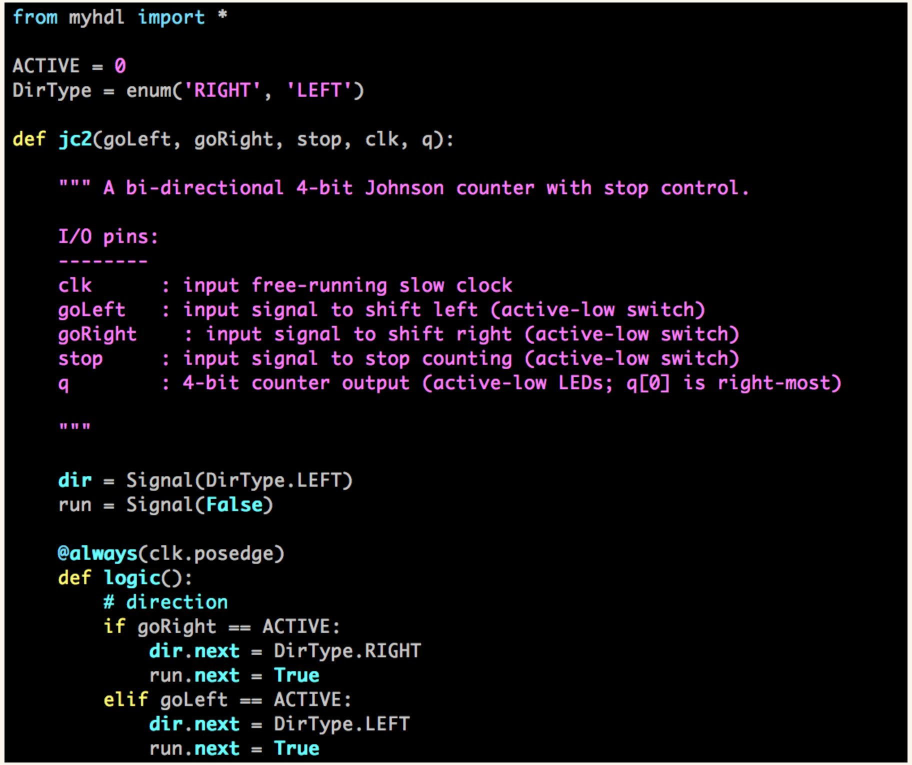
```
from migen.fhdl.std import *
from mibuild.platforms import m1
plat = m1.Platform()
led = plat.request("user_led")
m = Module()
counter = Signal(26)
m.comb += led.eq(counter[25])
m.sync += counter.eq(counter + 1)
plat.build_cmdline(m)
```

## MyHDL

“Design hardware with Python!”

- Closer to hardware than Migen?
- Lots of tools, docs, integration for real usage
- Basically a Python syntax for Verilog?
- Outputs Verilog -or- VHDL

# MyHDL



```
from myhdl import *

ACTIVE = 0
DirType = enum('RIGHT', 'LEFT')

def jc2(goLeft, goRight, stop, clk, q):

    """ A bi-directional 4-bit Johnson counter with stop control.

    I/O pins:
    --------
    clk      : input free-running slow clock
    goLeft   : input signal to shift left (active-low switch)
    goRight    : input signal to shift right (active-low switch)
    stop     : input signal to stop counting (active-low switch)
    q        : 4-bit counter output (active-low LEDs; q[0] is right-most)

    """

    dir = Signal(DirType.LEFT)
    run = Signal(False)

    @always(clk.posedge)
    def logic():
        # direction
        if goRight == ACTIVE:
            dir.next = DirType.RIGHT
            run.next = True
        elif goLeft == ACTIVE:
            dir.next = DirType.LEFT
            run.next = True
```

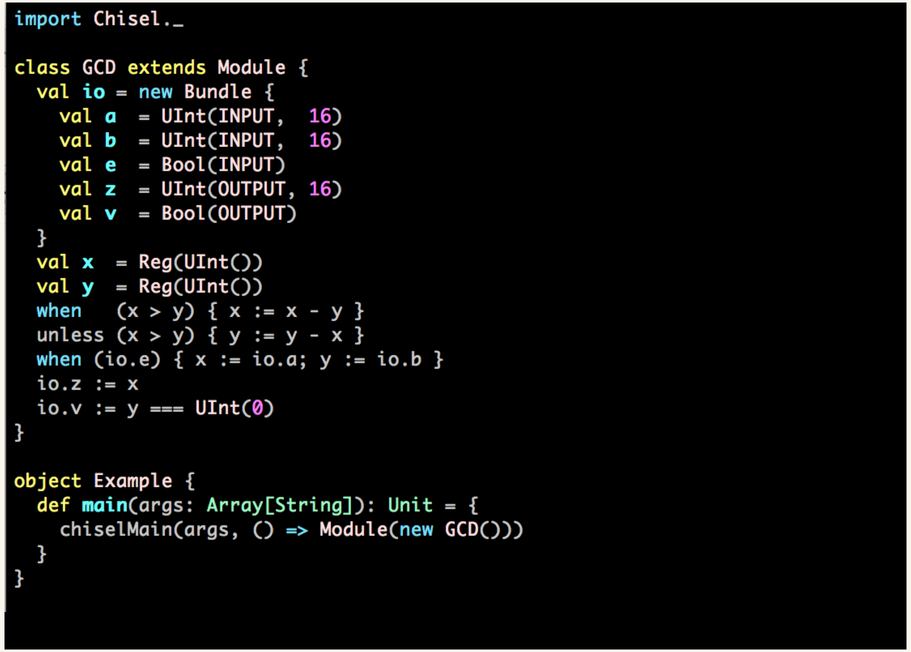
## Chisel

“Hardware construction embedded in Scala”

- Often one-to-one with Verilog
- Uses abstraction to generalize

OO - functional programming - parameterized types

# Chisel

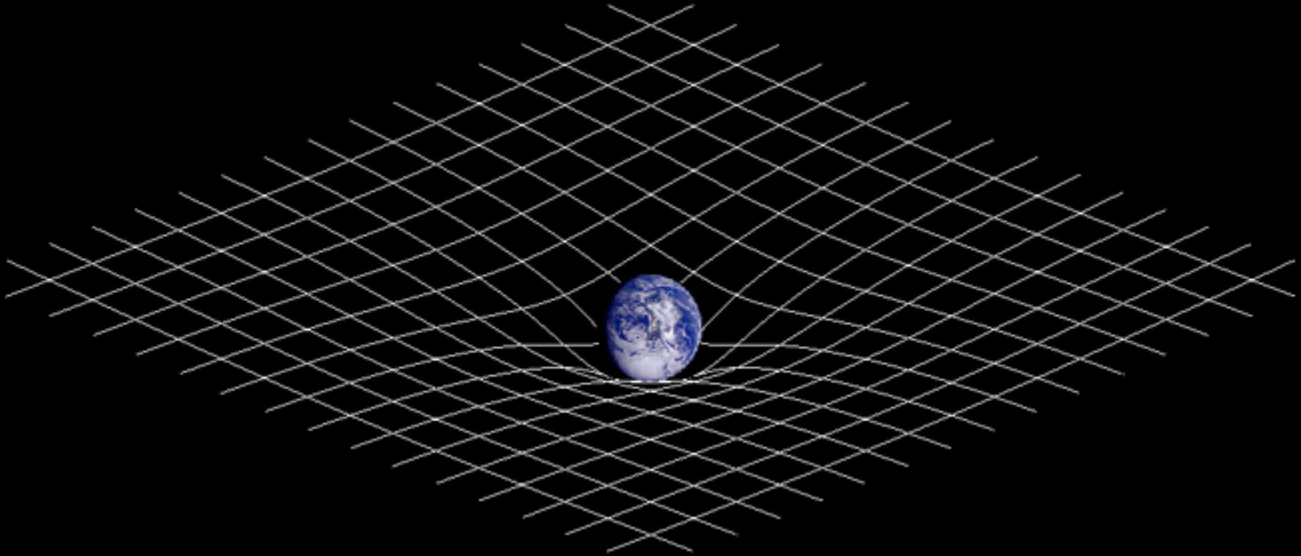


```scala
import Chisel._

class GCD extends Module {
  val io = new Bundle {
    val a  = UInt(INPUT,  16)
    val b  = UInt(INPUT,  16)
    val e  = Bool(INPUT)
    val z  = UInt(OUTPUT, 16)
    val v  = Bool(OUTPUT)
  }
  val x  = Reg(UInt())
  val y  = Reg(UInt())
  when   (x > y) { x := x - y }
  unless (x > y) { y := y - x }
  when (io.e) { x := io.a; y := io.b }
  io.z := x
  io.v := y === UInt(0)
}

object Example {
  def main(args: Array[String]): Unit = {
    chiselMain(args, () => Module(new GCD()))
  }
}
```

## Why We Need a Free Toolchain

- Flexible targeting
- New HDL experiments
- Faster builds
- Longevity

## History of Sharing Space & Time



http://upload.wikimedia.org/wikipedia/commons/2/22/Spacetime_curvature.png

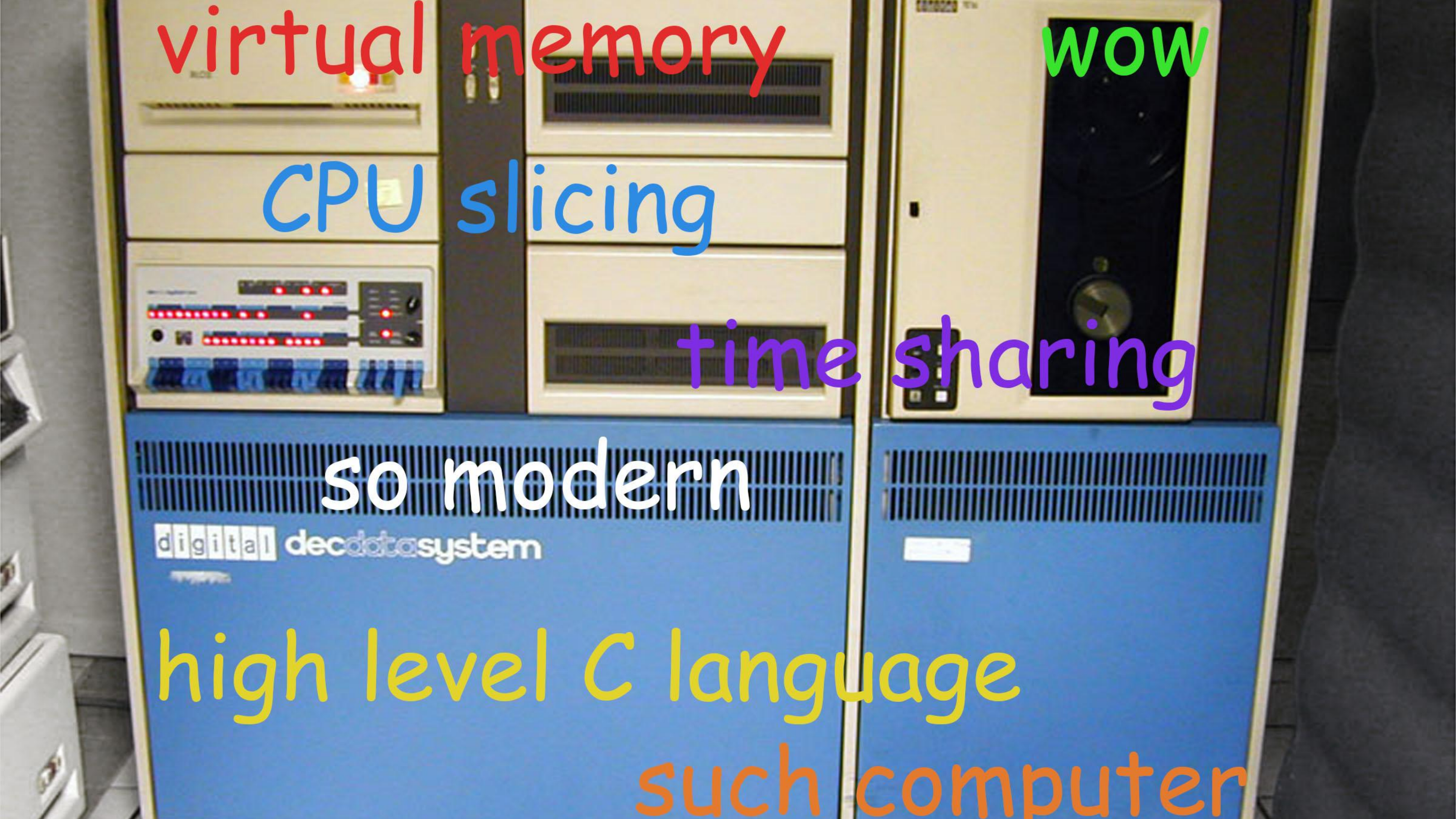
## In the Beginning

Custom CPUs, Applications, Peripherals

All wired together to build a COMPUTER
(ok, at the time this was really cool)

## UNIX, 1972: New Ideas

http://commons.wikimedia.org/wiki/File:PDP-11-70-DDS570.jpg


virtual memory
wow
CPU slicing
time sharing
so modern
digital decdatasystem
high level C language
such computer

## Infrastructure We’ve Gained

- Virtual Memory
- Timesharing
- OS APIs
- Networking
- Device Drivers
- Compilers
- Libraries

## Manifesto

## Balboa Vision

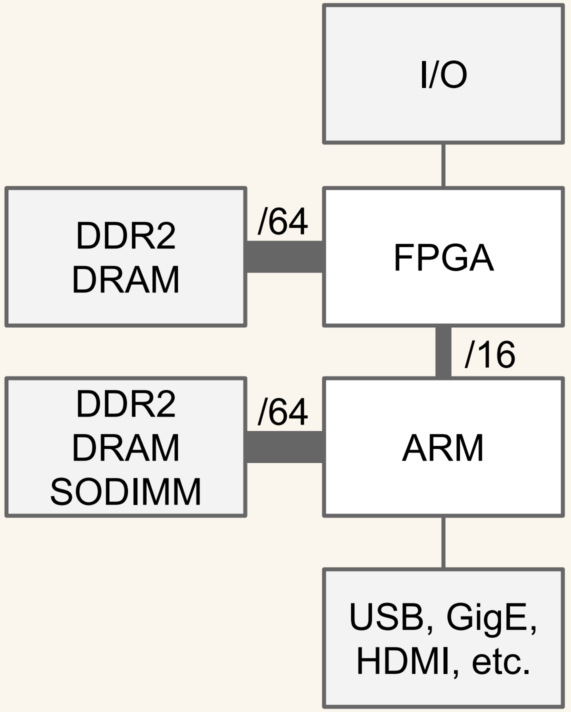
To let us do more than one thing at a time on the FPGA, and to do so flexibly.

## Balboa Definition

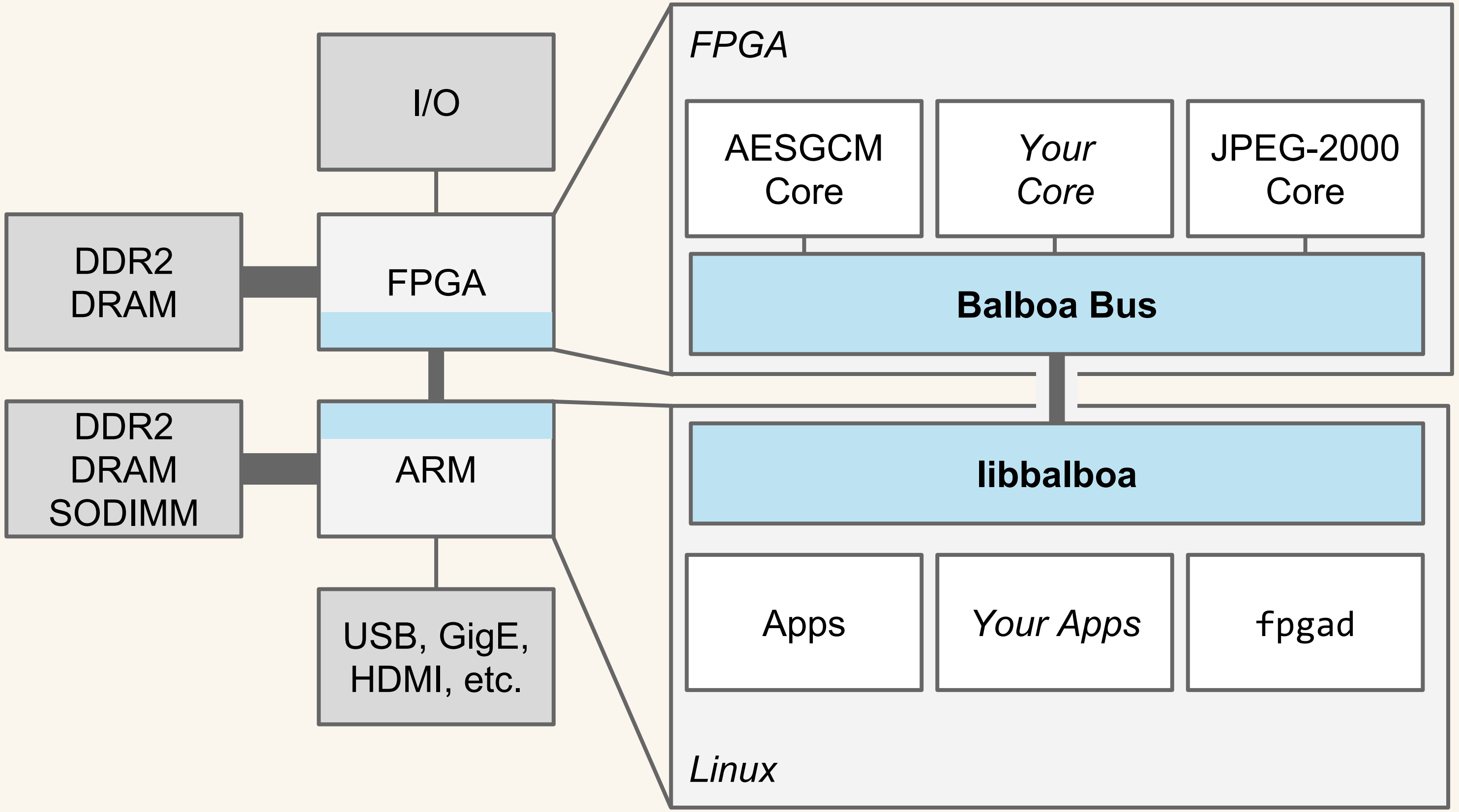
Balboa is:

- a library and some control software
  - makes sure management happens correctly
- a bus you can plug into on the FPGA, when you’re writing the core.

# Novena Hardware Platform



# Balboa Architecture



## Virtual Hardware and Space-sharing

FPGAs are huge, and cheap; Novena is perfect for experimenting!

## Balboa Vision: Where We’re Going

(Why having Balboa will be cool)

- Interoperability of accelerator cores
- Current tooling is not great
- FPGAs currently not widely used because of tool roadblocks
- FPGAs can be more useful

## Balboa Goals

- You write cores in your chosen HDL
- Fast, direct access to cores from app code
- Standard interfaces for cores and apps
- User chooses what runs, when

## Results

What we currently have:

- Multiple cores running on FPGA, one at a time
- Apps can **mmap()** their core
- No kernel driver (yet)

## What’s Next

## Seven Security Issues

A few challenges we’ll need to overcome

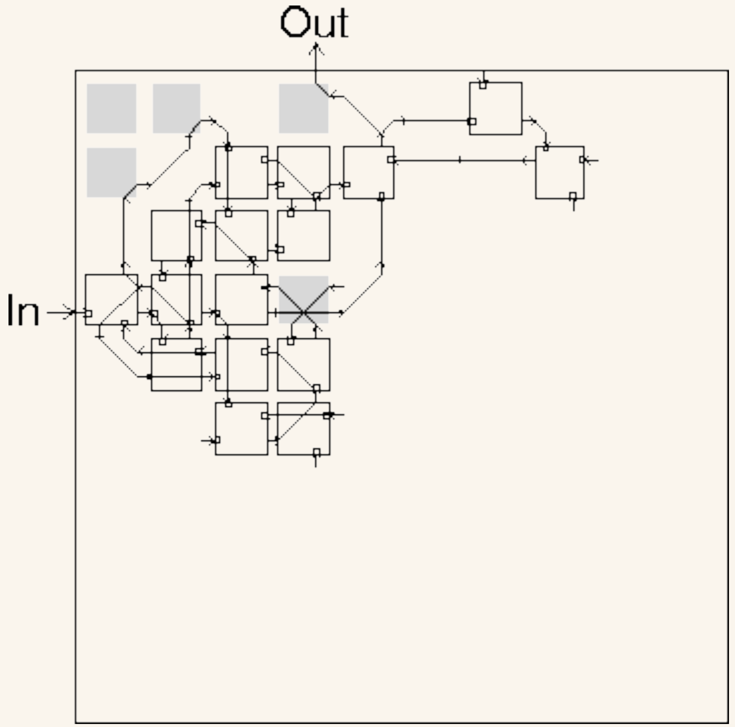
## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

“*An Evolved Circuit*”

Adrian Thompson, ICES96

http://bit.ly/evolved_circuit



(https://web.archive.org/web/20120302224543/http://www.sussex.ac.uk/Users/adrianth/ices96/paper.html)

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

Bitstream exploits

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

Bitstream exploits

Bus protocol

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

Bitstream exploits

Bus protocol

Malicious app

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

Bitstream exploits

Bus protocol

Malicious app

Timing attacks

## Seven Awesome Hacks ●●●●●●●

Electromagnetic coupling

Untrusted IO to bitstream

Bitstream exploits

Bus protocol

Malicious app

Timing attacks

Hardware backdoors

## Seven Next Projects

Here’s where we could use your help

## Seven Next Projects ●●●●●●●

Combine cores into one image

## Seven Next Projects ●●●●●●●

Combine cores into one image

Dynamically reconfigure FPGA slicewise

# Seven Next Projects ●●●●●●●

Combine cores into one image

Dynamically reconfigure FPGA slicewise

Inter-core bus arbitration

## Seven Next Projects ●●●●●●●

Combine cores into one image

Dynamically reconfigure FPGA slicewise

Inter-core bus arbitration

Multiple mmap()ed cores

## Seven Next Projects ●●●●●●●

Combine cores into one image

Dynamically reconfigure FPGA slicewise

Inter-core bus arbitration

Multiple mmap()ed cores

I/O

## Seven Next Projects ●●●●●●●

Combine cores into one image

Dynamically reconfigure FPGA slicewise

Inter-core bus arbitration

Multiple mmap()ed cores

I/O

FPGA/DRAM allocation

## Seven Next Projects •••••••

Combine cores into one image

Dynamically reconfigure FPGA slicewise

Inter-core bus arbitration

Multiple mmap()ed cores

I/O

FPGA/DRAM allocation

Higher level implementation support

## Seven Next Projects

Join us to find out more!

wiki: https://balboa.is

github: https://github.com/balboa-fpga/

twitter: #balboaFPGA

mailing list (coming soon)

# Thank you!

Star Simpson
@starsandrobots

Andy Isaacson
@eqe