OSHUG - Open FPGA Toolchains

From Past to Present

David Shah
@fpga_dave
Symbiotic EDA

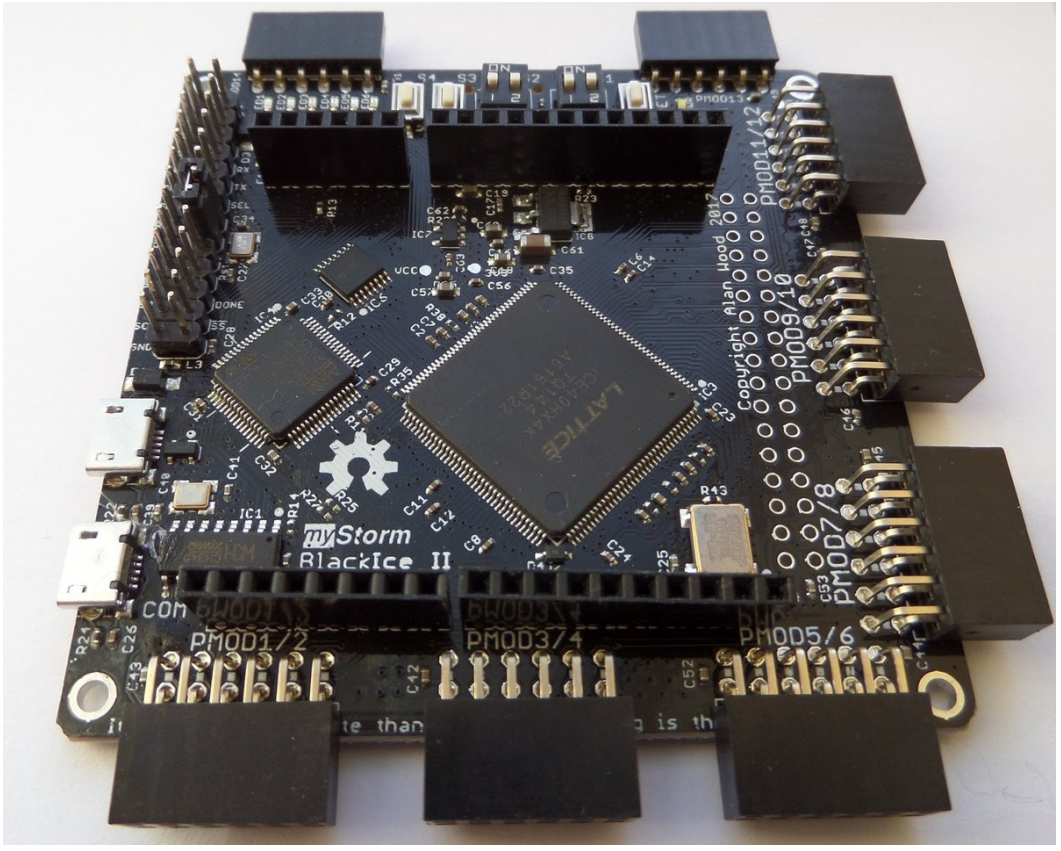
Slides: fpga.dev/oshug.pdf

Symbiotic EDA

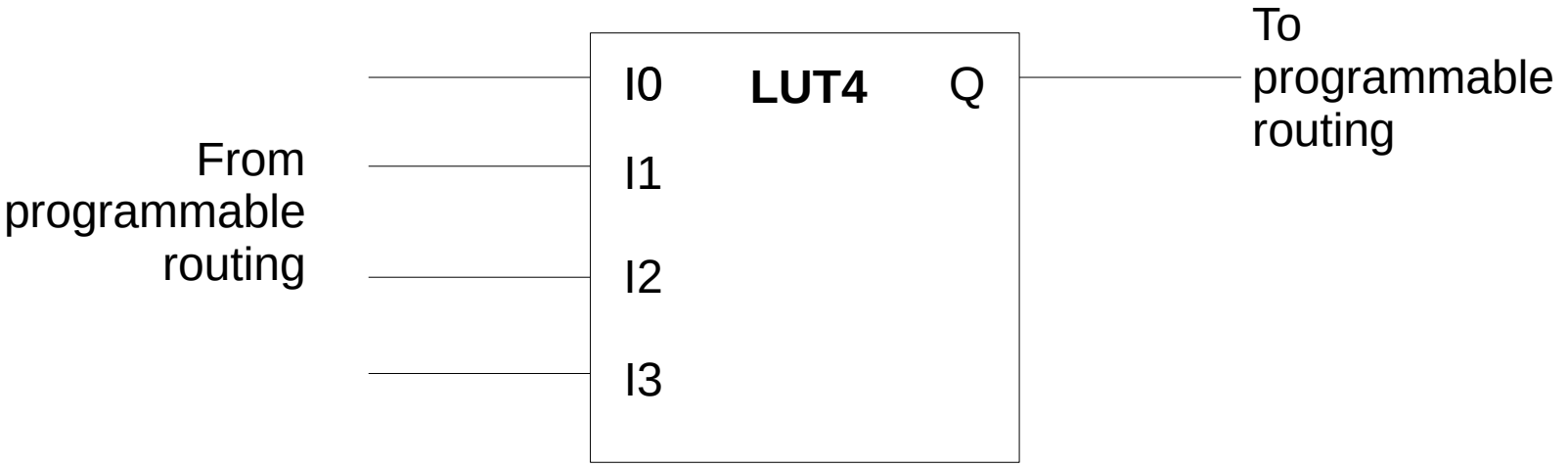
FPGA?

- Field Programmable Gate Array
- Chip containing **user-programmable** digital logic
- Can be (re)programmed “in the field” - older devices programmed once by literally burning fuses

FPGA?


myStorm
BlackIce II
PMOD1/2
PMOD3/4
PMOD5/6
PMOD7/8
PMOD9/10
PMOD11/12
Copyright Alan Wood 2017
LATTICE

FPGA Logic



Look Up Table (LUT) – basic logic element of an FPGA

a K-input LUT has 2^K bits of memory storing its **function**
e.g. 4-input AND: `1000 0000 0000 0000`
4-input OR: `1111 1111 1111 1110`

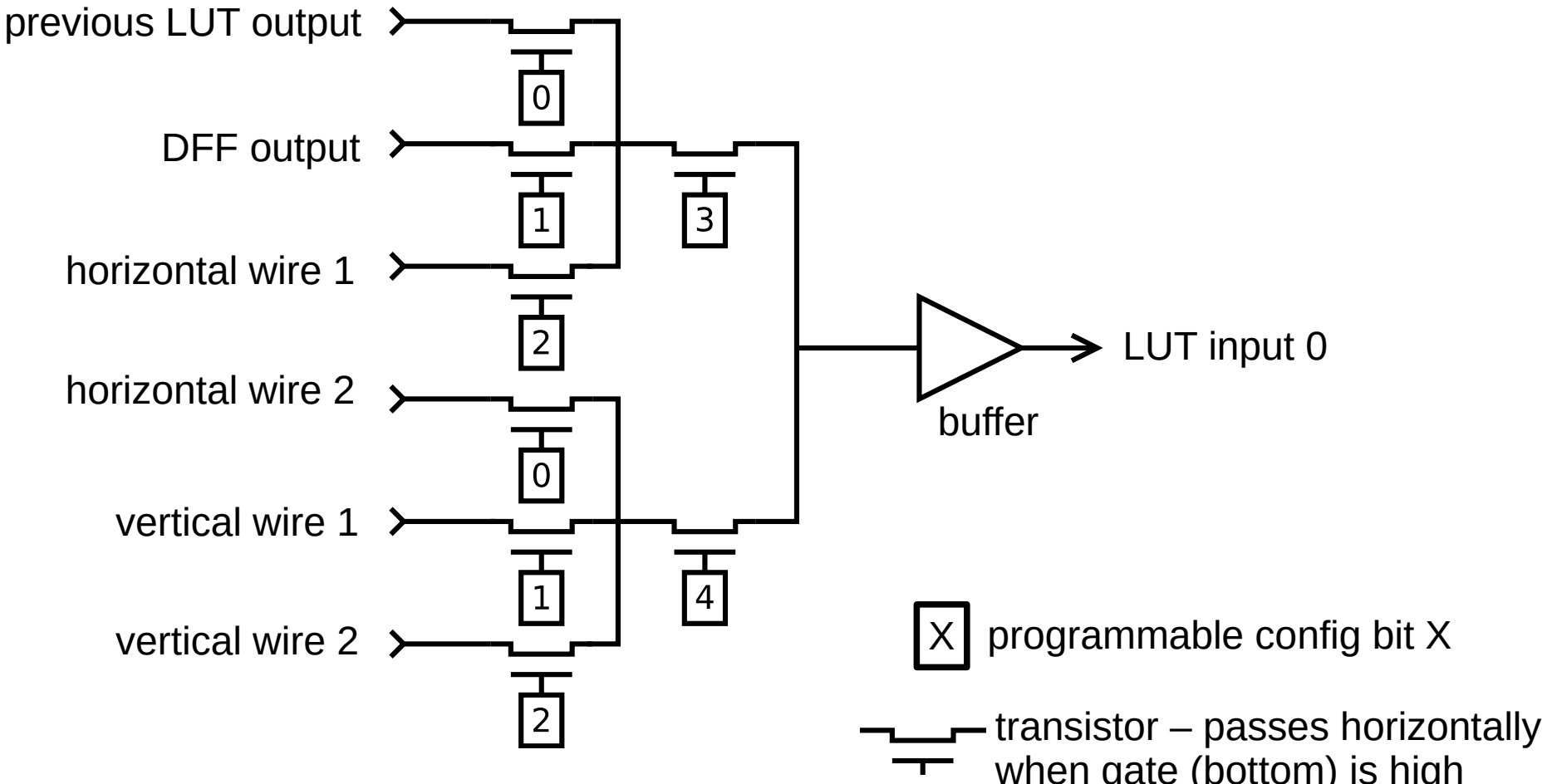
FPGA Logic

D-type flip flop (DFF) – basic storage element of an FPGA

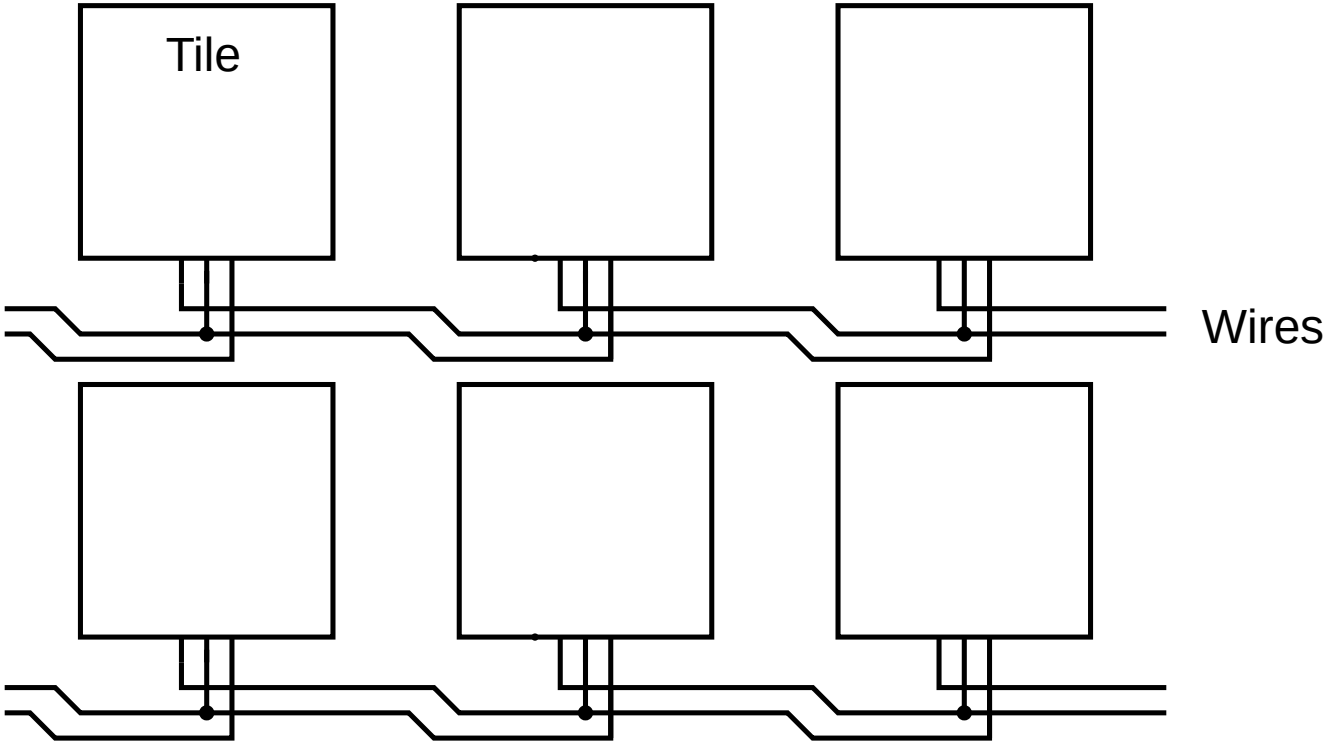
D is stored at the rising or falling edge of **CLK**

Combine DFFs and LUTs to build sequential logic such as state machines, counters, CPUs, ...

FPGA Routing

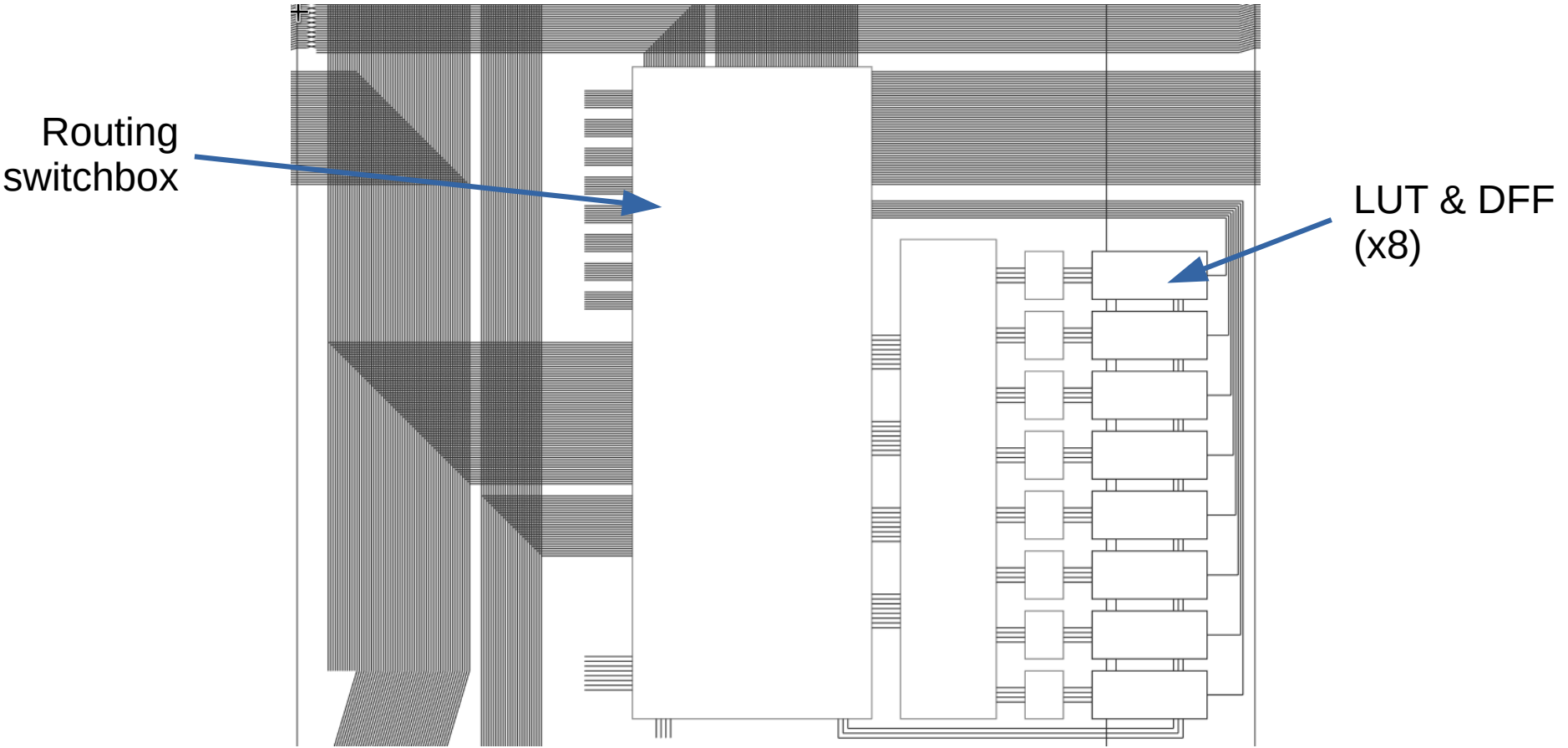


FPGA Structure



(vertical interconnect not shown for clarity)

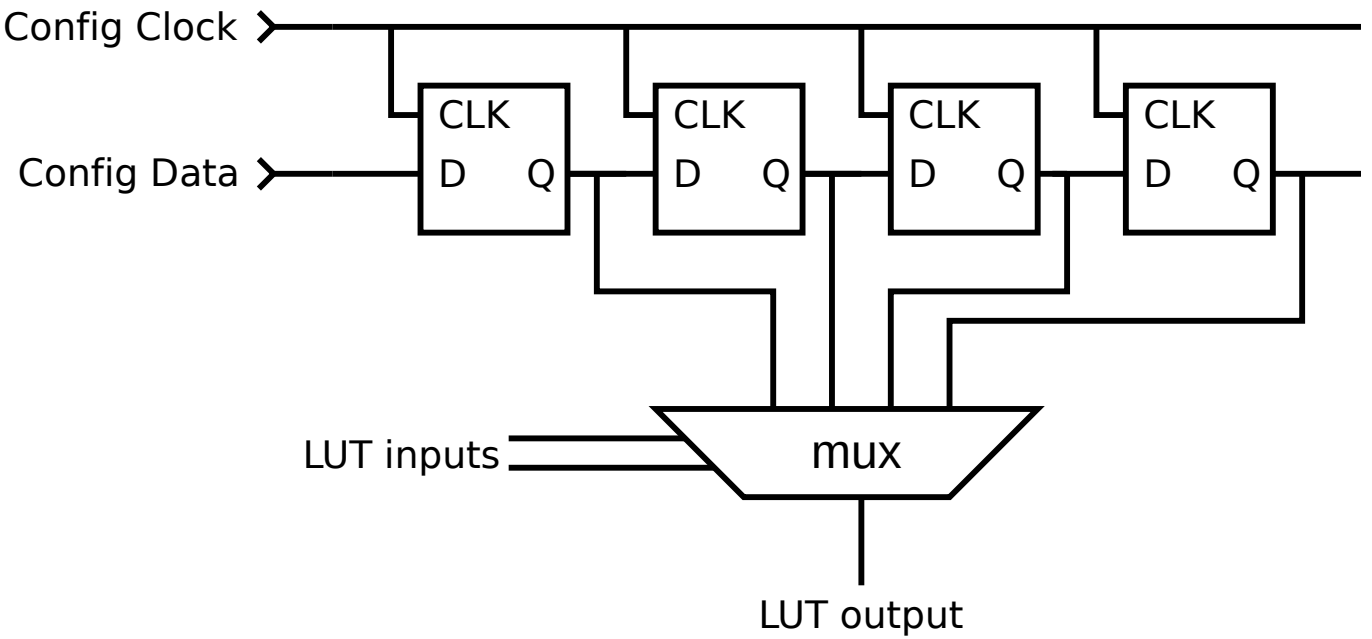
FPGA Structure



FPGA Structure

- Modern FPGAs aren’t just LUTs and DFFs!
- Block RAM: dual port SRAM, usually 4-36 kbit
- DSP: multiplier and adders
- High speed IO blocks for DDR memory, HDMI, PCI Express, etc
- Even CPUs! (Zynq ARM cores, Virtex-II Pro PPC)

FPGA Configuration



2-input FPGA LUT, showing configuration shift register

FPGA Configuration

- Logic and routing in a modern FPGA uses shift register type structures for configurability
- The programming file for an FPGA, called a **bitstream**, loads all of these shift registers
- Typical bitstreams comprised of commands, for both loading configuration and other tasks such as initialising block RAM

FPGA Configuration

- No (public) documents telling you what the bits inside a bitstream actually do!
- Unlike most other kinds of programmable chip – microcontrollers usually have register guides, etc
- Expectation is to use vendor-provided proprietary design software

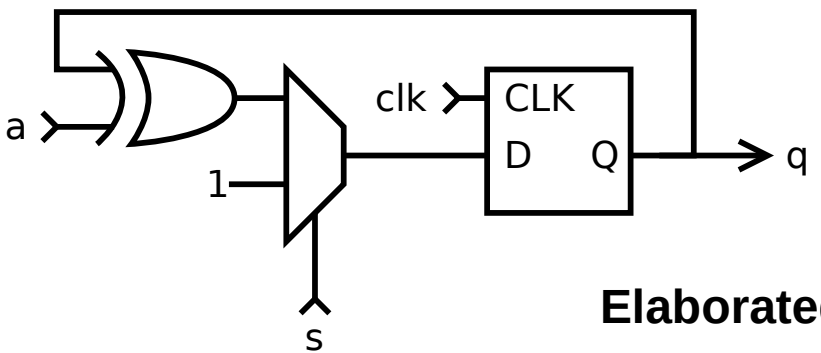
FPGA Toolchains

- FPGA designs are written in a hardwareD description language (HDL), usually Verilog or VHDL
- Also higher level frameworks such as Migen/LiteX (Python based), SpinalHDL (Scala based) – still generate Verilog though

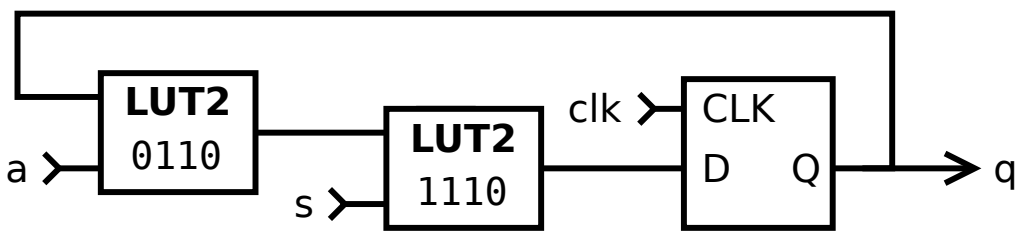
Synthesis

```
module top(
    input clk,
    input a, s,
    output reg q
);

always @(posedge clk)
    if (s)
        q <= 1;
    else
        q <= q ^ a;

endmodule
```

Verilog Source

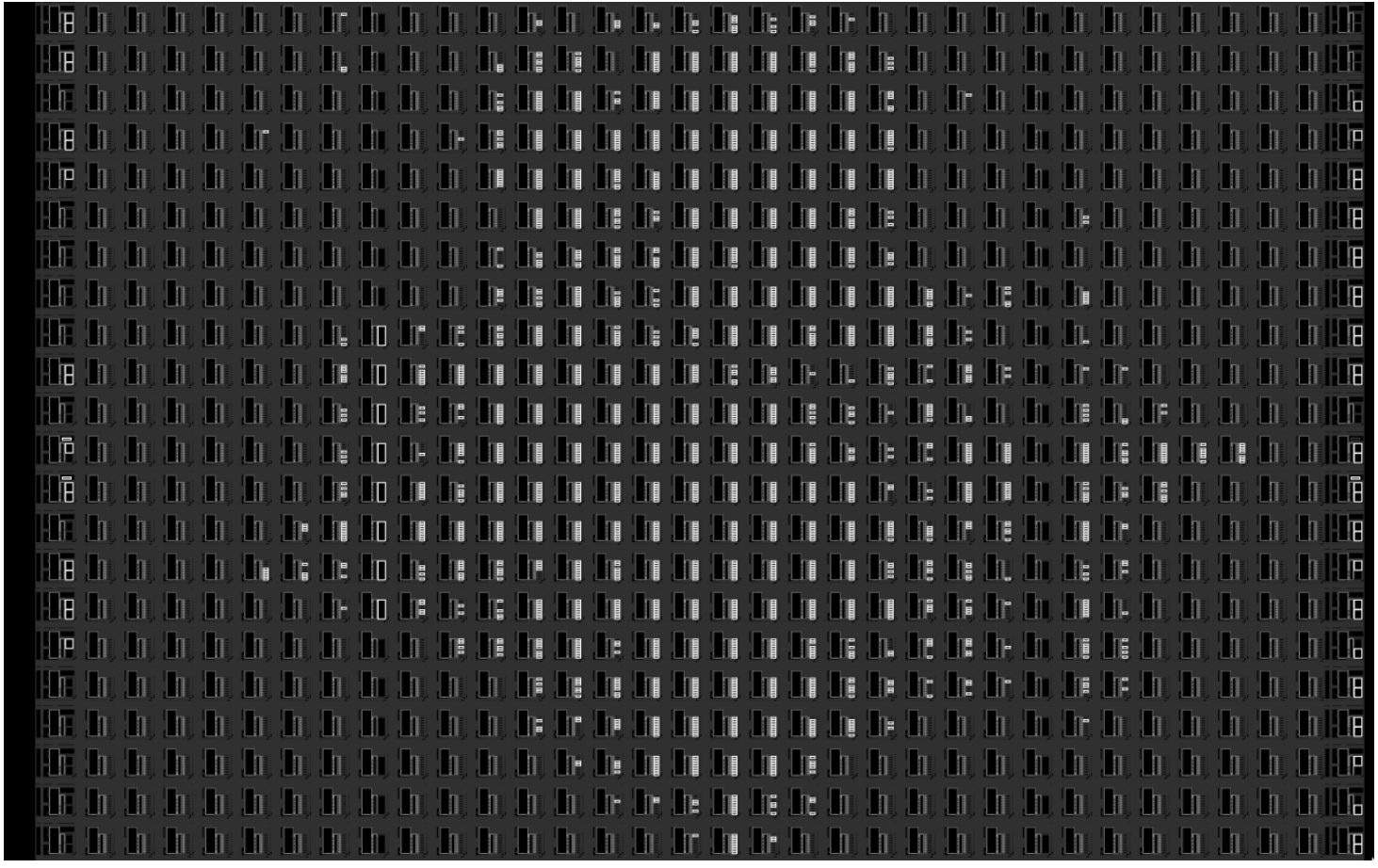


Elaborated Circuit

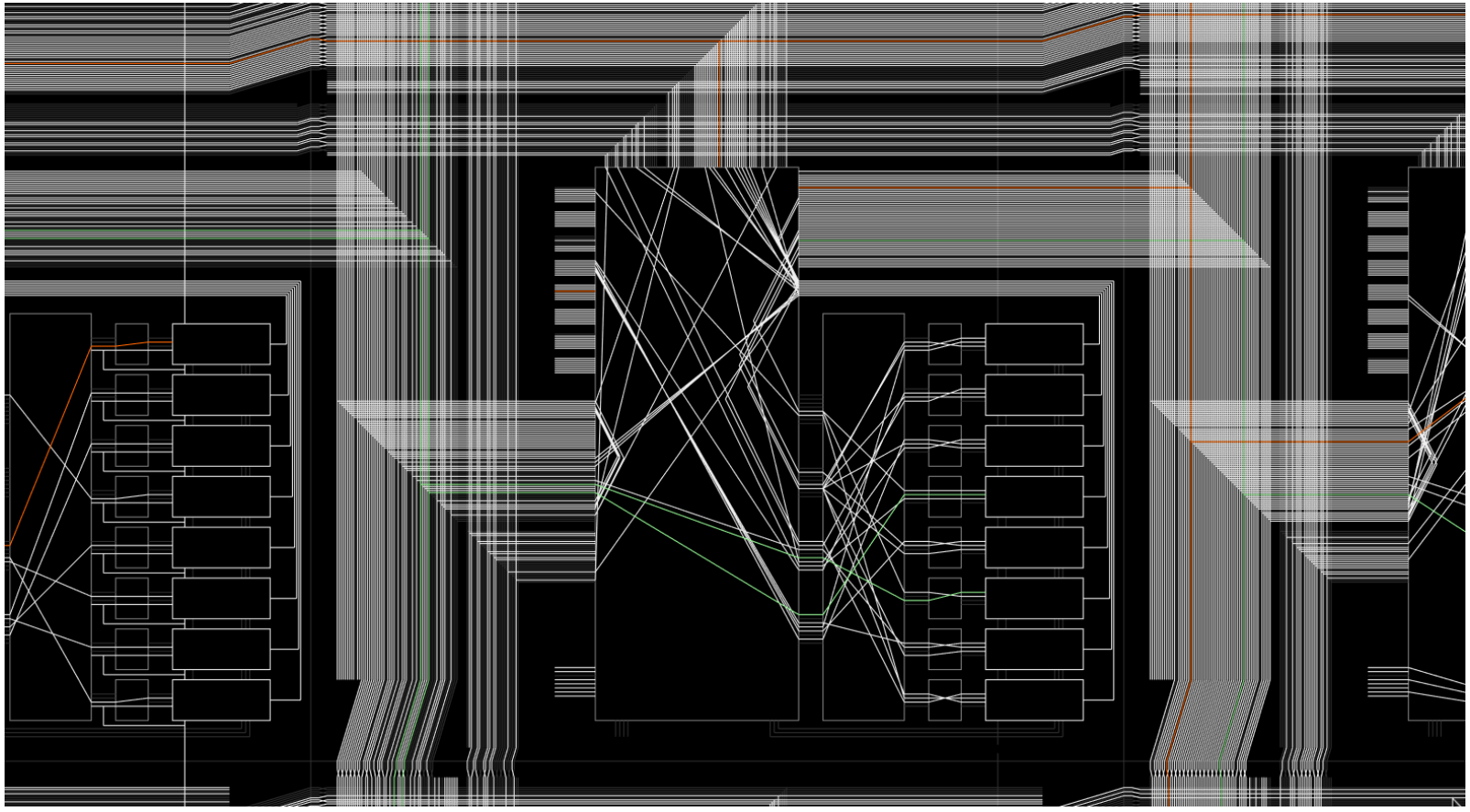


Mapped to 2-input LUTs

Placement



Routing



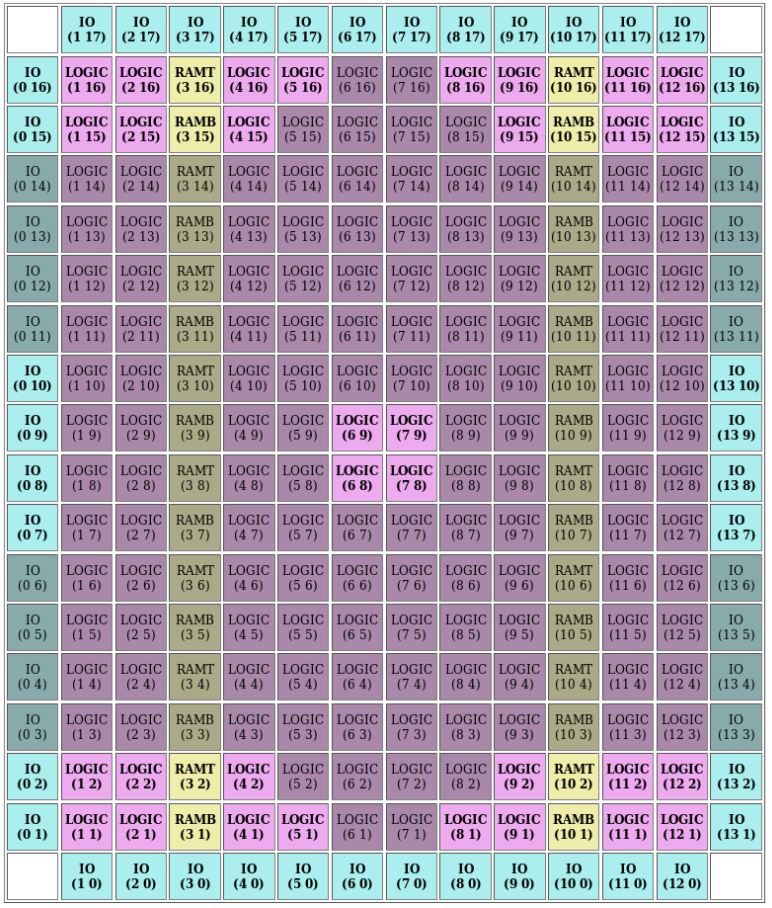
Open FPGA Flows

- Need to document bitstreams to build open source tools
- **Project Icestorm**: bitstream documentation for iCE40 FPGAs by Clifford Wolf & Matthias Lasser
- Combined with Yosys & arachne-pnr to build the first useful open FPGA flow
- arachne-pnr later replaced by nextpnr

iCE40 FPGAs

- 384 – 8k logic cells
- 1 logic cell: 4-input LUT, flipflop, carry logic for adders
- Up to 128kbit block RAM in 4kbit blocks
- 8 global networks to distribute clocks, resets and clock enables to all tiles
- Ultra and UltraPlus parts have some extra features

iCE40 FPGAs


IO (1 17) IO (2 17) IO (3 17) IO (4 17) IO (5 17) IO (6 17) IO (7 17) IO (8 17) IO (9 17) IO (10 17) IO (11 17) IO (12 17)
IO (0 16) LOGIC (1 16) LOGIC (2 16) RAMT (3 16) LOGIC (4 16) LOGIC (5 16) LOGIC (6 16) LOGIC (7 16) LOGIC (8 16) LOGIC (9 16) RAMT (10 16) LOGIC (11 16) LOGIC (12 16) IO (13 16)
IO (0 15) LOGIC (1 15) LOGIC (2 15) RAMB (3 15) LOGIC (4 15) LOGIC (5 15) LOGIC (6 15) LOGIC (7 15) LOGIC (8 15) LOGIC (9 15) RAMB (10 15) LOGIC (11 15) LOGIC (12 15) IO (13 15)
IO (0 14) LOGIC (1 14) LOGIC (2 14) RAMT (3 14) LOGIC (4 14) LOGIC (5 14) LOGIC (6 14) LOGIC (7 14) LOGIC (8 14) LOGIC (9 14) RAMT (10 14) LOGIC (11 14) LOGIC (12 14) IO (13 14)
IO (0 13) LOGIC (1 13) LOGIC (2 13) RAMB (3 13) LOGIC (4 13) LOGIC (5 13) LOGIC (6 13) LOGIC (7 13) LOGIC (8 13) LOGIC (9 13) RAMB (10 13) LOGIC (11 13) LOGIC (12 13) IO (13 13)
IO (0 12) LOGIC (1 12) LOGIC (2 12) RAMT (3 12) LOGIC (4 12) LOGIC (5 12) LOGIC (6 12) LOGIC (7 12) LOGIC (8 12) LOGIC (9 12) RAMT (10 12) LOGIC (11 12) LOGIC (12 12) IO (13 12)
IO (0 11) LOGIC (1 11) LOGIC (2 11) RAMB (3 11) LOGIC (4 11) LOGIC (5 11) LOGIC (6 11) LOGIC (7 11) LOGIC (8 11) LOGIC (9 11) RAMB (10 11) LOGIC (11 11) LOGIC (12 11) IO (13 11)
IO (0 10) LOGIC (1 10) LOGIC (2 10) RAMT (3 10) LOGIC (4 10) LOGIC (5 10) LOGIC (6 10) LOGIC (7 10) LOGIC (8 10) LOGIC (9 10) RAMT (10 10) LOGIC (11 10) LOGIC (12 10) IO (13 10)
IO (0 9) LOGIC (1 9) LOGIC (2 9) RAMB (3 9) LOGIC (4 9) LOGIC (5 9) LOGIC (6 9) LOGIC (7 9) LOGIC (8 9) LOGIC (9 9) RAMB (10 9) LOGIC (11 9) LOGIC (12 9) IO (13 9)
IO (0 8) LOGIC (1 8) LOGIC (2 8) RAMT (3 8) LOGIC (4 8) LOGIC (5 8) LOGIC (6 8) LOGIC (7 8) LOGIC (8 8) LOGIC (9 8) RAMT (10 8) LOGIC (11 8) LOGIC (12 8) IO (13 8)
IO (0 7) LOGIC (1 7) LOGIC (2 7) RAMB (3 7) LOGIC (4 7) LOGIC (5 7) LOGIC (6 7) LOGIC (7 7) LOGIC (8 7) LOGIC (9 7) RAMB (10 7) LOGIC (11 7) LOGIC (12 7) IO (13 7)
IO (0 6) LOGIC (1 6) LOGIC (2 6) RAMT (3 6) LOGIC (4 6) LOGIC (5 6) LOGIC (6 6) LOGIC (7 6) LOGIC (8 6) LOGIC (9 6) RAMT (10 6) LOGIC (11 6) LOGIC (12 6) IO (13 6)
IO (0 5) LOGIC (1 5) LOGIC (2 5) RAMB (3 5) LOGIC (4 5) LOGIC (5 5) LOGIC (6 5) LOGIC (7 5) LOGIC (8 5) LOGIC (9 5) RAMB (10 5) LOGIC (11 5) LOGIC (12 5) IO (13 5)
IO (0 4) LOGIC (1 4) LOGIC (2 4) RAMT (3 4) LOGIC (4 4) LOGIC (5 4) LOGIC (6 4) LOGIC (7 4) LOGIC (8 4) LOGIC (9 4) RAMT (10 4) LOGIC (11 4) LOGIC (12 4) IO (13 4)
IO (0 3) LOGIC (1 3) LOGIC (2 3) RAMB (3 3) LOGIC (4 3) LOGIC (5 3) LOGIC (6 3) LOGIC (7 3) LOGIC (8 3) LOGIC (9 3) RAMB (10 3) LOGIC (11 3) LOGIC (12 3) IO (13 3)
IO (0 2) LOGIC (1 2) LOGIC (2 2) RAMT (3 2) LOGIC (4 2) LOGIC (5 2) LOGIC (6 2) LOGIC (7 2) LOGIC (8 2) LOGIC (9 2) RAMT (10 2) LOGIC (11 2) LOGIC (12 2) IO (13 2)
IO (0 1) LOGIC (1 1) LOGIC (2 1) RAMB (3 1) LOGIC (4 1) LOGIC (5 1) LOGIC (6 1) LOGIC (7 1) LOGIC (8 1) LOGIC (9 1) RAMB (10 1) LOGIC (11 1) LOGIC (12 1) IO (13 1)
IO (1 0) IO (2 0) IO (3 0) IO (4 0) IO (5 0) IO (6 0) IO (7 0) IO (8 0) IO (9 0) IO (10 0) IO (11 0) IO (12 0)

Project Icestorm

- **2019-02-23**: Initial support for iCE40 Ultra devices.
- **2018-01-30**: Released support for iCE40 UltraPlus devices.
- **2017-03-13**: Released support for LP384 chips (in all package variants).
- **2016-02-07**: Support for all package variants of LP1K, LP4K, LP8K and HX1K, HX4K, and HX8K.
- **2016-01-17**: First release of IceTime timing analysis. Video: https://youtu.be/IG5CpFJRnOk
- **2015-12-27**: Presentation of the IceStorm flow at 32C3 (Video on Youtube).
- **2015-07-19**: Released support for 8k chips. Moved IceStorm source code to GitHub.
- **2015-05-27**: We have a working fully Open Source flow with Yosys and Arachne-pnr! Video: http://youtu.be/yUiNlmvVOq8
- **2015-04-13**: Complete rewrite of IceUnpack, added IcePack, some major documentation updates
- **2015-03-22**: First public release and short YouTube video demonstrating our work: http://youtu.be/u1ZHcSNDQMM

  Source: clifford.at/icestorm

Project Icestorm

- Manual analysis of bitstreams to determine commands
- Automatic creation of large numbers of bitstreams to discover routing & functionality config bits
- Also built tools for packing/unpacking bitstreams, generating databases for place-and-route
- First targetted LP/HX1K; then LP/HX 4K/8K; then Ultra/UltraPlus

Yosys

- Open source Verilog synthesis framework
- Development started by Clifford Wolf in 2012
- Support for multiple FPGA families (iCE40, Xilinx 7-series, ECP5, Microsemi) & ASIC synthesis
- Support for formal verification and many other tasks

Yosys

- Typical flow goes from Verilog to a synthesised circuit containing LUTs, flipflops, etc in BLIF, EDIF or JSON format
- Support for inferring block RAM from Verilog arrays
- Experimental support for iCE40 DSP inference from multiplies & adds in Verilog

arachne-pnr

- First open source place-and-route tool for iCE40
- Developed by cseed in 2015 using Project Icestorm
- Fast, simple and lightweight – fulfilled its purpose well
- Tied to iCE40 & hard to port to other FPGA architectures
- Not timing-driven

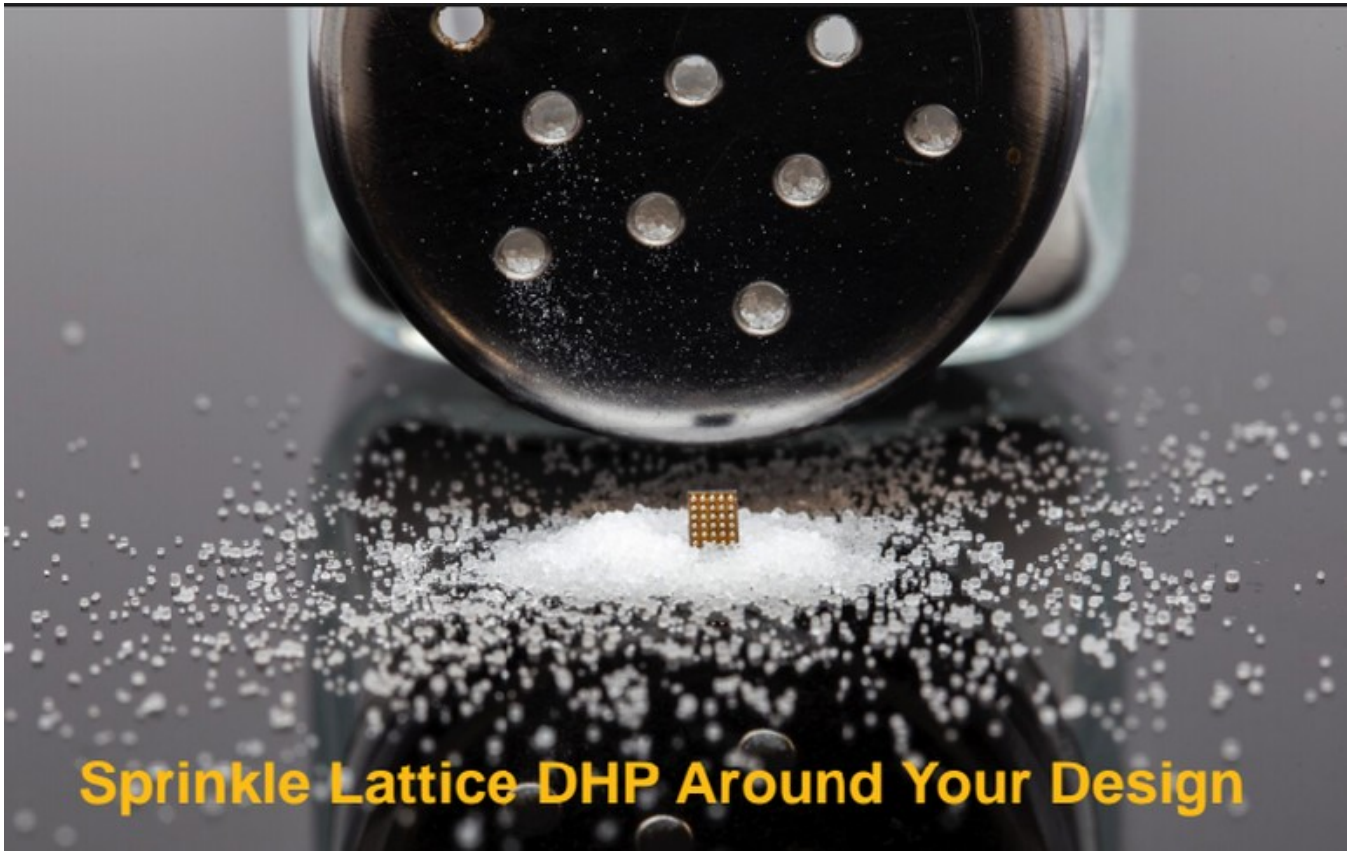
Icestorm Flow

- Complete Yosys, arachne-pnr & icestorm flow released in May 2015
- Lattice quickly sold out of icesticks!
- Enabled a wide range of open source & open hardware FPGA projects

iCE40 UltraPlus

- New series of iCE40 FPGAs released by Lattice in 2016-17
- 5k logic cells
- 8 16x16 DSP cores, 1Mbit single-port RAM – in addition 120kbit usual dual-port block RAM
- Constant current RGB LED pins
- PWM, SPI and I^2C hard IP
- Ultra low power – 100µW idle, CNN accelerator ~8mW

iCE40 UltraPlus



Source: Lattice Semiconductor

iCE40 UltraPlus

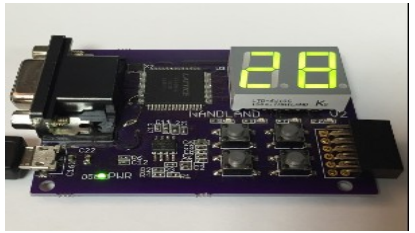
- Icestorm & arachne-pnr support added by January 2018
- Required instantiation of DSPs and single-port RAM (cannot be inferred for Verilog arithmetic / arrays)
- Support for DSP inference from multiply/add operators added February 2019
- Single-port RAM inference support still an outstanding TODO

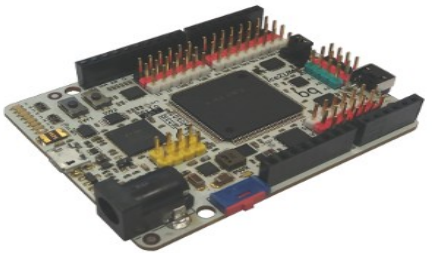
Open FPGA Hardware

@folknology
Alan Wood

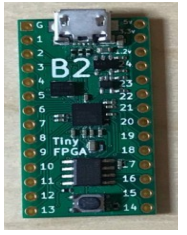
IceStorm 2015 – Vendor Hardware

IceStorm 2016 – Open FPGA

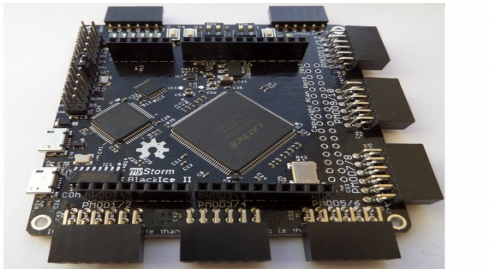


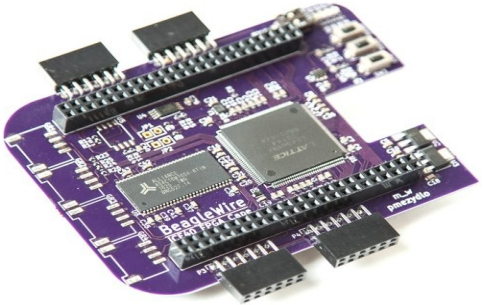


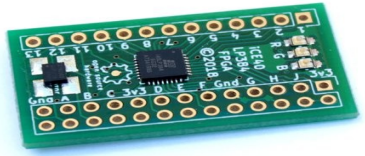
IceStorm 2017 - 2nd Gen OSH



IceStorm 2018 - Present







Open FPGA Toolchains

From Present to Future

David Shah
@fpga_dave
Symbiotic EDA

Slides: fpga.dev/oshug.pdf

Symbiotic EDA

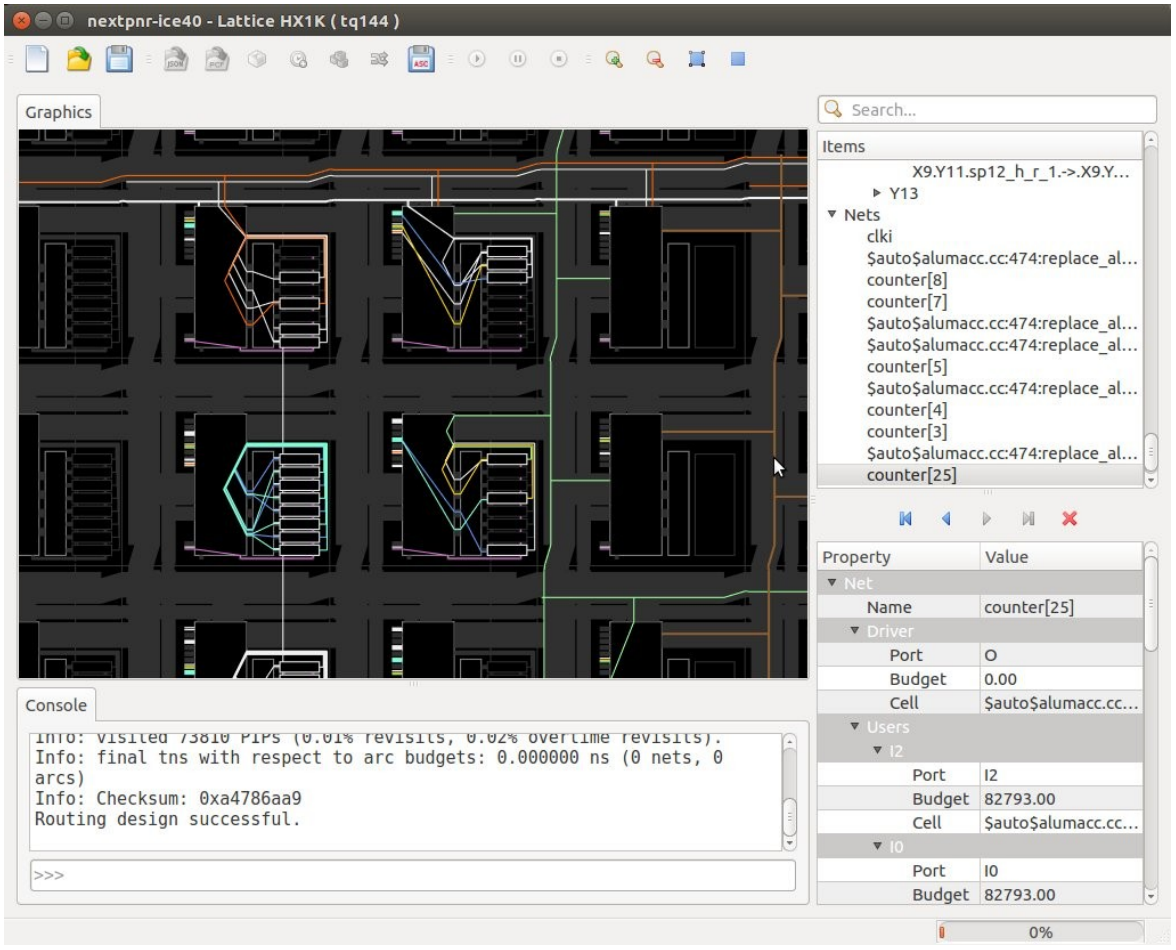
nextpnr

- New open source multi-architecture place and route tool
- Development started early May 2018
- Aimed primarily at real silicon (unlike academic open PnR tools such as VPR)
- Timing driven throughout

nextpnr

- Support for Lattice iCE40 and ECP5 FPGAs
- Work in progress to support various Xilinx FPGAs
- Future “generic” architecture will allow building FPGA using Python API

nextpnr


nextpnr-ice40 - Lattice HX1K (tq144)
Graphics
Search...
Items
X9.Y11.sp12_h_r_1.->.X9.Y...
Y13
Nets
clki
$auto$alumacc.cc:474:replace_al...
counter[8]
counter[7]
$auto$alumacc.cc:474:replace_al...
$auto$alumacc.cc:474:replace_al...
counter[5]
$auto$alumacc.cc:474:replace_al...
counter[4]
counter[3]
$auto$alumacc.cc:474:replace_al...
counter[25]
Property
Value
Net
Name
counter[25]
Driver
Port
O
Budget
0.00
Cell
$auto$alumacc.cc...
Users
I2
Port
I2
Budget
82793.00
Cell
$auto$alumacc.cc...
I0
Port
I0
Budget
82793.00
Console
Info: visited 73810 PIPs (0.01% revisits, 0.02% overtime revisits).
Info: final tns with respect to arc budgets: 0.000000 ns (0 nets, 0 arcs)
Info: Checksum: 0xa4786aa9
Routing design successful.
>>>
0%

nextpnr

- Architectures in nextpnr implement an API rather than providing fixed data files
- Choose how you store the device database based on device size and external constraints
- Custom packer and other functions can be architecture-provided

nextpnr Arch API

- Blackbox ID types: BelId, WireId, PipId
- getBels(), getPips(), getWires(), getPipsUphill(wire): return “some kind of range” of BelId, PipId, etc
- Range must implement begin(), end()
- Iterators must implement ++, *, !=
- Could be anything from a std::vector to custom walker of a deduplicated database!

nextpnr Arch API

- Arch code stored in its own folder, different binary for each arch built
- Enables heavy compile-time optimisation and arch-specific types compared to virtual functions
- Avoids n^2 build complexity of C++ templates

nextpnr

- iCE40 architecture uses a flat database, containing details of everything in the chip with no attempt to remove repetition
- ECP5 much larger, this would mean a database in the GB
- So we use a deduplicated database approach

nextpnr

- During database creation, we build a full flat database in memory using relative coordinates
- Then we split it into grid locations, and use a hash to find identical grid locations
- Identical grid locations only need their content to be stored once in the final database

nextpnr

- Text configuration used as an intermediate format between nextpnr and bitstream generation
- Avoids need for low level bitstream code in nextpnr

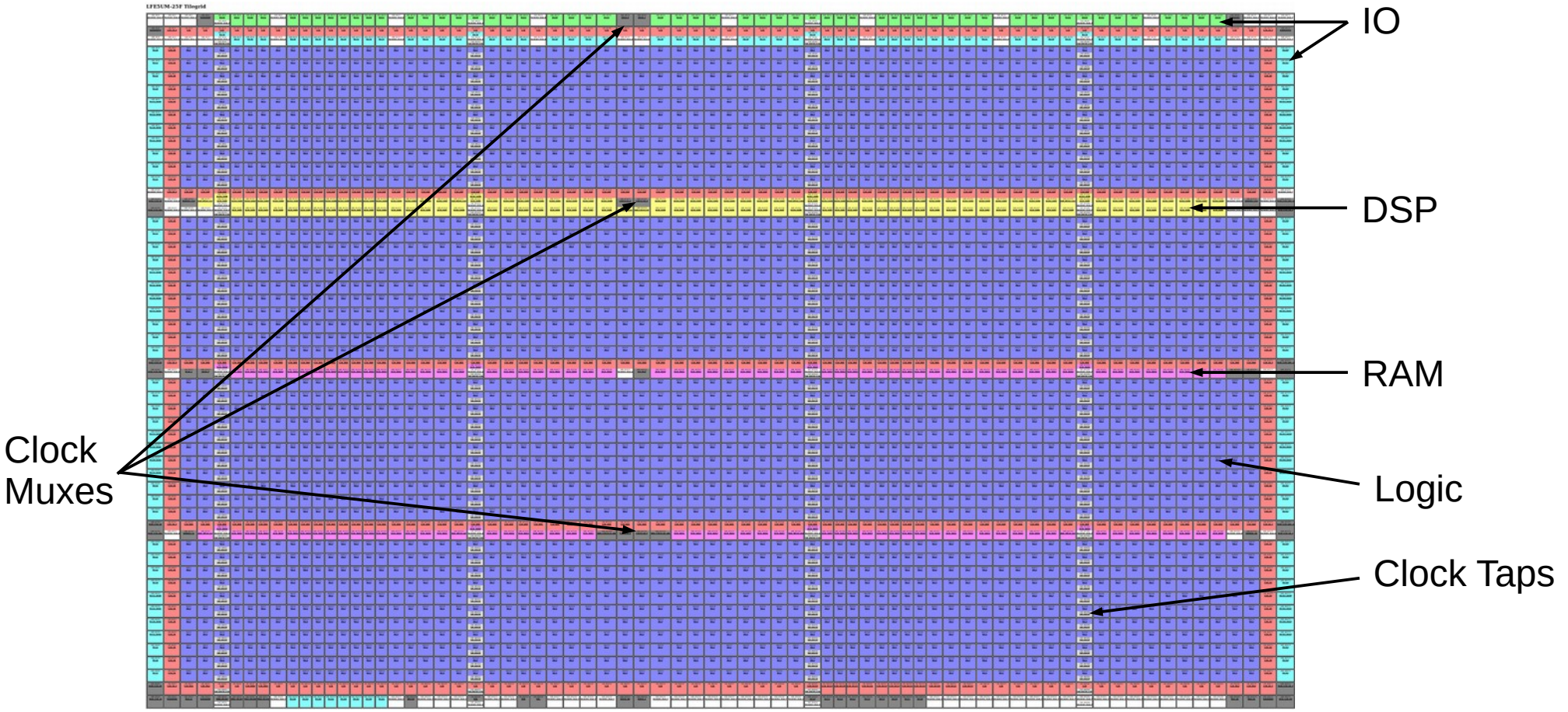
ECP5 FPGA

- Up to 85k logic cells (LUT4+carry+FF)
- Up to 3.7Mb block RAM (in 18Kb blocks), 156 18x18 DSPs
- Available with 3Gbps or 5Gbps SERDES for PCIe, USB 3.0, etc
- Single-quantity pricing starts from $5 (“12k” LE)

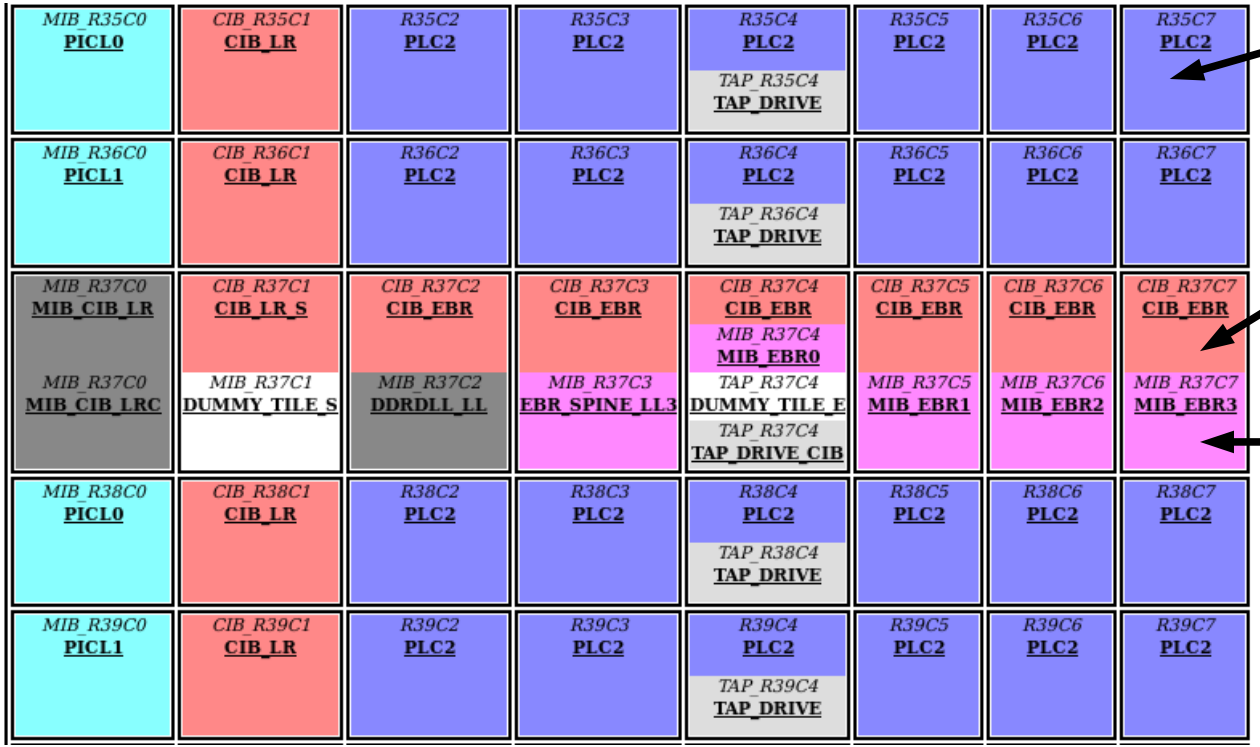
ECP5 Architecture

- Split up into **tiles** of different types. Logic tiles split into 4 **slices**
- **Slice:** 2 LUT + 2FF; carry + 2FF; 16x2 RAM + 2FF; also cascade muxes
- Fixed interconnect **wires**
- **Arcs** connect **wires** together and are configurable or fixed (aka **pip**)
- All arcs and wires are unidirectional – mux topology
- Dedicated global clock network connects to all tiles

ECP5 Architecture



ECP5 Architecture



Logic tiles contain both logic and interconnect

CIB tiles contain interconnect for non-logic functions

MIB tiles contain non-logic functionality (EBR, DSP, IO, etc)

More than one tile at a location is possible!

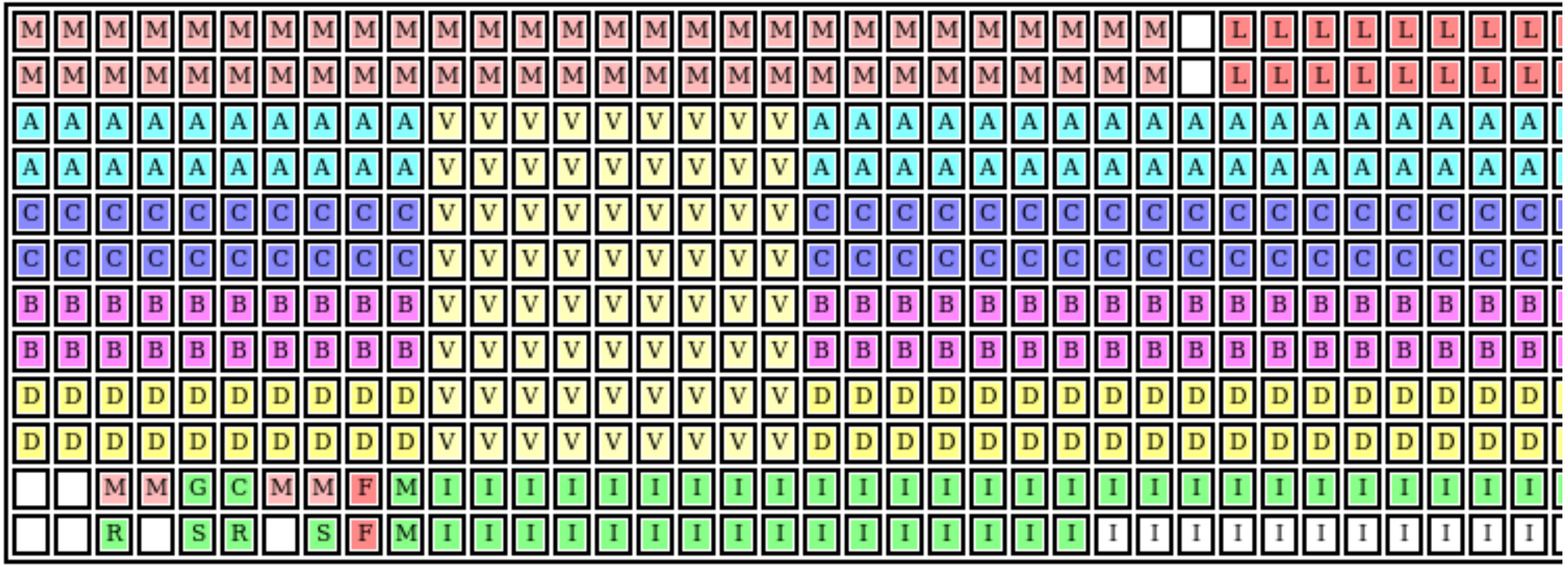
Project Trellis

- Open source bitstream docs & tools for ECP5 FPGAs
- Development started March 2018
- Basic set of bitstream docs May 2018
- Proof-of-concept flow June 2018
- Complete bitstream docs Nov 2018
- Near-complete flow Feb 2019

Trellis Status

- Bit and routing documentation for almost all functionality (missing: obscure DSP modes)
- Timing documentation for fabric, logic cells, IO and BRAM
- Timing-driven Yosys & nextpnr flow supporting majority of functionality including BRAM, PLL, SERDES, DDR memory IO

Trellis Database



Database

Normalised netname
Nominal position is x+3

Frame 104, bit 9
inside tile

Mux driving E3_H06E0003

Source	F101B8	F101B9	F102B8	F103B8	F103B9	F104B8	F104B9	F105B9
W3_H06E0003	1	-	-	1	-	-	-	-
S3_V06N0003	-	1	-	1	-	-	-	-
V06N0003	-	-	1	1	-	-	-	-
W3_H06E0303	1	-	-	-	1	-	-	-
V06S0003	-	1	-	-	1	-	-	-
N3_V06S0003	-	-	1	-	1	-	-	-
W1_H02E0001	1	-	-	-	-	1	-	-
N1_V01S0000	-	1	-	-	-	1	-	-
V01N0001	-	-	1	-	-	1	-	-
W1_H02E0301	1	-	-	-	-	-	1	-
Q0	-	1	-	-	-	-	1	-
Q3	-	-	1	-	-	-	1	-
H01E0001	1	-	-	-	-	-	-	1
F0	-	1	-	-	-	-	-	1
F3	-	-	1	-	-	-	-	1

Database

Configuration word SLICEA.K0.INIT

Default value: 16'b1111111111111111

SLICEA.K0.INIT[0]	!F25B10
SLICEA.K0.INIT[1]	!F24B10
SLICEA.K0.INIT[2]	!F23B10
SLICEA.K0.INIT[3]	!F22B10
SLICEA.K0.INIT[4]	!F21B10
SLICEA.K0.INIT[5]	!F20B10
SLICEA.K0.INIT[6]	!F19B10
SLICEA.K0.INIT[7]	!F18B10
SLICEA.K0.INIT[8]	!F17B10
SLICEA.K0.INIT[9]	!F16B10
SLICEA.K0.INIT[10]	!F15B10
SLICEA.K0.INIT[11]	!F14B10
SLICEA.K0.INIT[12]	!F13B10
SLICEA.K0.INIT[13]	!F12B10
SLICEA.K0.INIT[14]	!F11B10
SLICEA.K0.INIT[15]	!F10B10

Database

Configuration Setting EBR0.DP16KD.DATA_WIDTH_A

Default value: 18

Value	F40B0	F47B0	F51B0	F78B0
1	1	1	1	1
2	1	0	1	1
4	1	0	1	0
9	0	0	1	0
18	0	0	0	0

Configuration Setting EBR0.DP16KD.WRITEMODE_A

Default value: NORMAL

Value	F7B0	F101B0
NORMAL	0	0
READBEFOREWRITE	0	1
WRITETHROUGH	1	0

Text Configuration

- Need to make use of & test fuzz results
- Tools to convert bitstreams to/from a text config format
- Check that output is logical for simple designs
- Check for unknown bits in larger designs

Text Configuration

```
.tile R53C71:PLC2
arc: A1 W1_H02E0701
arc: A3 H02E0701
arc: A4 H02E0501
arc: A5 V00B0000
arc: A7 W1_H02E0501
arc: B0 S1_V02N0301
arc: S3_V06S0303 W3_H06E0303
arc: W1_H02W0401 V02S0401
word: SLICEA.K0.INIT 1100110000000000
word: SLICEA.K1.INIT 1010101000000000
enum: SLICEA.CCU2.INJECT1_0 NO
enum: SLICEA.CCU2.INJECT1_1 NO
enum: SLICEA.D0MUX 1
enum: SLICEA.D1MUX 1
enum: SLICEA.MODE CCU2
```

Timing

- Need to know how large internal delays are to determine if a design can work at a given frequency
- Like bitstream format, not enough vendor documentation
- Delays for cells (LUTs, etc) extracted from SDF files
- Interconnect delays determined using least-squares linear fit

Capabilities

- Linux-capable VexRiscv SoC with Ethernet and DDR3 memory
- 64-bit RISC-V SoC
- High-speed IO interfaces such as PCIe, DDR3 memory, HDMI, cameras

Current Work

- Faster analytic placer
- More optimisations in synthesis - using better interfaces to ABC, the logic optimisation & mapping framework Yosys uses
- DSP, shift register and improved RAM inference
- Support for Xilinx devices in nextpnr

Analytic Placement

- Analytic placer to augment the current simulated annealing placer
- SA used by arachne-pnr, originally in nextpnr and most “first-generation” FPGA flows (ISE, older Quartus)
- Analytic placement can be both faster and give higher quality results

Analytic Placement

- Based around solving a system of equations to determine optimum placement of cells
- But naïve approach would put every cell more-or-less at one location in the center!
- One option: iterative solving and then spreading
- Add a weight in later iterations to keep cells close to their spread (legal) position

DSP Inference

- Converting the * operator in Verilog and surrounding functions to hard DSP primitives
- Clifford has developed “pmgen” framework for netlist pattern matches
- Support for iCE40 DSPs including registers, adders and accumulator muxes

Open FPGA Toolchains

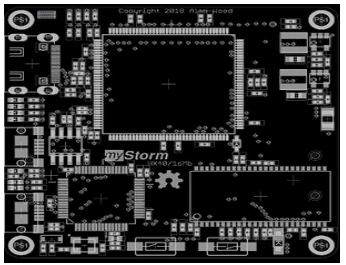
ECP5 SoC demo

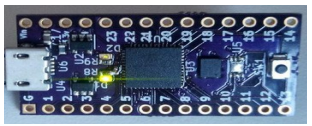
David Shah
@fpga_dave
Symbiotic EDA

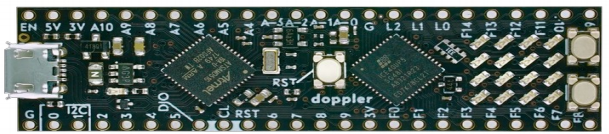
Slides: fpga.dev/oshug.pdf

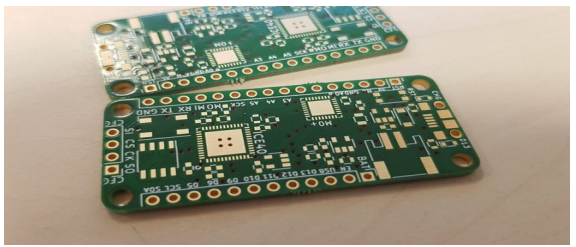
Symbiotic EDA

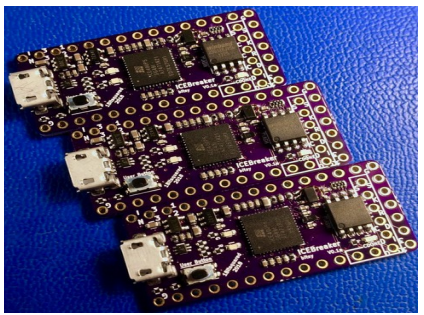
IceStorm - Near Future

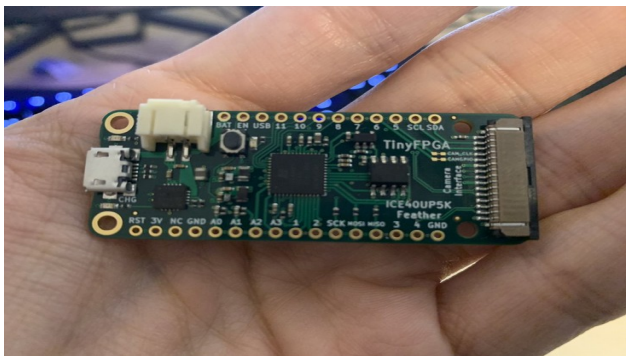




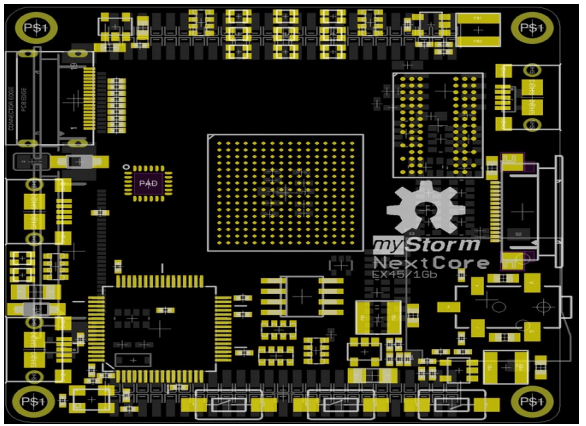


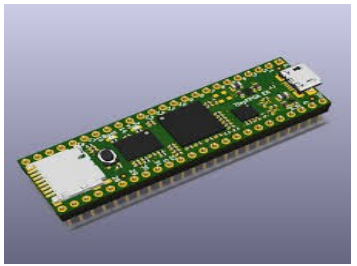






NextPnr – OpenFPGA ECP5





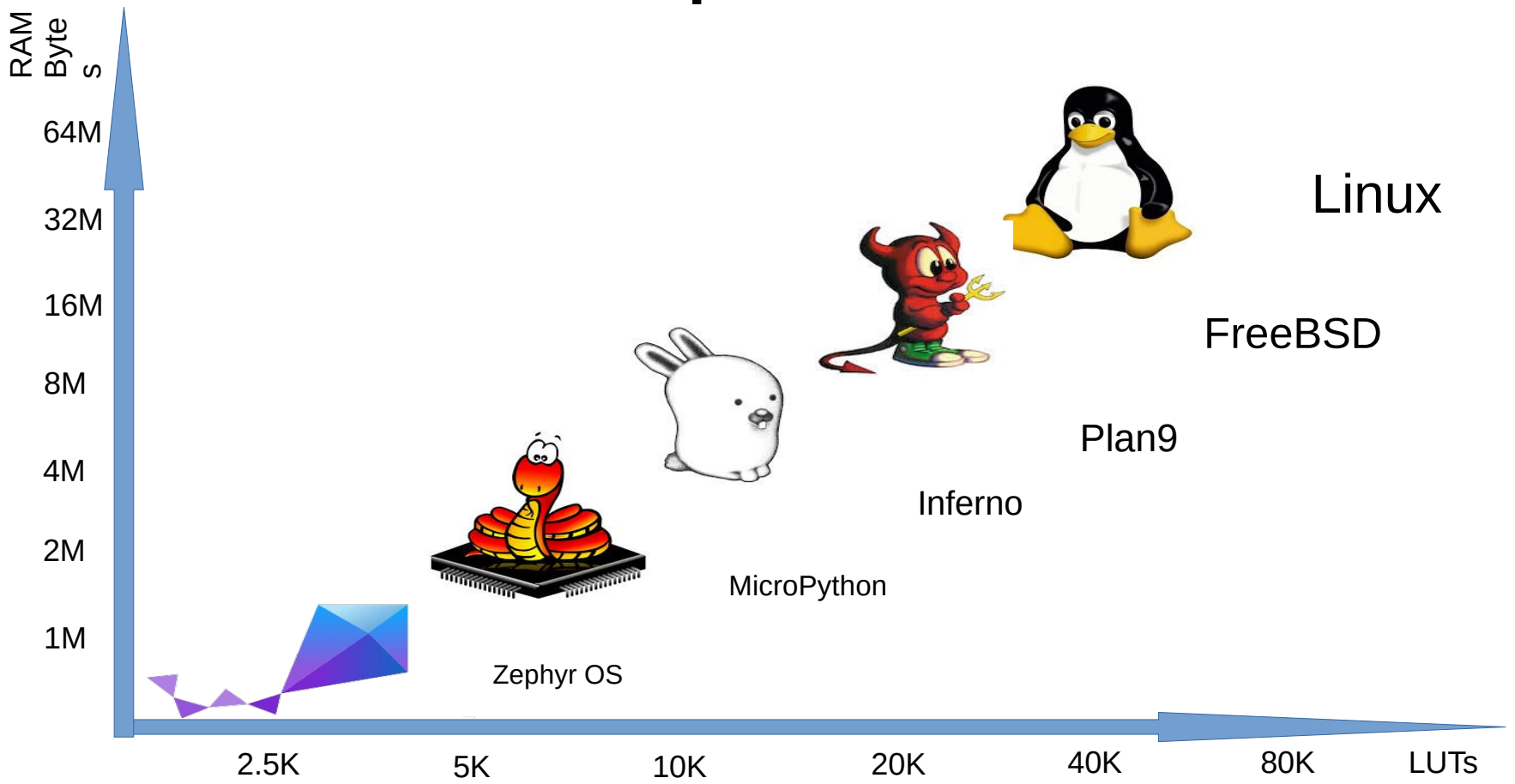
OpenFPGA – ECP5 Features

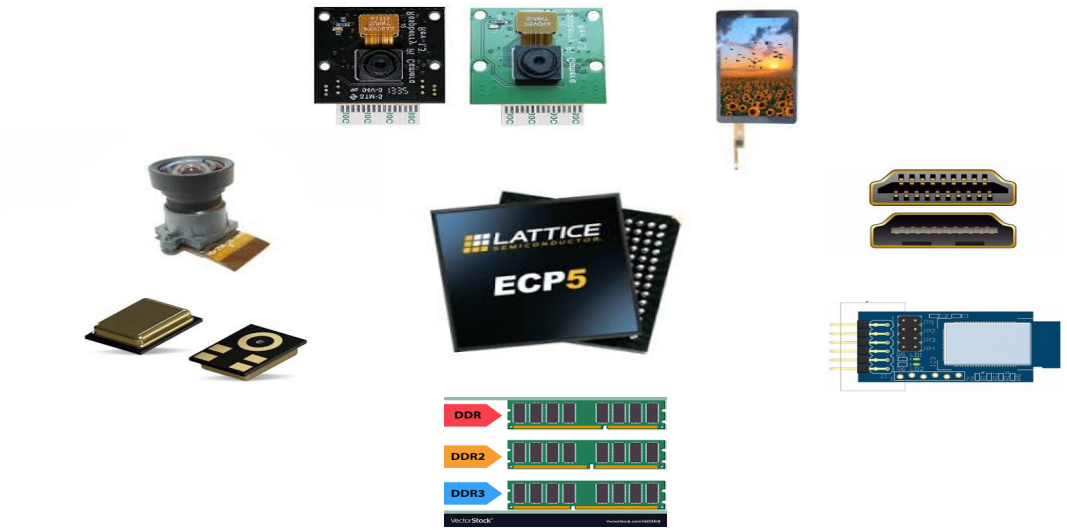
- Many more resources Up to 85K Luts
- Built in Blcok Ram up to 3.7Mbit
- DSP units up 156 18 x 18 Multiply
- Optional Serdes packages at 3/5Gbps
- IO gearing to support DDR RAM and DDR LVDS or high speed Lvds for CSI or MIPI

OpenFPGA – ECP5 new apps

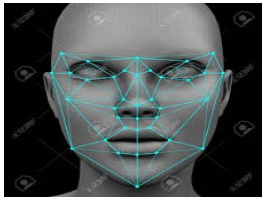
- SOCs and multicores with larger/faster memories, buses and IO
- Sophisticated Operating System support using above combined with larger resources (Luts) e.g. Free BSD or Linux embedded apps
- Video and Frame buffers for faster graphics output (HDMI up to 1080p 30Fps), faster edp using optional Serdes resources
- Many channel realtime 24 bit Audio processing and voice recognition
- Embedded Video capture, processing and Machine Vision applications
- Custom embedded application accelerators, RL, ANNs and NTMs.
- Robotics multiple FOC BLDC motor control, complex biped and quadraped propulsion systems

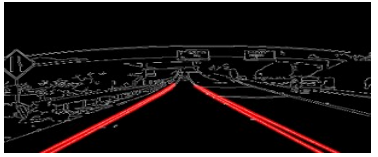
SOC Requirements

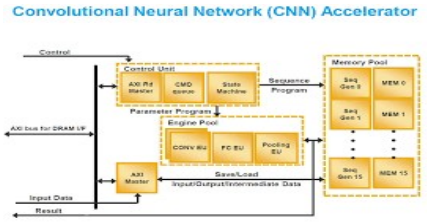


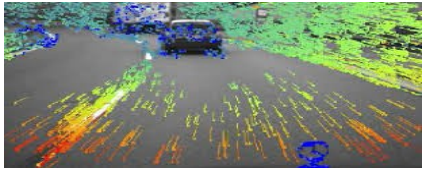

LATTICE
SEMICONDUCTOR
ECP5
DDR
DDR2
DDR3
VectorStock

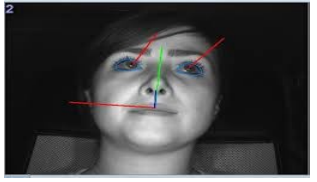
Machine Vision & Learning

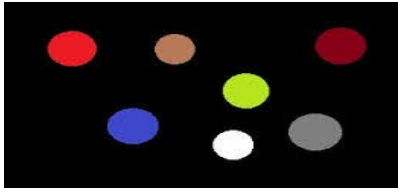






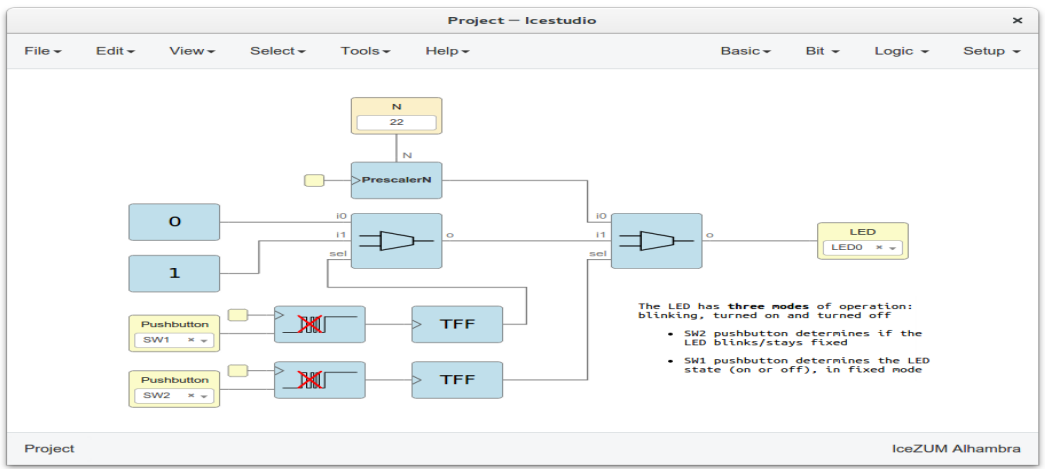


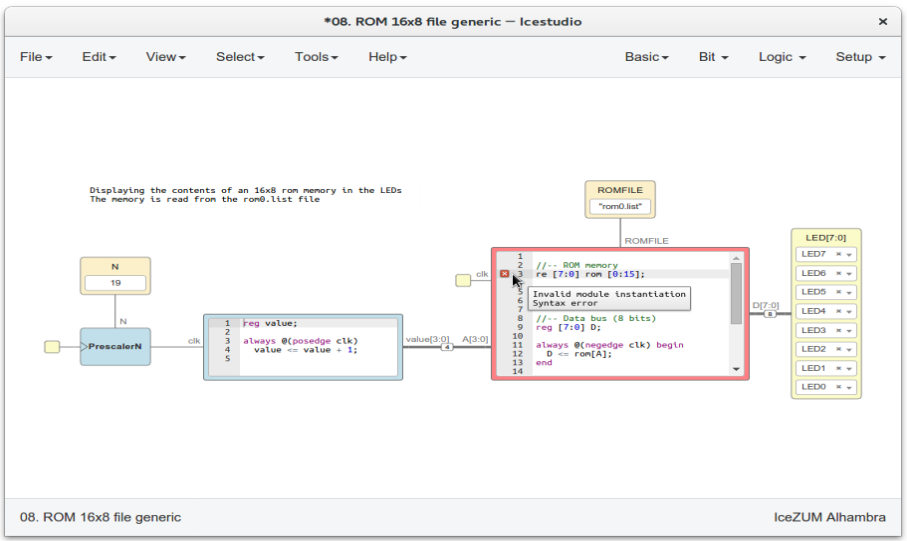




IceStudio

icestudio





Recognise This?

```
writeBack plug new Area{
  import writeBack._
  cache.io.cpu.writeBack.isValid := arbitration.isValid && input(MEMORY_ENABLE)
  cache.io.cpu.writeBack.isStuck := arbitration.isStuck
  cache.io.cpu.writeBack.isUser  := (if(privilegeService != null) privilegeService.isUser(writeBack)
  if(genAtomic) cache.io.cpu.writeBack.clearAtomicEntries := service(classOf[IContextSwitching]).isC

  if(catchSomething) {
    exceptionBus.valid := cache.io.cpu.writeBack.mmuMiss || cache.io.cpu.writeBack.accessError || ca
    exceptionBus.badAddr := cache.io.cpu.writeBack.badAddr
    exceptionBus.code.assignDontCare()
    when(cache.io.cpu.writeBack.illegalAccess || cache.io.cpu.writeBack.accessError){
      exceptionBus.code := (input(MEMORY_WR) ? U(15) | U(13)).resized
    }
    when(cache.io.cpu.writeBack.unalignedAccess){
      exceptionBus.code := (input(MEMORY_WR) ? U(6) | U(4)).resized
    }
    when(cache.io.cpu.writeBack.mmuMiss){
      exceptionBus.code := (input(MEMORY_WR) ? U(15) | U(13)).resized
    }
  }
  arbitration.haltItself.setWhen(cache.io.cpu.writeBack.haltIt)

  val rspShifted = Bits(32 bits)
  rspShifted := cache.io.cpu.writeBack.data
  switch(input(MEMORY_ADDRESS_LOW)){
    is(1){rspShifted(7 downto 0) := cache.io.cpu.writeBack.data(15 downto 8)}
    is(2){rspShifted(15 downto 0) := cache.io.cpu.writeBack.data(31 downto 16)}
    is(3){rspShifted(7 downto 0) := cache.io.cpu.writeBack.data(31 downto 24)}
  }
```

Recognise This?

```
class Timer(Module, AutoCSR):
    def __init__(self, width=32):
        self._load = CSRStorage(width)
        self._reload = CSRStorage(width)
        self._en = CSRStorage()
        self._update_value = CSR()
        self._value = CSRStatus(width)

        self.submodules.ev = EventManager()
        self.ev.zero = EventSourceProcess()
        self.ev.finalize()

        ###

        value = Signal(width)
        self.sync += [
            If(self._en.storage,
                If(value == 0,
                    # set reload to 0 to disable reloading
                    value.eq(self._reload.storage)
                ).Else(
                    value.eq(value - 1)
                )
            ).Else(
                value.eq(self._load.storage)
            ),
            If(self._update_value.re, self._value.status.eq(value))
        ]
        self.comb += self.ev.zero.trigger.eq(value != 0)
```

Recognise This?

```
int maybe_handle_illegal(struct pt_regs *regs)
{
    uint32_t insn;
    get_user(insn, (uint32_t*)regs->sepc);
    //DBGMSG("trying to handle illegal instr %08x", insn);
    if ((insn & 0x7F) == 0x2F) {
        //DBGMSG("handling A instr %08x", insn);
        // Atomic instructions - FIXME: make these actually atomic...
        uint32_t rs1v = get_register_value(regs, (insn >> 15) & 0x1F);
        uint32_t rs2v = get_register_value(regs, (insn >> 20) & 0x1F);
        uint32_t rdv = 0;
        uint32_t funct = (insn >> 27) & 0x1F;
        if (funct == 0x2) {
            // LR
            get_user(rdv, (uint32_t*)rs1v);
        } else if (funct == 0x3) {
            // SC
            put_user(rs2v, (uint32_t*)rs1v);
            rdv = 0;
        } else {
            //AMO
            uint32_t opres;
            get_user(rdv, (uint32_t*)rs1v);
            switch (funct) {
                case 0x00: //AMOADD
                    opres = rdv + rs2v;
                    break;
                case 0x01: // AMOSWAP
```

Demo Time!

- VexRiscv based SoC on the ECP5 Versa
- VexRiscv: 32-bit RISC-V processor in SpinalHDL (Scala based)
- LiteX used for SoC integration (Python based)
- Uses LiteX cores – litedram for DDR3 and liteeth for Gigabit Ethernet
- Built with nextpnr, Yosys & Trellis

Demo Time!

- Experimental Linux port to VexRiscv
- Non-standard MMU, timer & interrupts (work to fix these in progress)
- Missing atomic instructions needed by userspace emulated in illegal instruction trap handler

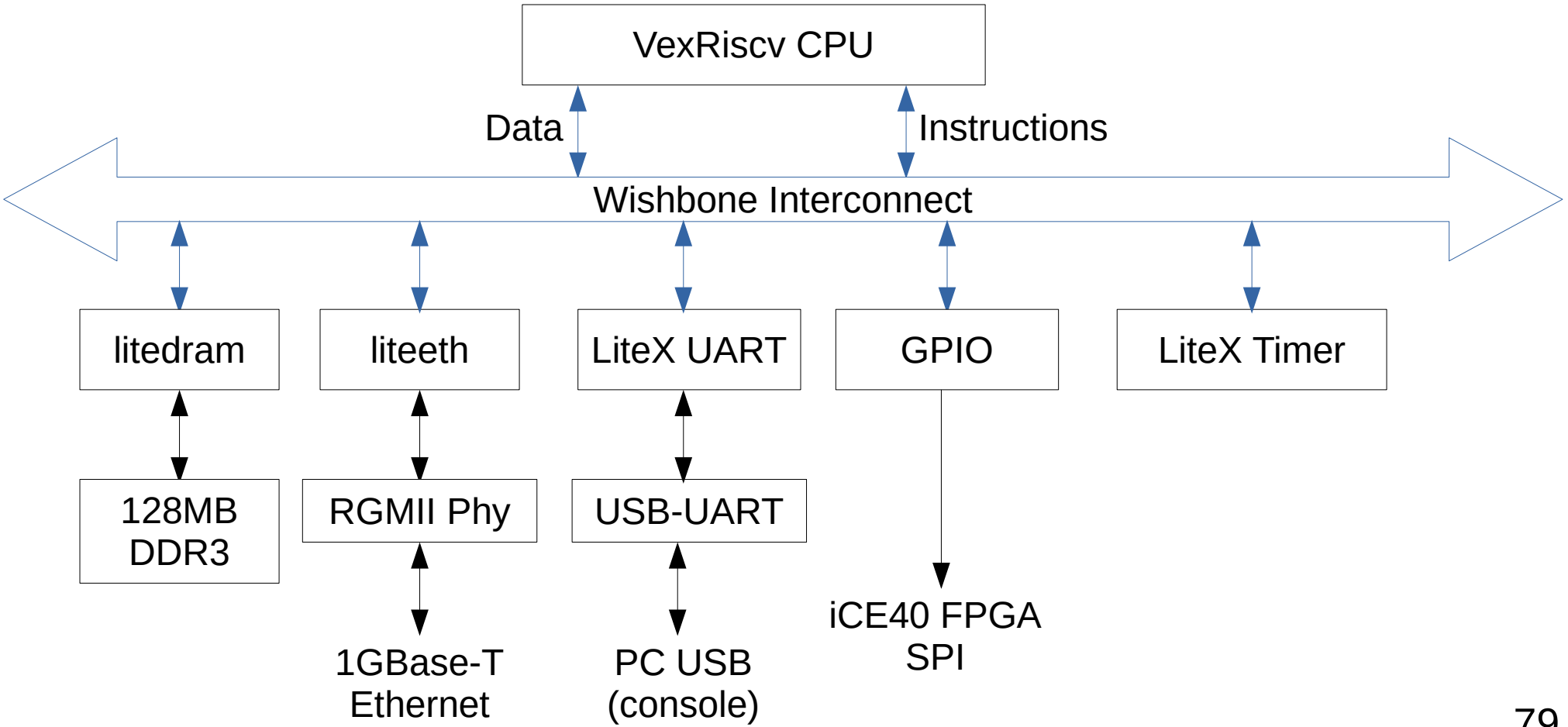
Demo Time!

- 128MB of RAM is enough to run the open FPGA toolchain
- C/C++ tools with few dependencies, easy to cross compile for RISC-V
- Buildroot rootfs with FPGA tools on an NFS share for ease of development
- LiteX provides TFTP bootloader, handy for kernel dev

Demo Time!

- Mystorm Ice board (iCE40LP384)
- SPI programming port connected to some GPIO on the Versa
- Easy to add GPIO with LiteX
- Simple bitbang programming tool

Demo Time!


VexRiscv CPU
Data
Instructions
Wishbone Interconnect
litedram
liteeth
LiteX UART
GPIO
LiteX Timer
128MB
DDR3
RGMII Phy
USB-UART
1GBase-T
Ethernet
PC USB
(console)
iCE40 FPGA
SPI