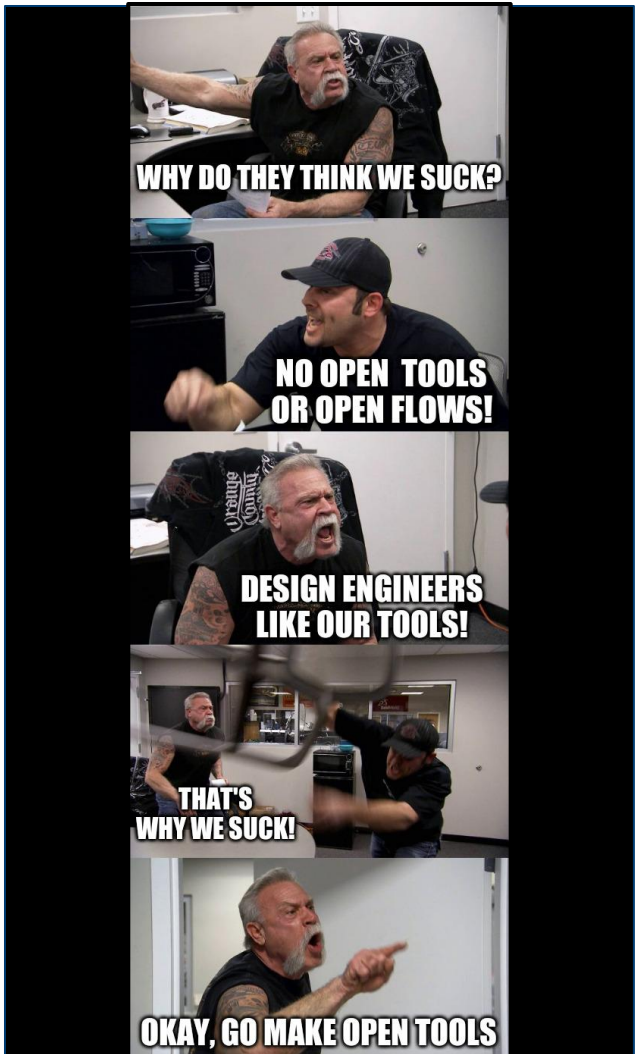
Everything Wrong With FPGAs

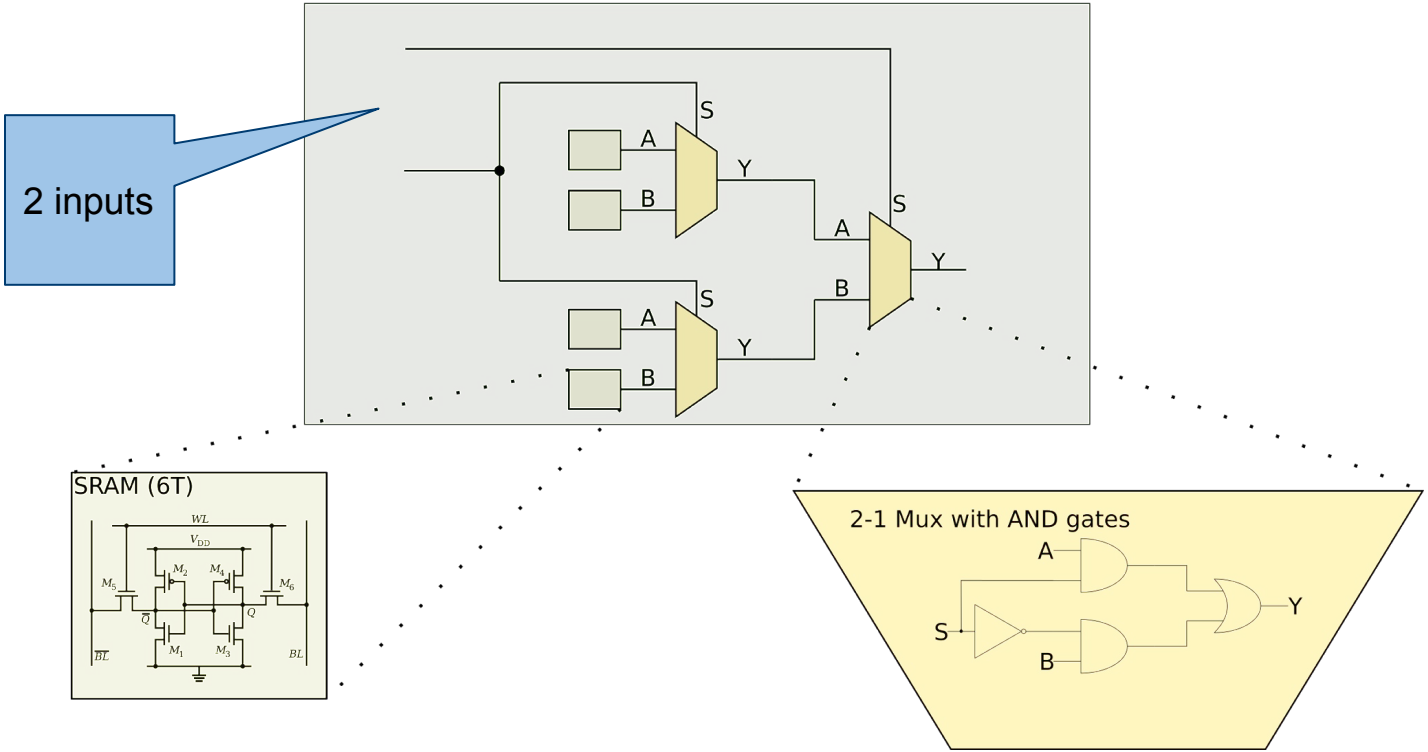
Ben Widawsky
ben@bwidawsk.net
@widawsky


WHY DO THEY THINK WE SUCK?
NO OPEN TOOLS
OR OPEN FLOWS!
DESIGN ENGINEERS
LIKE OUR TOOLS!
THAT'S
WHY WE SUCK!
OKAY, GO MAKE OPEN TOOLS

Agenda

1. What is an FPGA
2. Tools
3. How are FPGAs being used
4. Similarities to graphics / avoiding the pitfalls

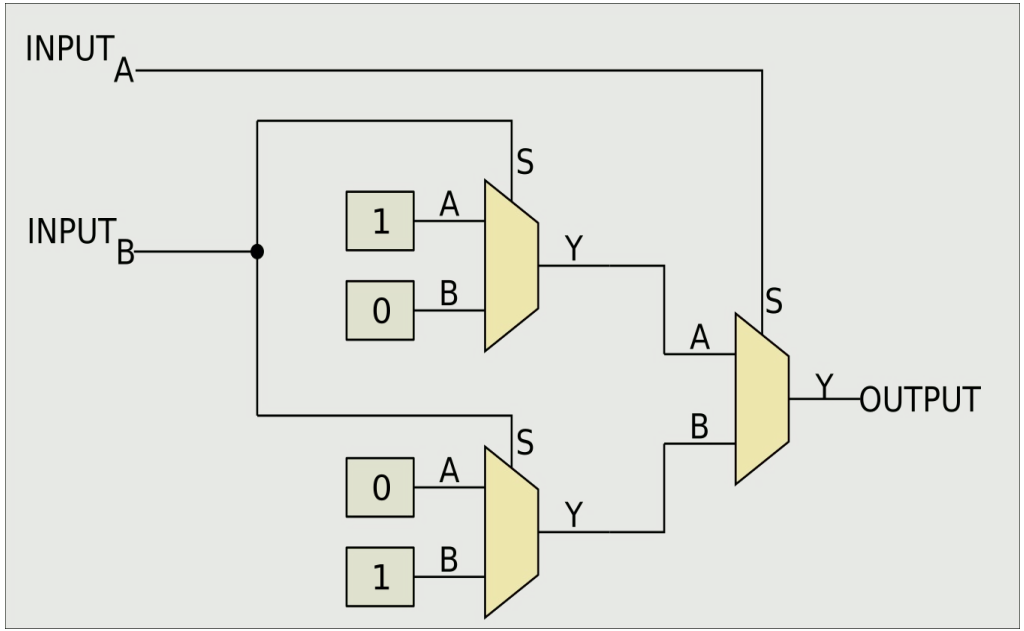
LUT2s


2 inputs
S
A
B
Y
S
A
B
Y
S
A
B
Y
SRAM (6T)
WL
V_{DD}
M_5
M_2
M_4
M_6
Q
$\overline{Q}$
$\overline{BL}$
M_1
M_3
BL
2-1 Mux with AND gates
A
S
B
Y

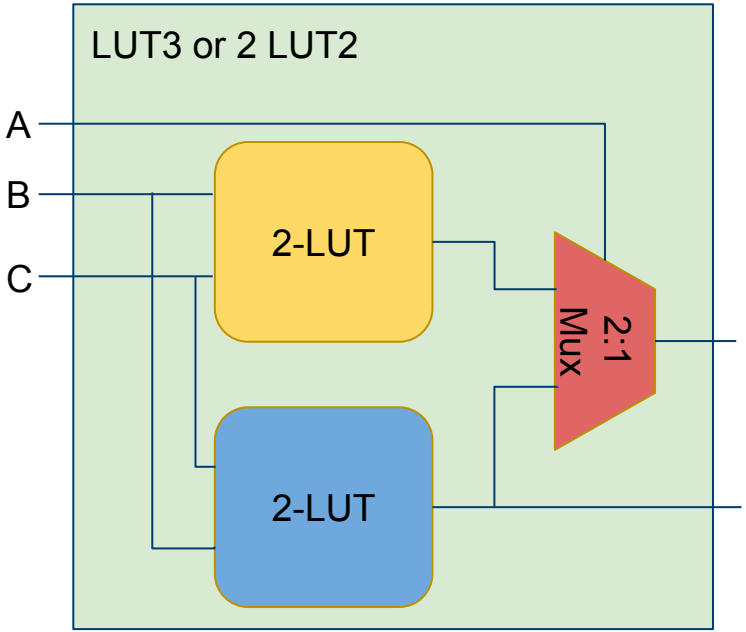
XNOR with LUT2

XNOR

$INPUT_A$	$INPUT_B$	OUTPUT
0	0	1
0	1	0
1	0	0
1	1	1



LUTs as Building Blocks

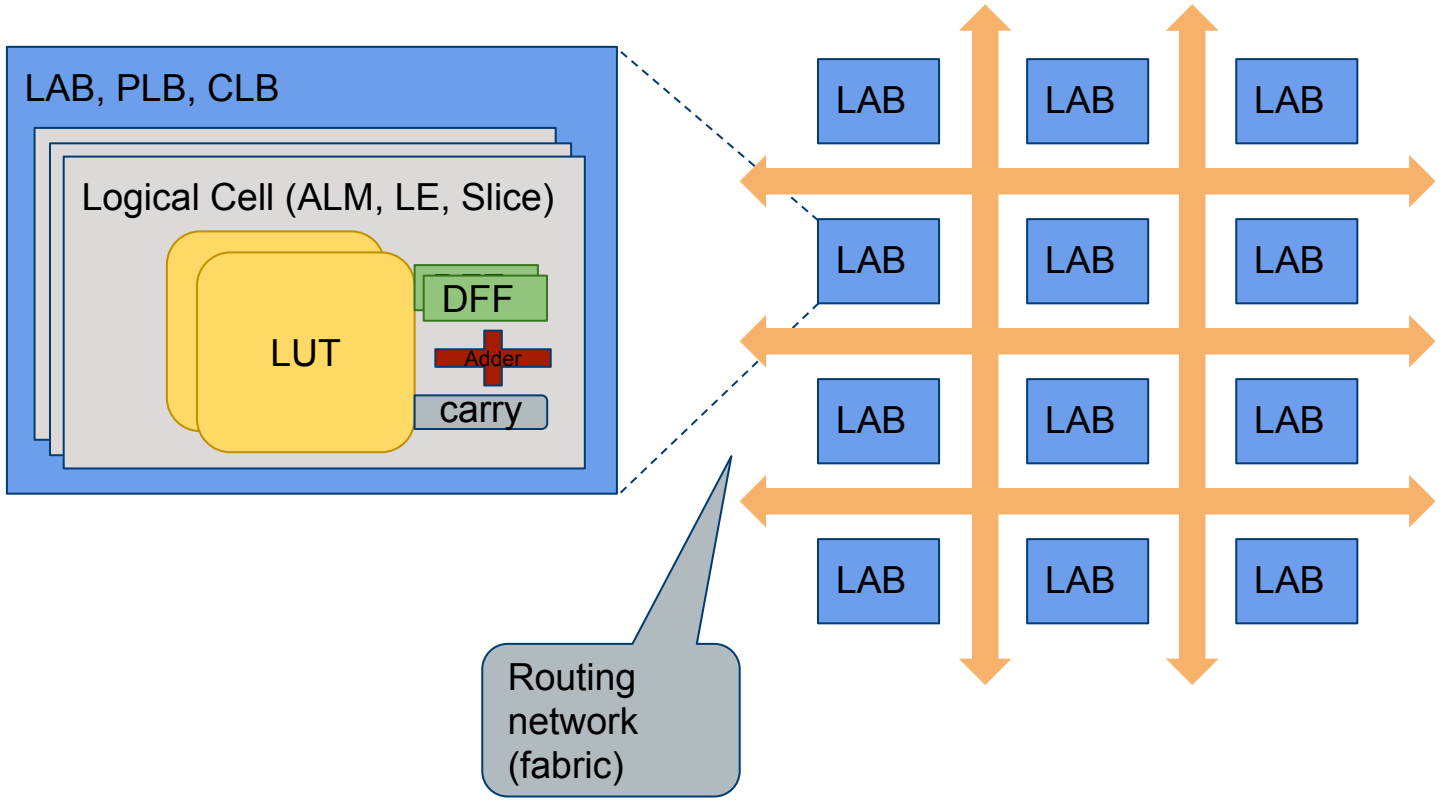


2 LUT3 + MUX = LUT4...

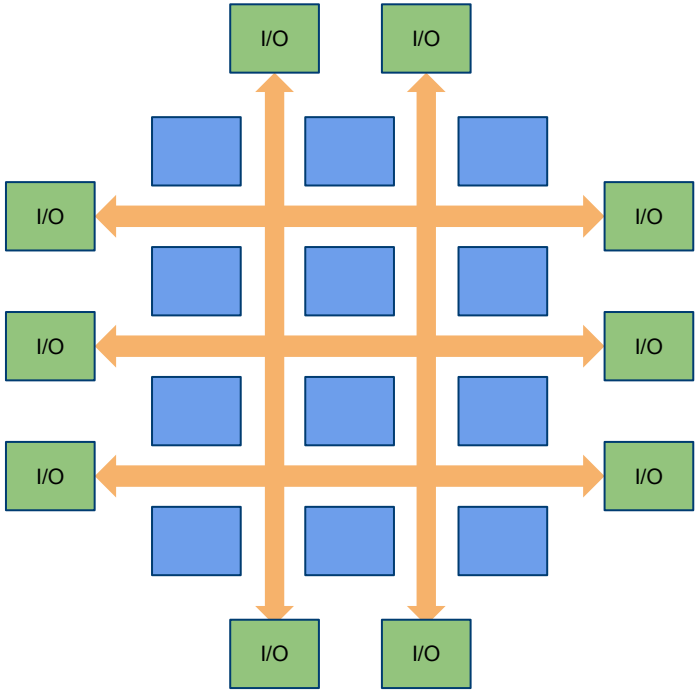
(A&B)|C SRAM

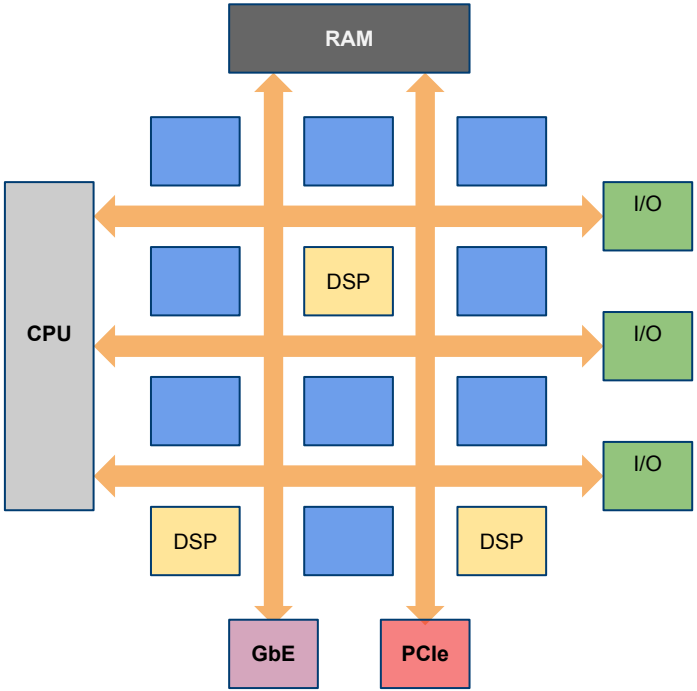
A	B	C	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Anatomy of an FPGA



But Wait, There's More!





What's Wrong With That?

- The programmable nature ultimately makes it less efficient than "hardwired" transistors.
- FPGAs differ enough to make it difficult to move between vendors
  - Unlike GPUs, standards are soft
    - Verilog/VHDL offer little and developers choose libraries over writing portable HDL
- FPGA hardware differs drastically
  - Standardize a description of FPGA floorplan

Tools

Current Landscape

- Vendor tools rule (synthesis, timing, place and route, bitstreams)
  - Quartus, Vivado, Diamond
    - Vivado's RapidWright is moving toward opening flow, allowing 3rd party place and route (Yosys supports this experimentally)
- 3rd party EDA tools exist (synthesis and timing)
  - Cadence, Mentor, Synopsys
- Open Source tools are in their infancy
  - Yosys/nextpnr, icarus
- Kernel interfaces is usable as of 5.2...current
  - DFL - Device Feature List
  - OPAE - Open Programmable Acceleration Engine
  - Per vendors interfaces ~20 of these

FPGA Tooling Flow

HDL -> Netlist -> Place & Route -> Bitstream -> Download

NP Complete
(circuit minimization, technology mapping)

NP Hard
(Optimal solution with finite time, acceptable solution in finite time is NP Complete)

Trivial

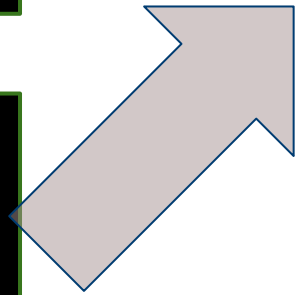
```
always @(posedge clk)
    if( RDY ) begin
        AI7 <= AI[7];
        BI7 <= temp_BI[7];
        OUT <= temp[7:0];
        CO  <= temp[8] | CO9;
        N   <= temp[7];
        HC  <= temp_HC;
    end

assign V = AI7 ^ BI7 ^ CO ^ N;
assign Z = ~|OUT;
```

```
11101001011100010101010100011110100 1
```

Synthesis Example (AND - LUT2)

```
bwidawsk@lundgren:~/work/fpga/projects/simple cat and.v
module AND (input x,
    input y,
    output reg a
);
    assign a = x & y;
endmodule
```

```
attribute \src
"/home/bwidawsk/work/fpga/projects/simple/and.v:1"
module \AND
  attribute \src
"/home/bwidawsk/work/fpga/projects/simple/and.v:3"
  wire output 3 \a
  attribute \src
"/home/bwidawsk/work/fpga/projects/simple/and.v:1"
  wire input 1 \x
  attribute \src
"/home/bwidawsk/work/fpga/projects/simple/and.v:2"
  wire input 2 \y
  cell $lut $abc$93$auto$blifparse.cc:492:parse_blif$94
    parameter \LUT 4'1000
    parameter \WIDTH 2
    connect \A { \x \y }
    connect \Y \a
  end
end
```



```
{
  "modules": {
    "AND": {
      "attributes": {
        "src": "/home/bwidawsk/work/fpga/projects/simple/and.v:1"
      },
      "ports": {
        "x": {
          "direction": "input",
          "bits": [ 2 ]
        },
        "y": {
          "direction": "input",
          "bits": [ 3 ]
        },
        "a": {
          "direction": "output",
          "bits": [ 4 ]
        }
      },
      "cells": {
        "$abc$93$auto$blifparse.cc:492:parse_blif$94": {
          "hide_name": 1,
          "type": "$lut",
          "parameters": {
            "LUT": 8,
            "WIDTH": 2
          },
          "attributes": {
          },
          "port_directions": {
            "A": "input",
            "Y": "output"
          },
...
```

Technology Mapping (A&B)|C - LUT3

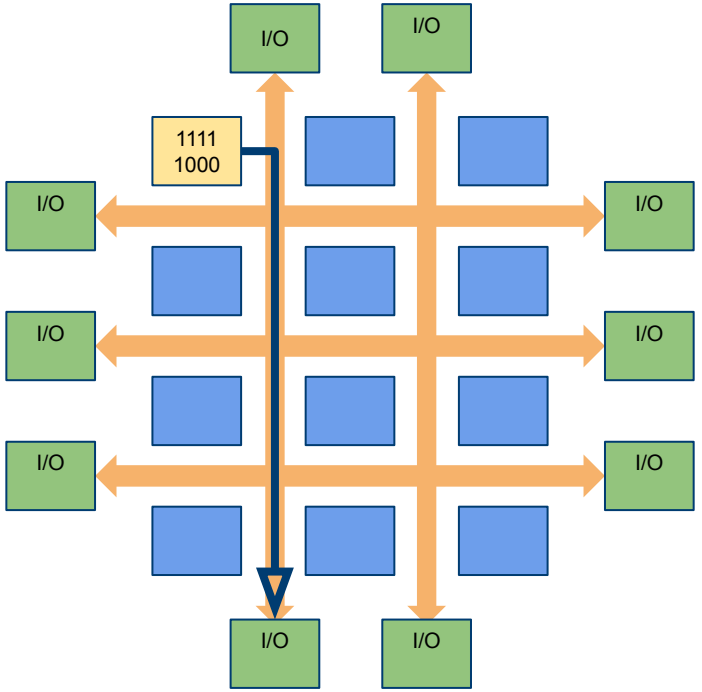
```
module MORE (input a,
             input b,
             input c,
             output reg o);
             assign o = (a&b)|c;
 endmodule
```

```
cell $lut $abc$49$auto$blifparse.cc:492:parse_blif$50
             parameter \LUT 8'11111000
             parameter \WIDTH 3
             connect \A { \c \a \b }
             connect \Y \o
end
```

~~AC~~	~~BA~~	~~CB~~	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Place and Route

- Place
  - Assign to specific locations on the chip
    - Minimize distance
    - Meet timing constraints
    - Make it fit
  - Utilize hard IP blocks
- Route
    - Find a route to connect nets
    - Reduce delay in critical nets
- Trade Offs
  - Area
  - Performance
  - Power

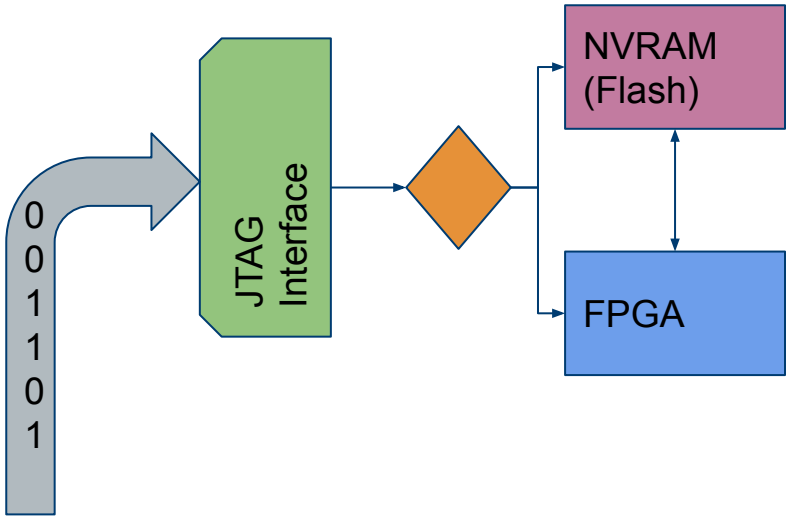


Bitstream Assembly

- Create the binary data
  - represents the LUT programming (synthesis results)
  - in the right locations (place results)
  - with the right connections (route results).
- Bitstream may be compressed, encrypted, signed, or all 3
  - Decompression may be done by first uploading logic to decompress

Programming

- Puts the bits on the card
- JTAG based
  - OpenOCD
  - Proprietary
- Volatile/non-volatile
- Partial Reconfiguration

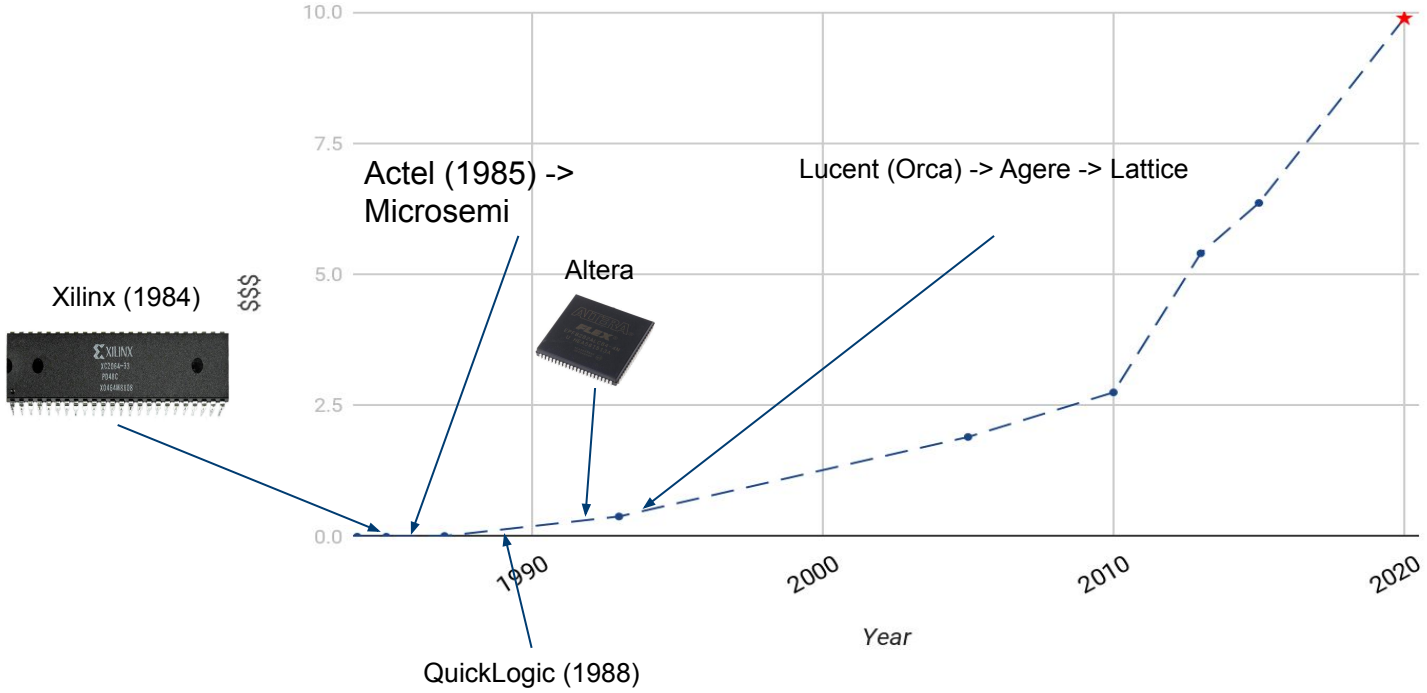


What's Wrong With That?

- Tools are proprietary and drastically differ by IHV
  - Can take multiple days to get up and running
- Tools require cumbersome license servers that are hard to use
  - Binary blobs using 'who knows what' compromised software
- Rely on the customer support of IHV
- Cost of entry is high enough that much IP generation is outsourced
  - End users often "connect boxes" only
- Customers are unable to improve or debug the tool flow
- Scripting requires TCL interfaces that aren't well documented


How Are They Used?

Market Size



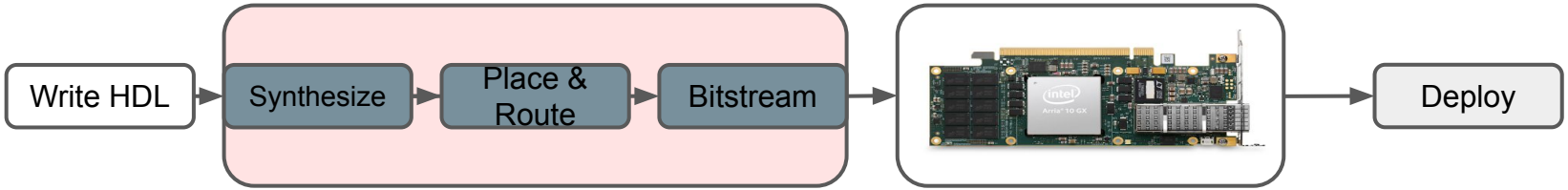
Solution without a problem

Traditional Vertical FPGA

- Description
  - Burn a design onto an FPGA for shipping, or as a POC
    - Write or buy all RTL for the device
- Reasoning
  - Shipping in low volume (ASIC economies of scale win at ~3 million units)
  - Implementing volatile specs
  - Can’t wait for ASIC turnaround
  - Simulate ASIC before RTL freeze
    - Much faster than soft simulators

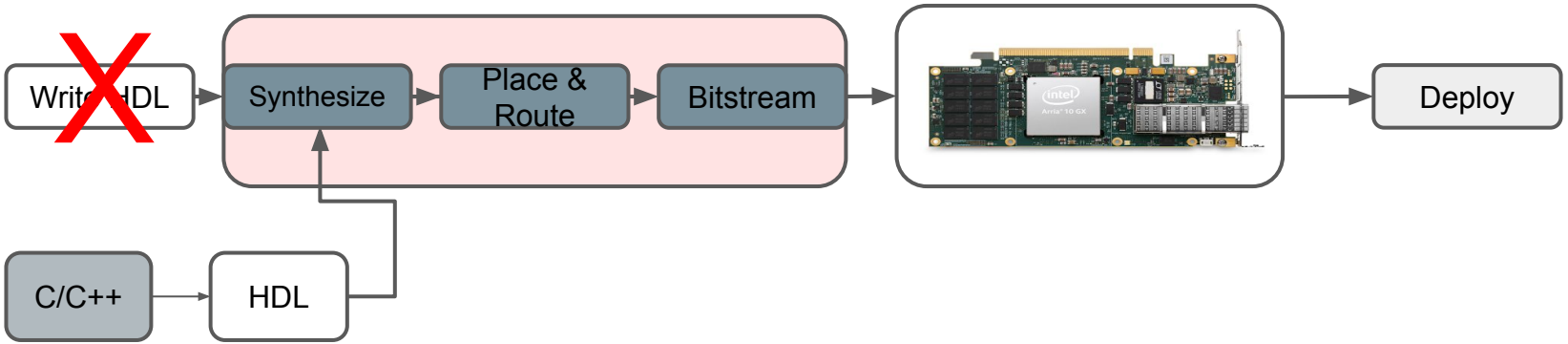
Traditional FPGA “Flow”

- FPGA is sitting “on a desk” during development
- Develop entire “stack”
  - RTL implementation, asset integration, and infrastructure
- At production, FPGA image is burned onto Flash that is fetched on boot



High Level Synthesis

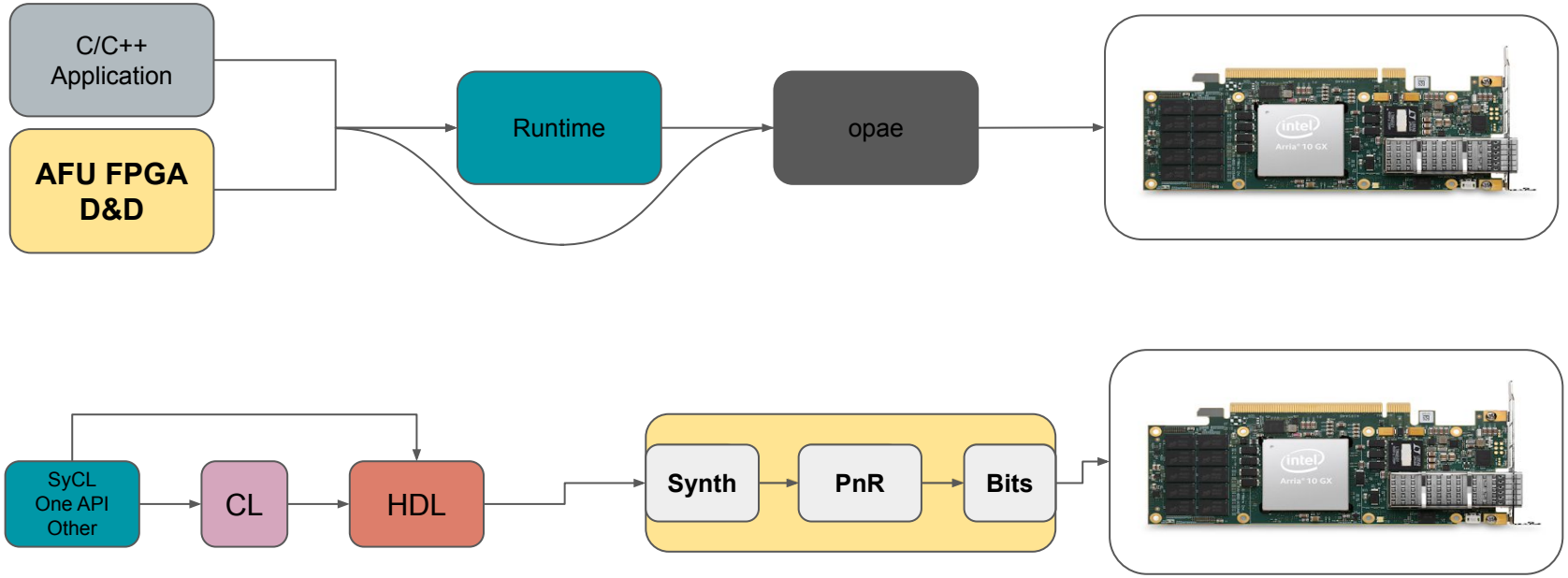
- Description
  - Enable high level language to make synthesizable HDL
    - C/C++ -> bitstream
  - Otherwise, same as vertical flow
- Reasoning
  - Many more software engineers than hardware engineers



FPGA As An Accelerator (FPGAAAA!)

- Description
  - Some FPGA somewhere runs your stuff fast
- Interfaces
  - OpenStack Cyborg (OPAE)
  - OpenVINO (DLA)
  - Kubernetes device plugin (OPAE)
- Raw Access
  - Azure
  - AWS

FPGAAAA “Flow”

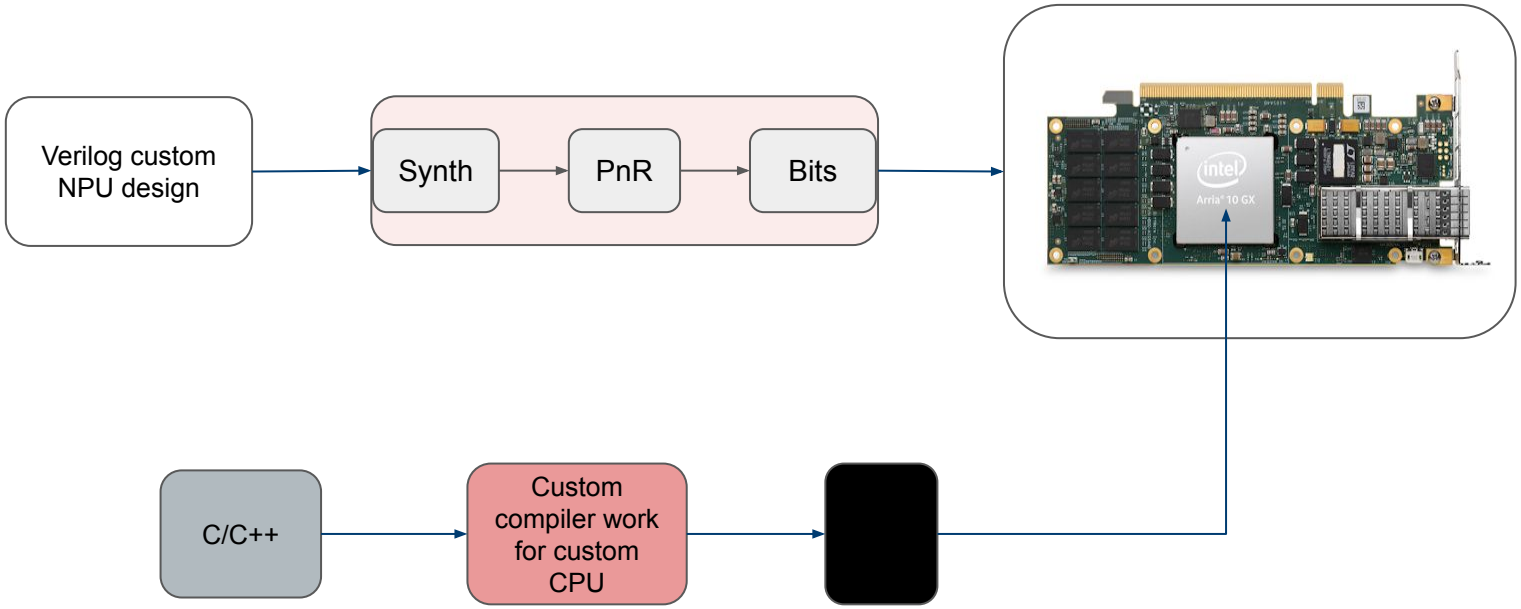


FPGA As An AI Accelerator (FPGAAAAA!!)

Microsoft's Project Brainwave claims win over Google TPU [*]

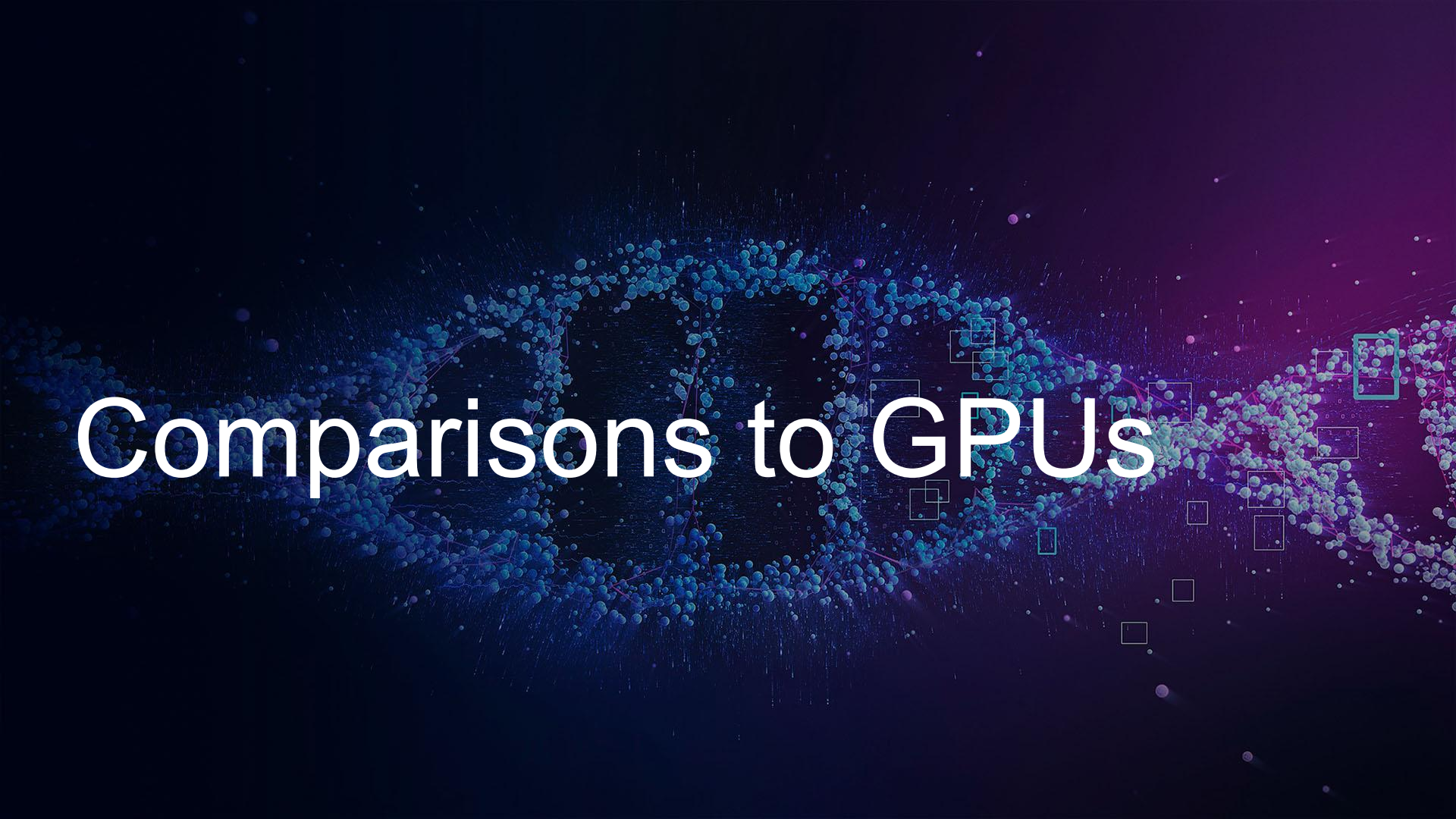
"Project Brainwave achieves more than an order of magnitude improvement in latency and throughput over state-of-the-art graphics processing units (GPUs) on large recurrent neural networks (RNNs), with no batching"

FPGAAAAA Flow



What's Wrong With That?

- Relatively small market
- Vertical FPGA developers learned a tool and don't want to switch
- Many accelerator flows actually go to OpenCL first
  - Can lead to less optimal designs
- OPAE is open and great, but…
  - Depends on proprietary FIU
  - AFUs are also usually proprietary
- FPGAAAAA is very resource intensive
- HLS hasn't really taken off
- Compilers did the proprietary thing... they lost
  - Intel, ARM as examples still sell compilers, but developers prefer GCC/LLVM


Comparisons to GPUs

Similarities

- Dense set of transistors providing more power than a CPU
  - New markets are growing fast and standards follow slowly
- Very limited set of vendors - Xilinx, Intera
- Lots of proprietary software.
- Lack of good documentation.
  - ALL supported hardware is reverse engineered
- Binary blob drivers
  - Currently, almost everything uploaded to an FPGA is proprietary
- Vibrant reverse engineering community
- DFL is DRM/GEM/KMS, OPAE is libdrm

Dissimilarities

- No strict standard APIs
  - Lack of conformance tests
  - Yosys and nextpnr should be the Mesa of FPGAs
  - Need a piglit
- Kernel interfaces in infancy
  - Consolidated interfaces driven by Intel
  - Provides a standardized way to upload non-standard functionality
    - ie. Should a gzip accelerator differ between Xilinx/Altera?
  - Lots of emphasis on opening the interface to the device and no emphasis on what's running on the device
- Limited interest in opening
  - There's no "whale" asking for this

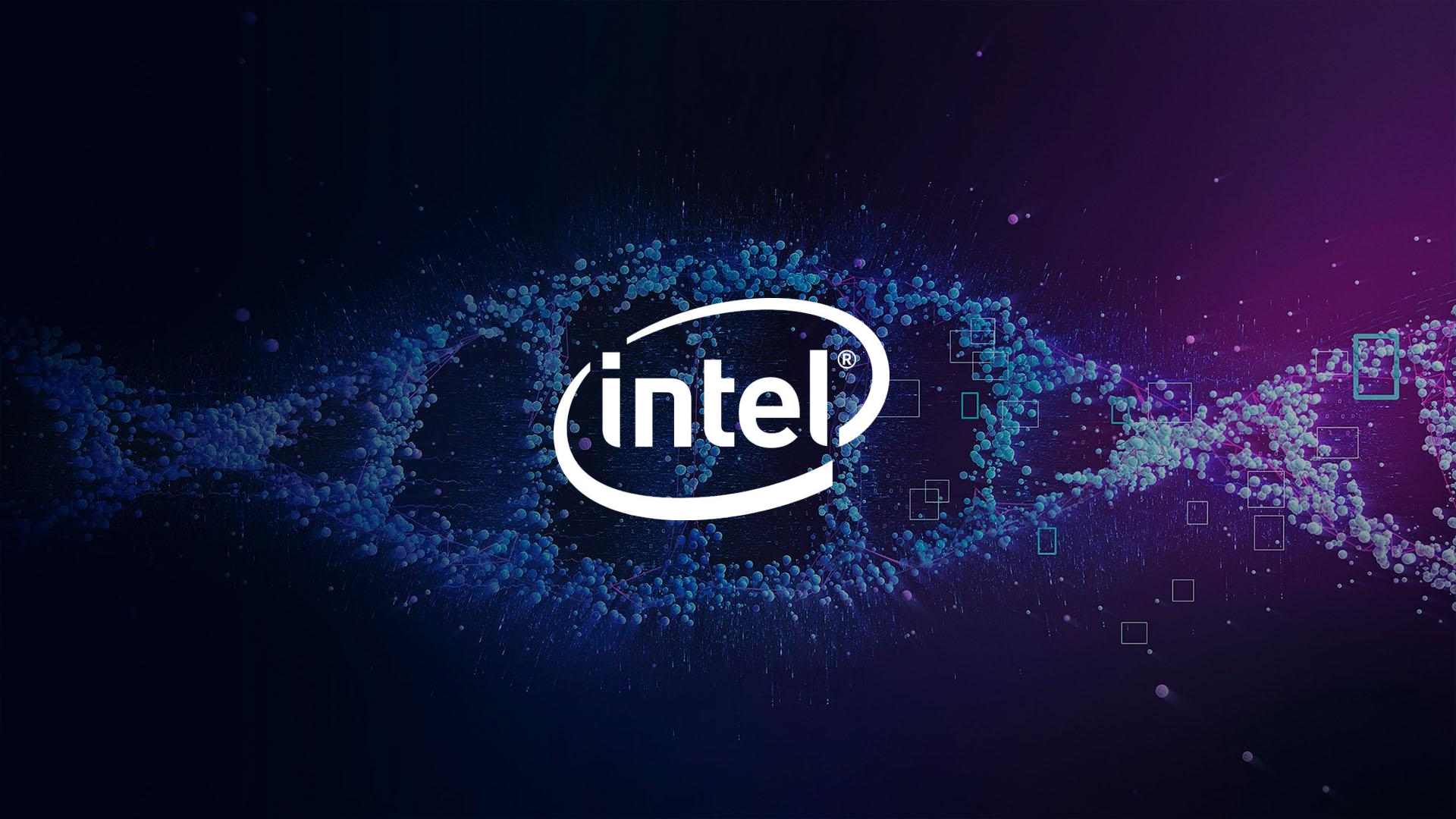
Learning From Mistakes of Graphics

General problem in FPGA are the same as graphics

1. Enforce an open userspace
2. Enforce unit testing early
3. Don’t allow interfaces to be merged supporting one type of hardware
4. Use existing kernel paradigms for datasets input/output
5. Make virtualization use case first class citizen
6. Customers need to demand open documentation
7. Don’t assume Windows is the dominant OS
8. Don’t assume open source can’t solve difficult engineering problems

Call to action

- Improve Yosys
  - Use and file bugs - make a GPU!!
    - ECP5 and ice40 work well
  - Look into utilizing Yosys for OpenCL
  - Feasibility on using yosys to synthesize AFUs
- Improve PNR
- Reverse engineering
  - Many efforts: prjtrellis, prjxray, prjmistral
  - Help add support for other platforms
- Work on standards
  - Bitstreams, floorplans
- Review the mailing list for DFL - linux-fpga@vger.kernel.org


intel®