CS448H

Agile Hardware Design

Pat Hanrahan
CA: Ross Daly

Winter 2017

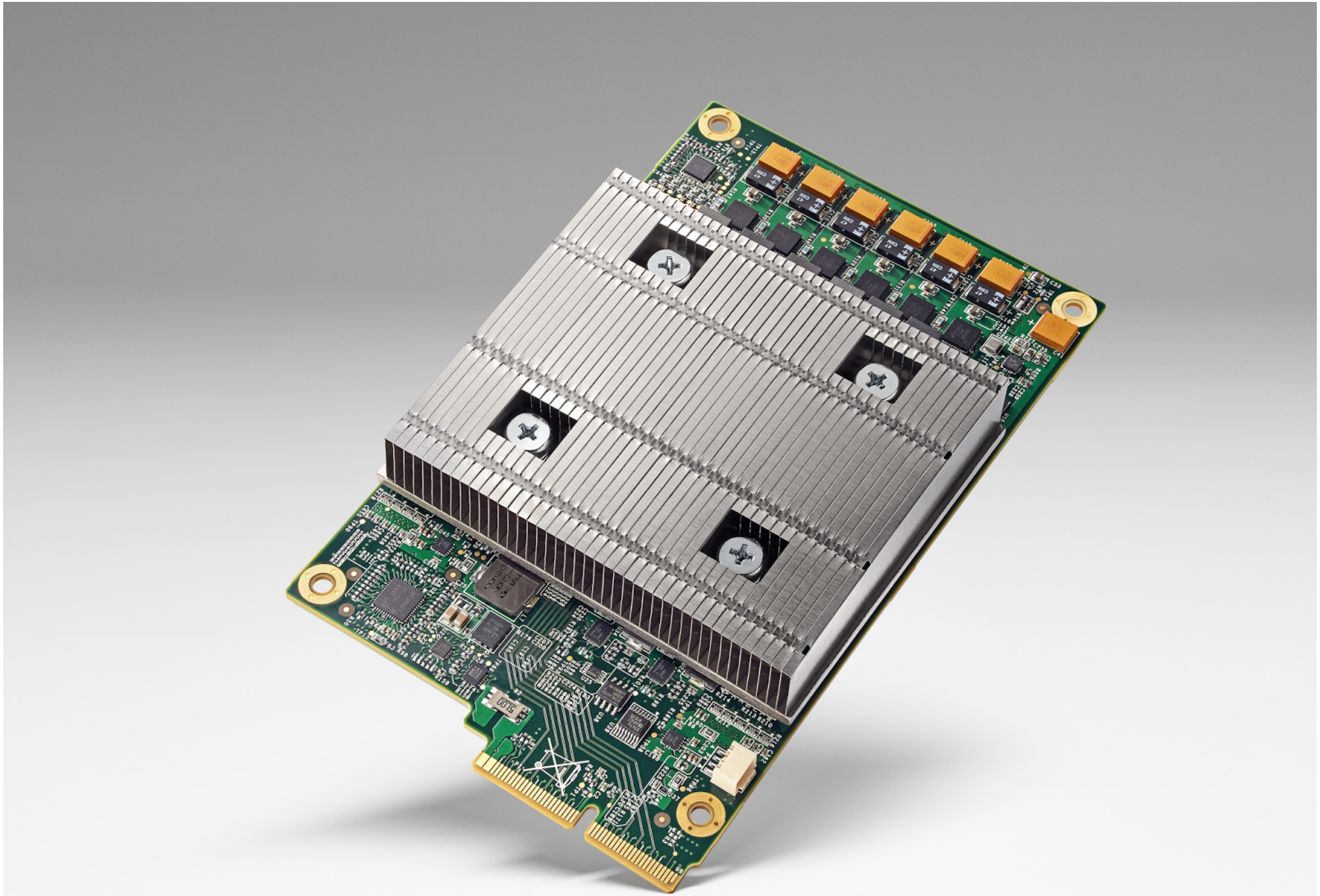
https://github.com/rdaly525/CS448H

Hardware Innovation

Apple iPhone 7

Google Pixel

Embedded Computing and the IOT



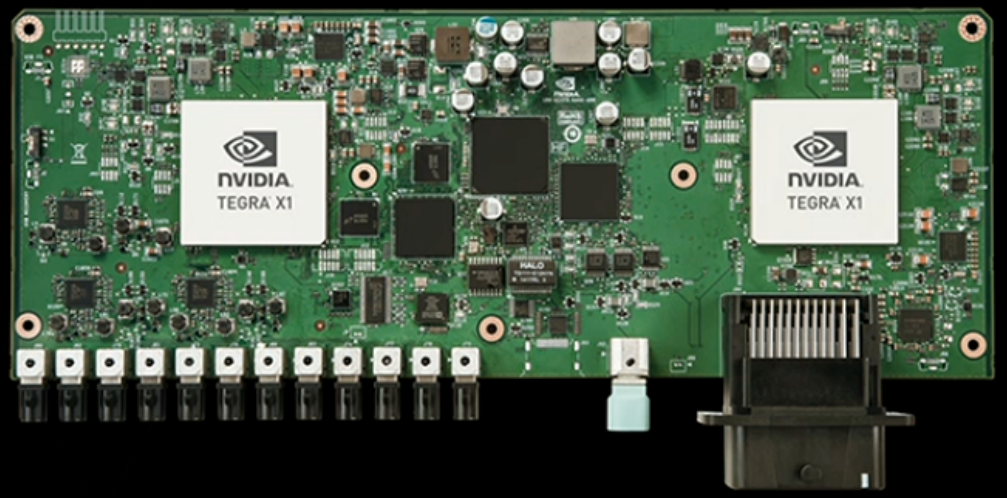
Google Tensor Processing Unit

INTRODUCING NVIDIA DRIVE™ PX

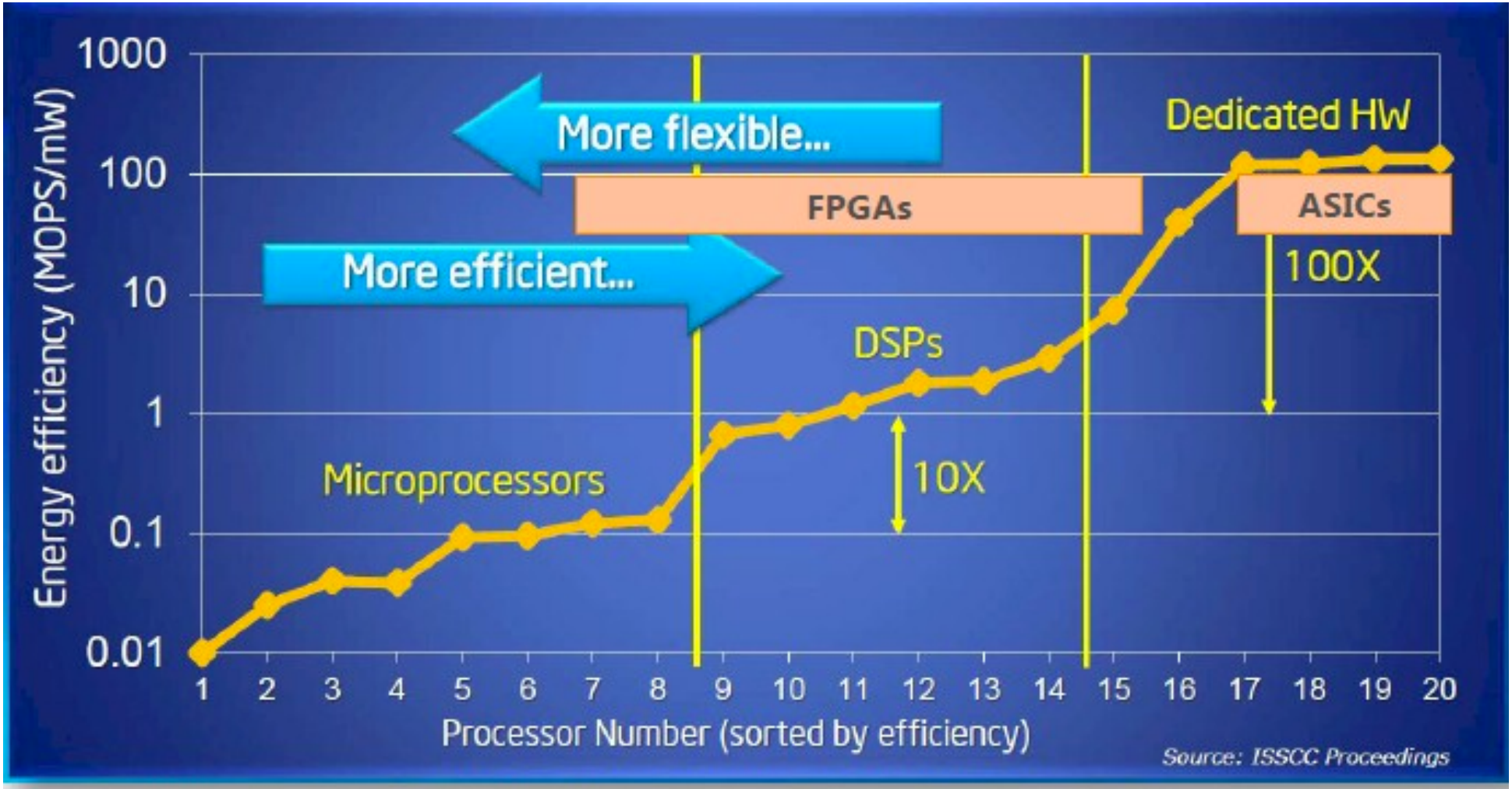
AUTO-PILOT CAR COMPUTER

Dual Tegra X1 • 12 camera inputs • 1.3 GPix/sec

- 2.3 Teraflops mobile supercomputer
- CUDA programmability
- Deep Neural Network Computer Vision
- Surround Vision



http://www.nvidia.com/object/drive-px.html



Source: Bob Broderson, Berkeley Wireless group

Hardware Design is Hard

Expensive

- **"$100M to build a chip"**
- **Large team of designers**
- **Hard to test and verify**
- **Expensive to make masks for chips**

=> Lots of money, long time to market, ...

Less innovation, fewer startups (compared to software)

Hardware IS Software

Basic abstractions

- **Combination logic (functions)**
- **Sequential logic (*state*)**
- **Parallel in *space***

Finite *state* machines (FSM)

The circuits specifying the FSM are programs

The design tools are meta-programs

Why Can't Hardware Design be as Easy and Fun as Software?

Easier?

Better Design Software!

Verilog is Horrible!

```
generate
   for (i = 0; i < (1 << `RAMS); i=i+1) begin : ram
     // RAMB16_S18_S18
     RAMB16_S2_S2
     ram(
       .DIA(0),
       // .DIPA(0),
       .DOA(insn[`w*i+`w1:`w*i]),
       .WEA(0),
       .ENA(1),
       .CLKA(sys_clk_i),
       .ADDRA({_pc}),

       .DIB(st1[`w*i+`w1:`w*i]),
       // .DIPB(2'b0),
       .WEB(_ramWE & (_st0[15:14] == 0)),
       .ENB(|_st0[15:14] == 0),
       .CLKB(sys_clk_i),
       .ADDRB(_st0[15:1]),
       .DOB(ramrd[`w*i+`w1:`w*i]));
   end
 endgenerate

 // Compute the new value of T.
 always @*
 begin
   if (insn[15])
     _st0 = immediate;
   else
     case (st0sel)
       4'b0000: _st0 = st0;
       4'b0001: _st0 = st1;
       4'b0010: _st0 = st0 + st1;
       4'b0011: _st0 = st0 & st1;
       4'b0100: _st0 = st0 | st1;
       4'b0101: _st0 = st0 ^ st1;
       4'b0110: _st0 = ~st0;
       4'b0111: _st0 = {16{(st1 == st0)}};
       4'b1000: _st0 = {16{($signed(st1) < $signed(st0))}};
4'b1001: _st0 = st1 >> st0[3:0];
       4'b1010: _st0 = st0 - 1;
       4'b1011: _st0 = rst0;
       4'b1100: _st0 = |st0[15:14] ? io_din : ramrd;
       4'b1101: _st0 = st1 << st0[3:0];
       4'b1110: _st0 = {rsp, 3'b000, dsp};
       4'b1111: _st0 = {16{(st1 < st0)}};
       default: _st0 = 16'hxxxx;
     endcase
```

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity grove_analogue is
    Port ( clk32     : in  STD_LOGIC;
           adc_clk   : out  STD_LOGIC;
           adc_dout  : in  STD_LOGIC;
           adc_din   : out  STD_LOGIC;
           adc_cs    : out  STD_LOGIC;
           c0_data   : out  STD_LOGIC_VECTOR (11 downto 0);
           c0_strobe : out  STD_LOGIC;
           c1_data   : out  STD_LOGIC_VECTOR (11 downto 0);
           c1_strobe : out  STD_LOGIC;
```

Some issues ...

1. Weak abstraction of hardware
2. Poor programming language
   - Poor type system
   - Poor support for meta-programming
3. Easy to write programs that can't be synthesized
   - Designed for simulation
4. Klunky tools (GUIs)

```
begin
   adc_cs <= cs;

process(clk32)
   begin
      if rising_edge(clk32) then
         -- Generate the serial clock
         adc_clk <= not count(0);

         if count(0) = '0' then
            captured <= captured(captured'high-1 downto 0) & adc_dout;
```

A Success Story

Inexpensive Microcontrollers

Atmel

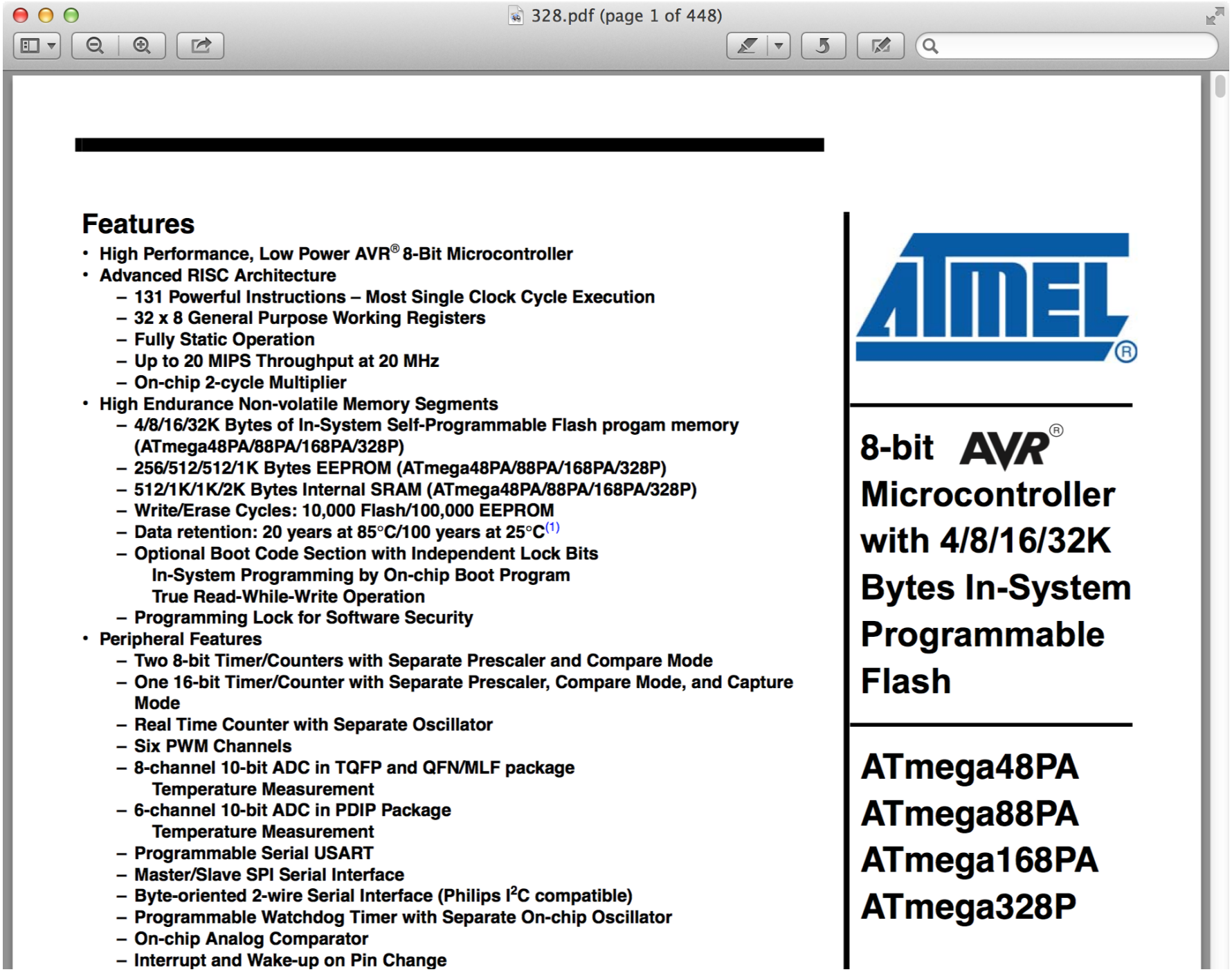
ATMEGA328-PU

ATmega Series 20 MHz 32 KB Flash 2 KB SRAM 8-Bit Microcontroller - DIP-28

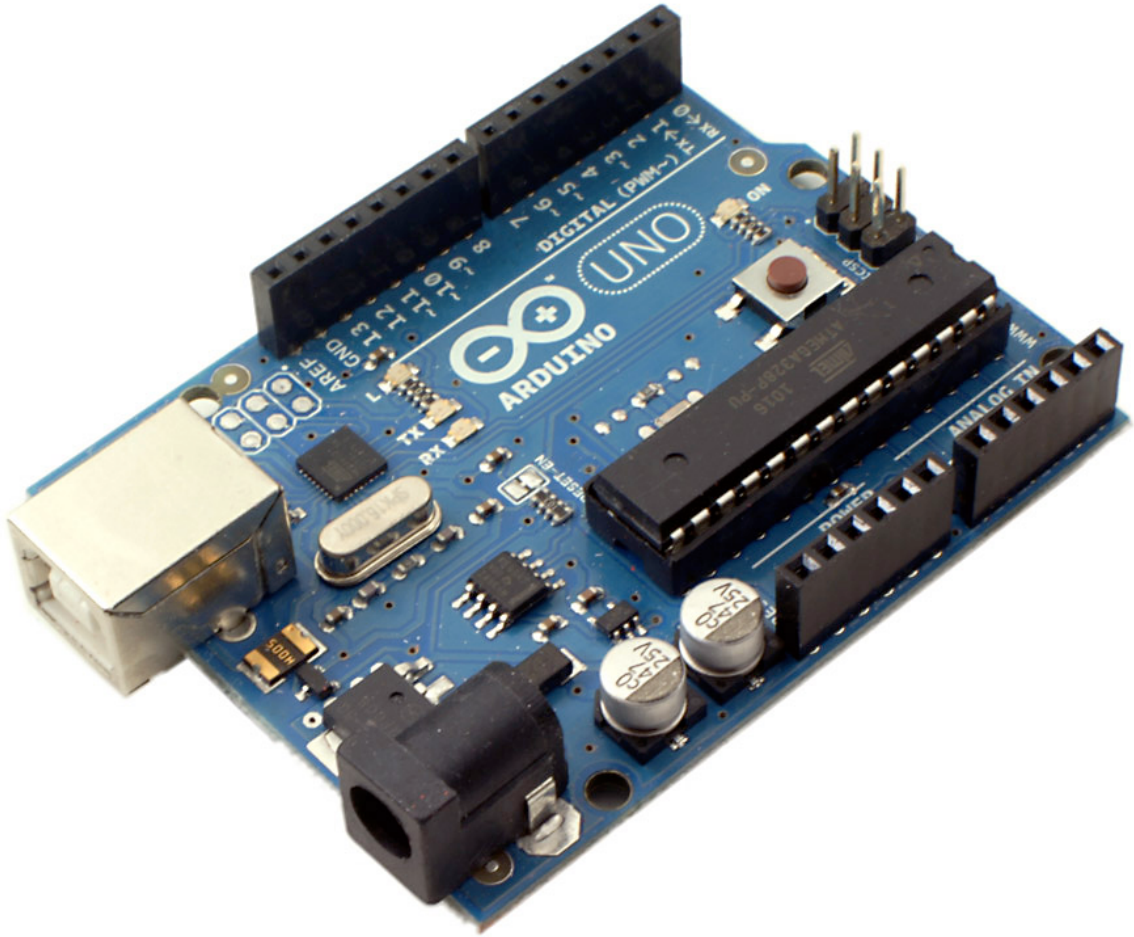
+ Add to BOM | Datasheet | 5

	Distributor	SKU	Stock	MOQ	Pkg		1	10	100	1,000	10,000
★	Arrow	ATMEGA328-PU	468 ◇	1		USD	2.04	1.91	1.78	1.71	1.71
★	Verical (2)	ATMEGA328-PU	468 ◇	14		USD			1.76	1.71	1.71
		ATMEGA328-PU	716 ◇	7		USD		3.88	3.88	3.88	3.88
★	Newark	68T2932	12	1		USD	3.46	3.08	2.52	1.71	1.71
★	Chip One Stop Japan	C1S124600199590	716 ◇	1		USD	3.10	2.70	1.84	1.78	1.78
★	Rochester Electronics	ATMEGA328-PU	4,963			USD	2.55	2.55	2.27	2.07	2.07
★	Chip One Stop Global	C1S124600199590	468 ◇	1		USD	3.33	2.95	1.86	1.84	1.84
★	Future Electronics	ATMEGA328-PU	115	10		USD		1.63	1.63	1.63	1.63
★	Farnell	1972087	1,775	1		USD *	2.69	2.51	2.04	1.70	1.70
★	element14 APAC	1972087	464	1		USD *	3.61	3.22	2.65	2.04	2.04
★	RS Components	7380435	926								
☆	Ewing Components	ATMEGA328-PU	7,013								

Low-Level Programming is Not Simple!

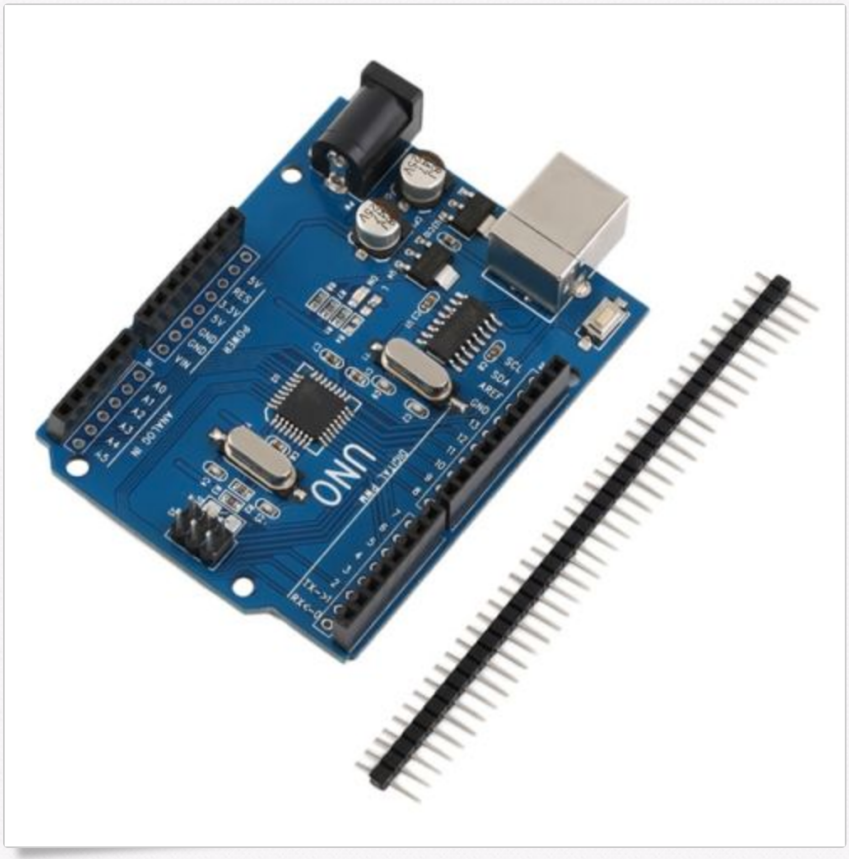


Arduino



"It had to be the equivalent of going out to dinner at a pizza place," Massimo Banzi

http://spectrum.ieee.org/geek-life/hands-on/the-making-of-arduino



UNO R3 ATmega328P Development Board With Boot Loader For Arduino UNO GK

Item condition: **New**

Option: - Select -

Quantity: 1 More than 10 available / 28 sold

Price: **C $1.28**
Approximately **US $0.97**

Buy It Now

Add to cart

1 watching

Add to watch list

Add to collection

28 sold	More than 57% sold	Free shipping

Shipping: **FREE** Standard Int'l Shipping | See details
See details about international shipping here.
Item location: Selangor, Malaysia
Ships to: Worldwide See exclusions

Delivery: Estimated between **Wed. Jan. 25 and Tue. Feb. 14**
Seller ships within 1 day after receiving cleared payment .
Please note the delivery estimate is **greater than 8 business days.**

Payments: PayPal VISA MasterCard AMERICAN EXPRESS DISCOVER

Seller information

keep-going11 (14891 ★)
98.7% Positive feedback

Follow this seller

Visit store: keep-going11
See other items

Wiring Makes it Easy!

```
/*
  Blink
  Turns on an LED on for one second, then off for one second, repeatedly.
 */

int led = 13; // LED connected to Pin 13

// the setup routine runs once when you press reset:
void setup() {
  pinMode(led, OUTPUT);
}

// the loop routine runs over and over again forever:
void loop() {
  digitalWrite(led, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);               // wait for a second (1000 msecs)
  digitalWrite(led, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);               // wait for a second
}
```

Keys to Wide Adoption

1. **Wiring (easy to use)**
   - ***Wiring* by Hernando Barragán**
   - **Based on *Processing* by Ben Fry / Casey Reas**

2. **Open source hardware and software (<$)**
   - **AVR port of** gcc
   - **Arduino board is open hardware**

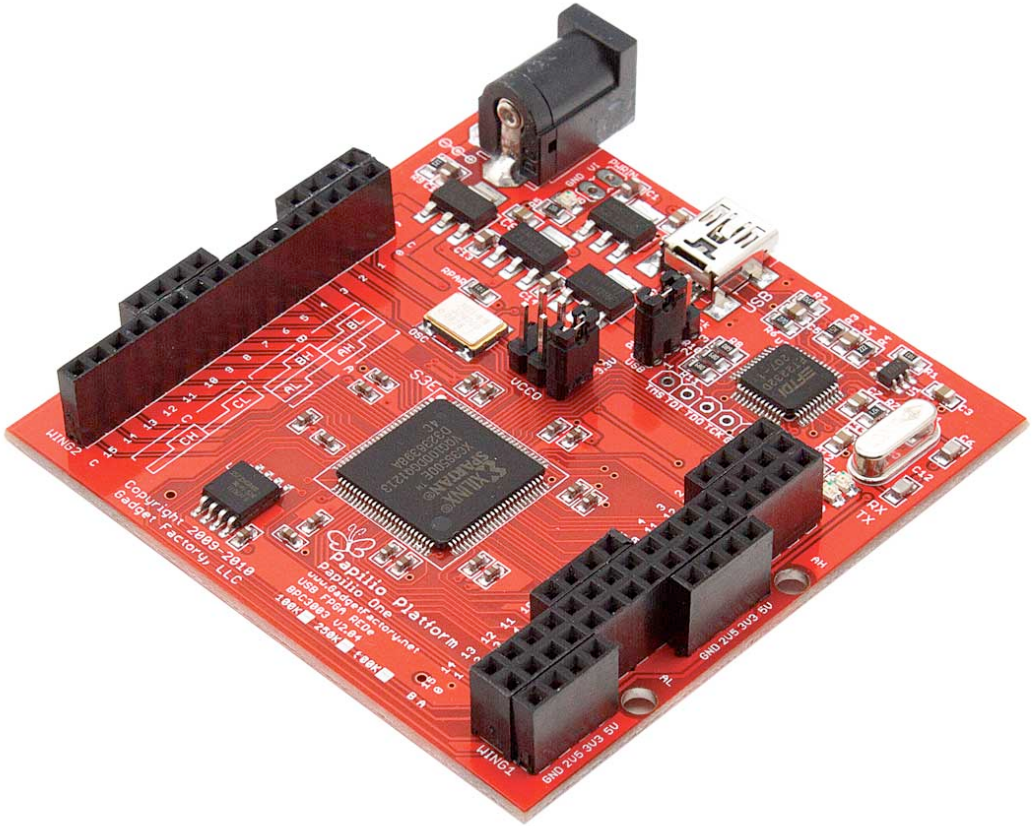
Is now having a big impact on embedded computing

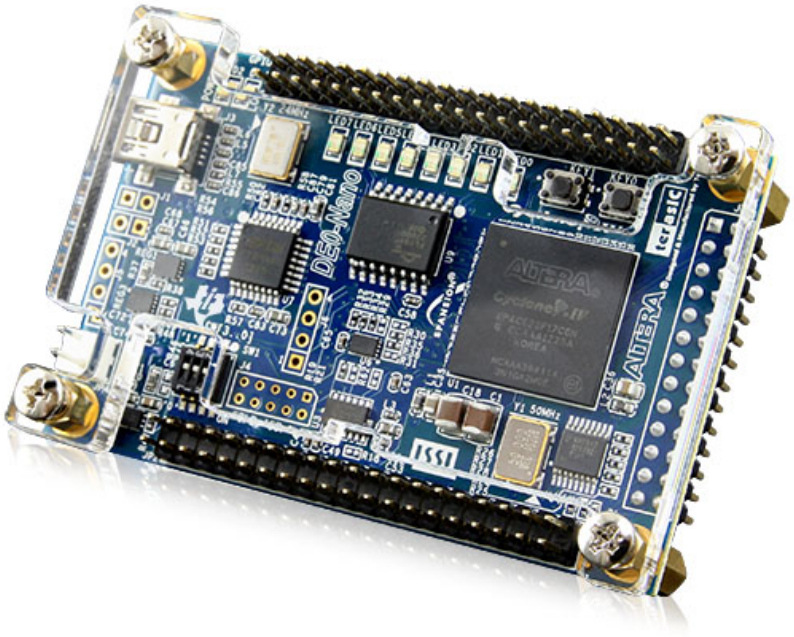
Reconfigurable Hardware

FPGA
(Field Programmable Gate Array_

Xilinx

Altera

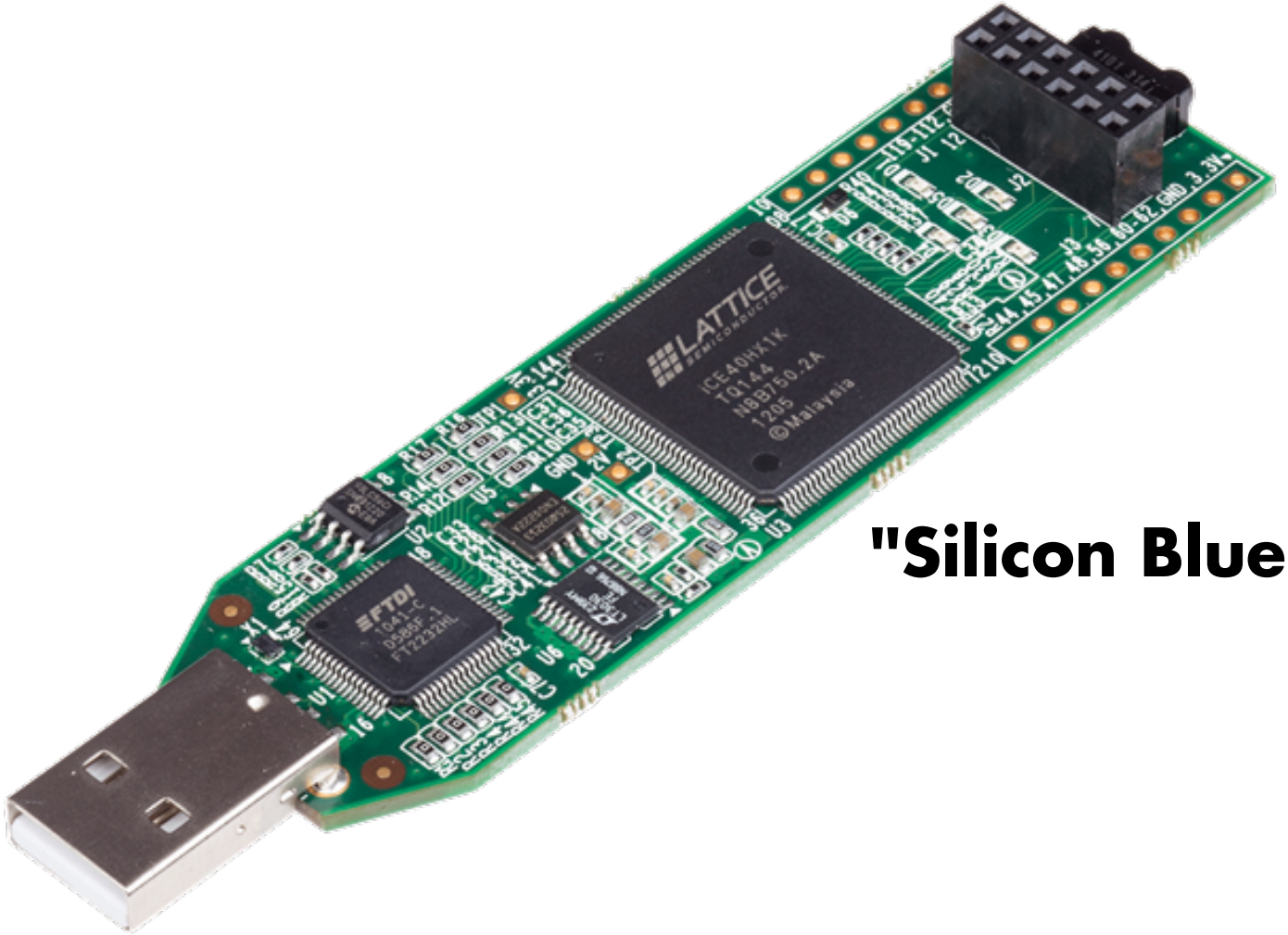




Papilio

DE0 Nano

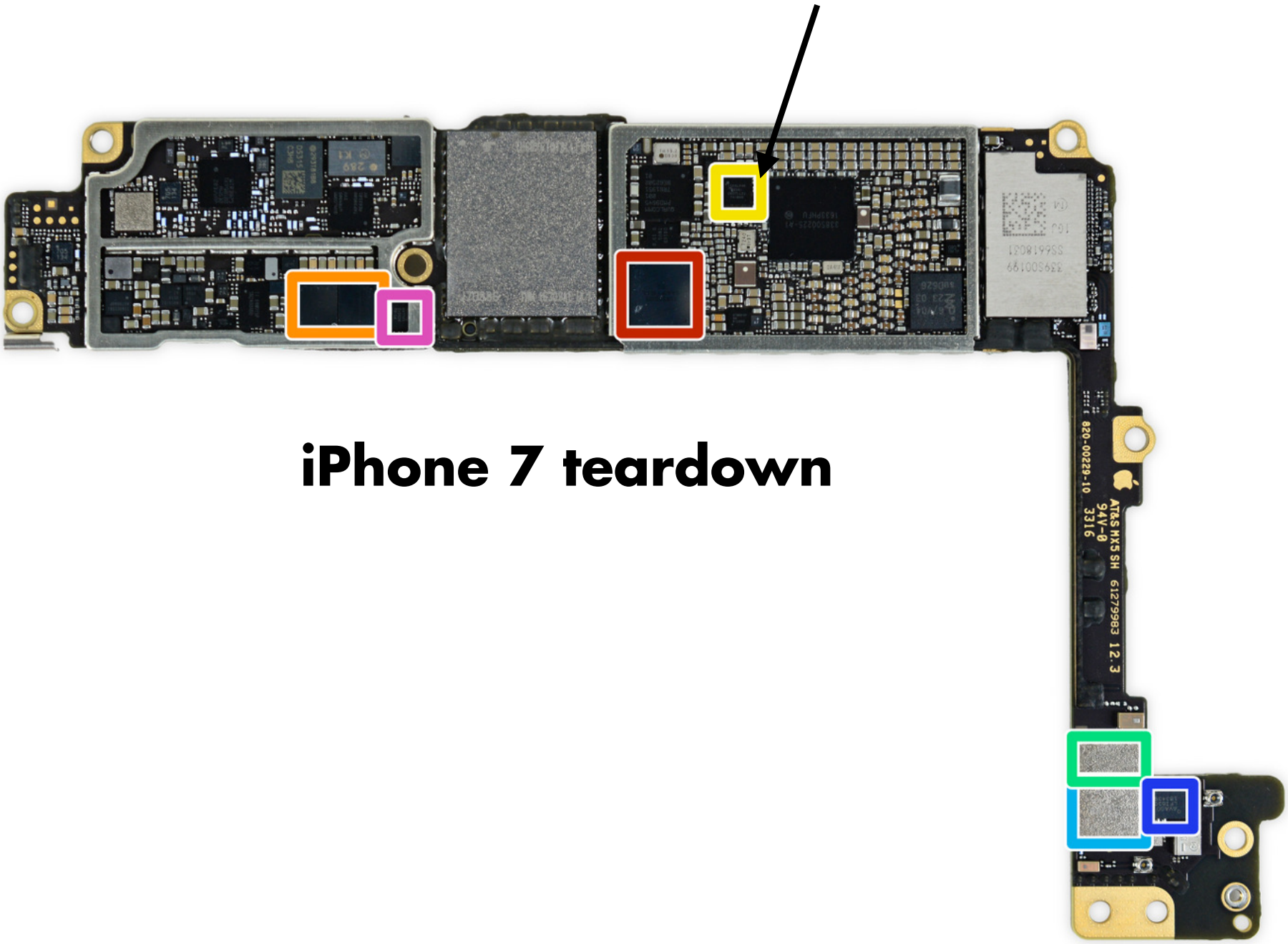
Lattice Icestick ($22)



"Silicon Blue" ICE40 FPGA

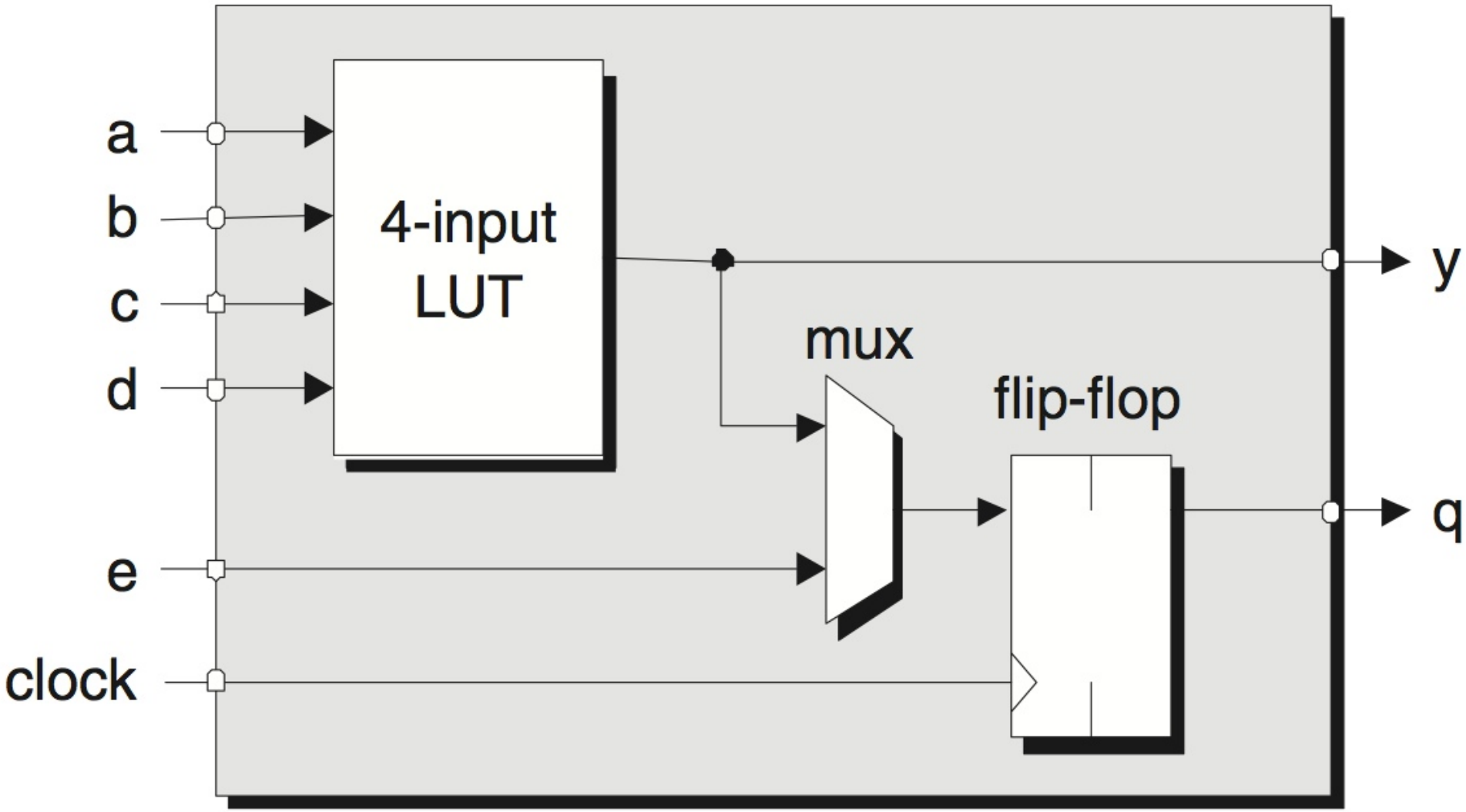
Open source toolchain: yosys, arachne, icestorm

Lattice Semiconductor ICE5LP4K

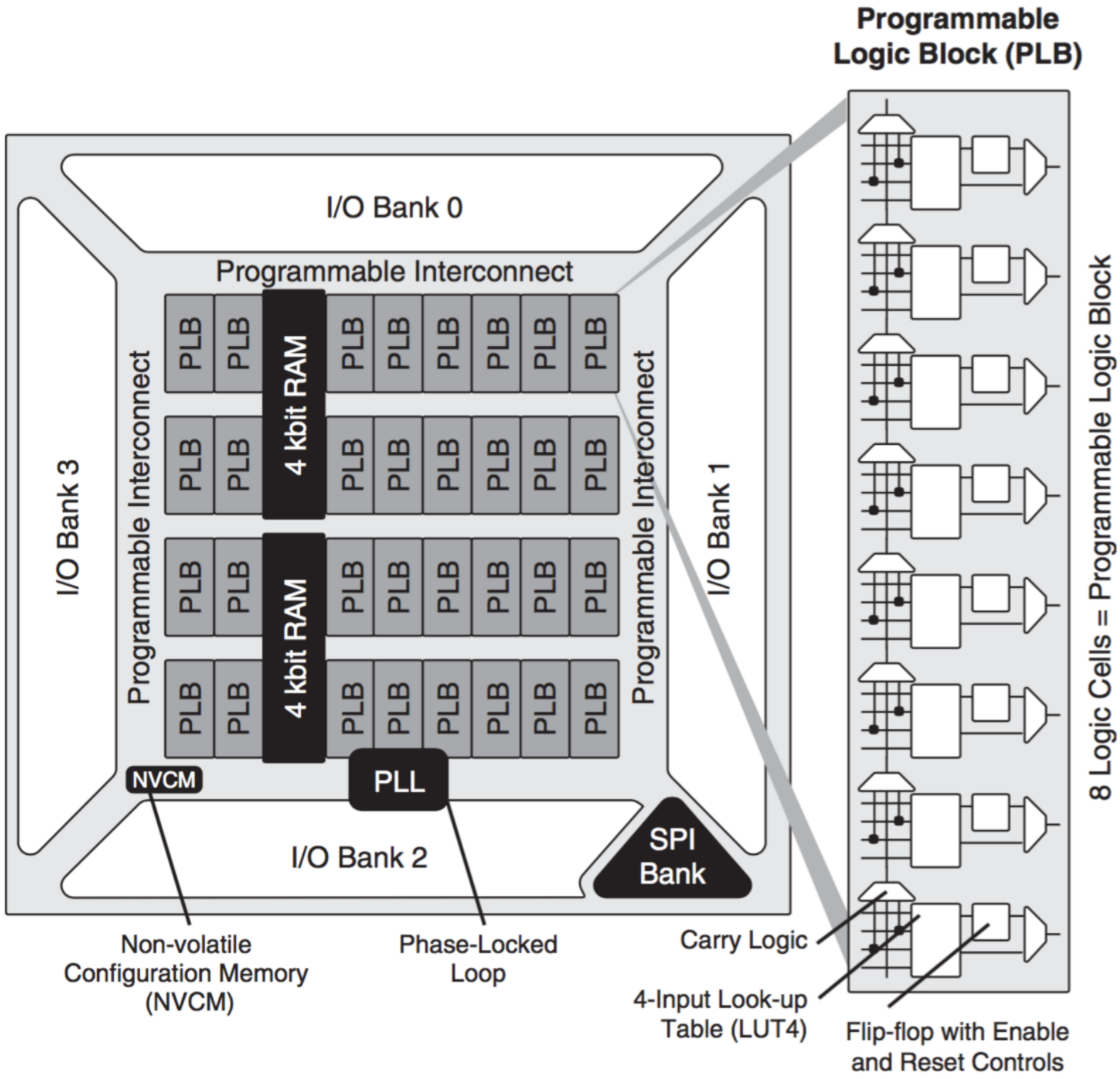


iPhone 7 teardown

FPGAs are Simple!



Basic Elements of Logic: LUT + MUX + FF

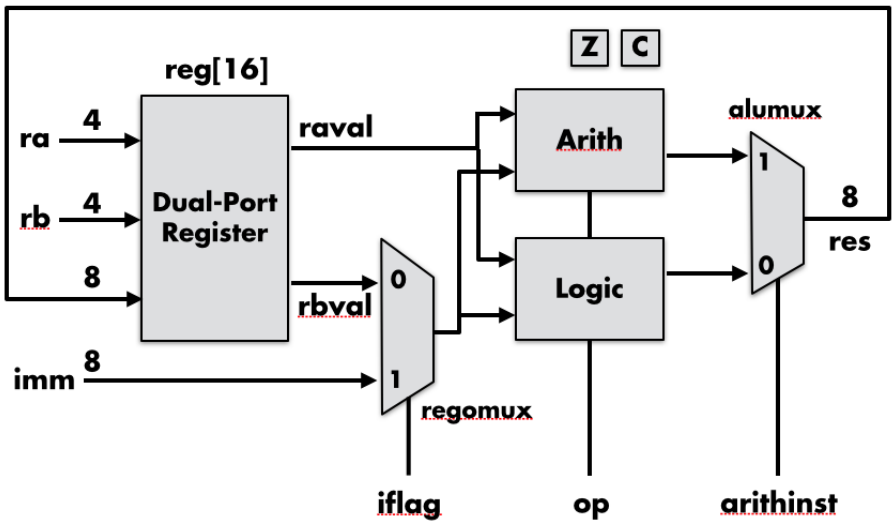

Programmable
Logic Block (PLB)
I/O Bank 0
Programmable Interconnect
I/O Bank 3
Programmable Interconnect
Programmable Interconnect
I/O Bank 1
PLB
4 kbit RAM
4 kbit RAM
NVCM
PLL
I/O Bank 2
SPI
Bank
8 Logic Cells = Programmable Logic Block
Non-volatile
Configuration Memory
(NVCM)
Phase-Locked
Loop
Carry Logic
4-Input Look-up
Table (LUT4)
Flip-flop with Enable
and Reset Controls

Magma / Icestick Demo

```
mem = readmem('a.mem')

ADDRN = 10
DATAN = 18
N = 8

# program memory
rom = ROMB( mem, DATAN, site=(0,0) )

# condition code registers
z = FF( site=(2,0) )
c = FF( site=(2,2) )

# condition codes
expr = '(~A & ~(B^C)) | (A & ~(B^D))'
cc = LUT( expr, site=(3,6) ) ( zcflag, nflag, z, c )
jump = LUT( 'A & (~B | (B & C))', site=(3,7) )( jumpinst, ccflag, cc )

# register write, z and c write
regwr = LUT( 'A & ((B&C)|(~B&D))', site=(3,8) ) ( step,wflag,aluinst,ioinst )
car = zwr   = LUT( 'A & B', site=(3,9) ) ( step, aluinst )
iord = LUT('A & ~B', site=(3,10)) (ioinst, wflag)
iowr = LUT('A &  B', site=(3,11)) (ioinst, wflag)

print 'Building sequencer'
seq = Sequencer( ADDRN, site=(0,0) ) # site must be even, ...
pc = seq( 1, addr, jump, ce=step )
inst = rom( pc, ce=step )

print 'Building registers'
regimux = Mux( 2, N, site=(5,0) )
reg = DualPortRegister( N, site=(6,0) ) # site x must be even
regomux = Mux( 2, N, site=(7,0) )

# register values
raval, rbval = reg[0], reg[1]
rbval = regomux( [imm, rbval], iflag )

print 'Building logic unit'
logicunit = Logic( N, site=(8,0) )

print 'Building arith unit'
arithunit = Arith( N, site=(9,0) )
alumux = Mux( 2, N, site=(11,0) )

print 'Wiring logic unit'
logicres = logicunit( raval, rbval, op[0], op[1] )
```

Magma Hackathon

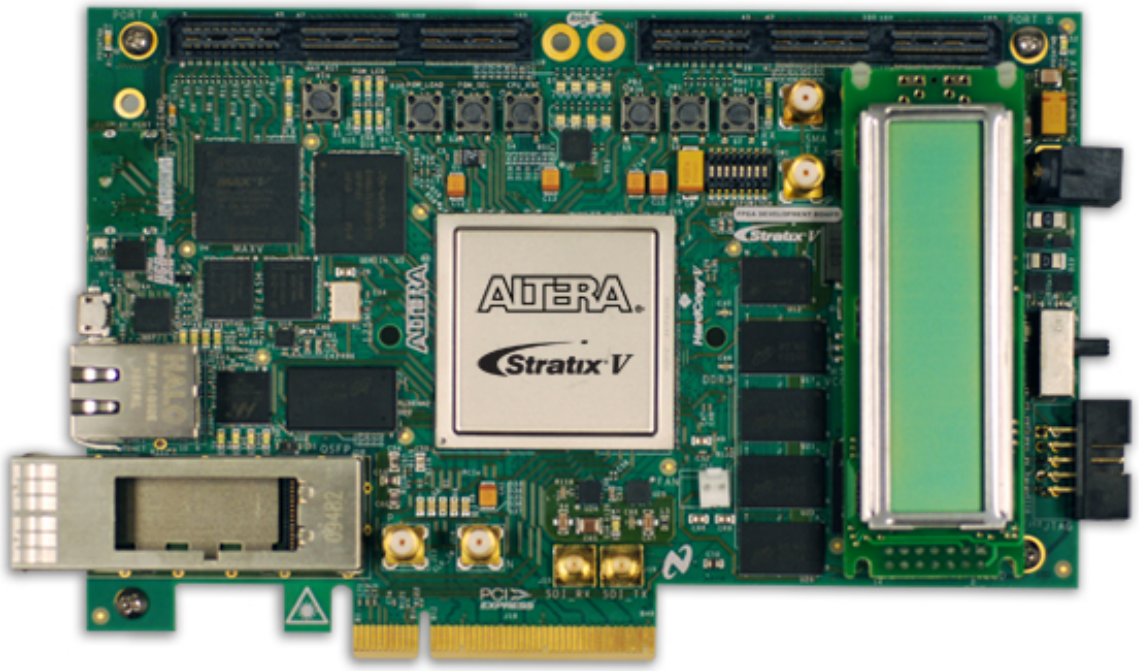
Learn to build a simple processor in a day



Better Software

Better FPGAs/CGRAs

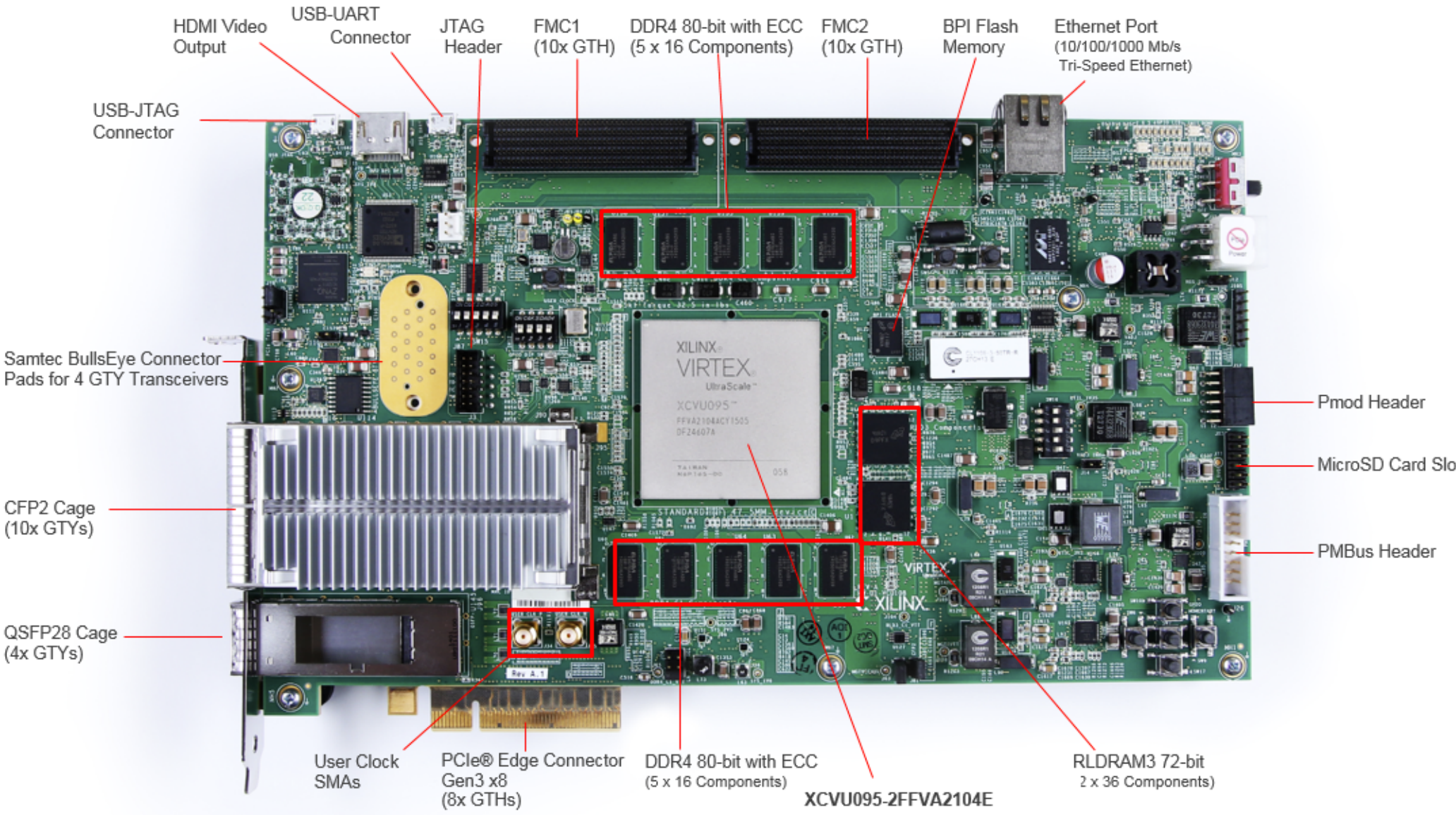
Altera Stratix 5



Microsoft Project Catapult

https://www.microsoft.com/en-us/research/project/project-catapult/

Xilinx Ultrascale+



AWS EC2 F1

https://aws.amazon.com/ec2/instance-types/f1/

1 core, 32 cores, 1680 cores — RISC-V scales up! A 1-core Si-Five HiFive-1, a 2x2x8=32-core GRVI Phalanx in a Digilent Arty / XC7A35T, and a 30x7x8=1680-core GRVI Phalanx in a Xilinx VCU118 / XCVU9P.

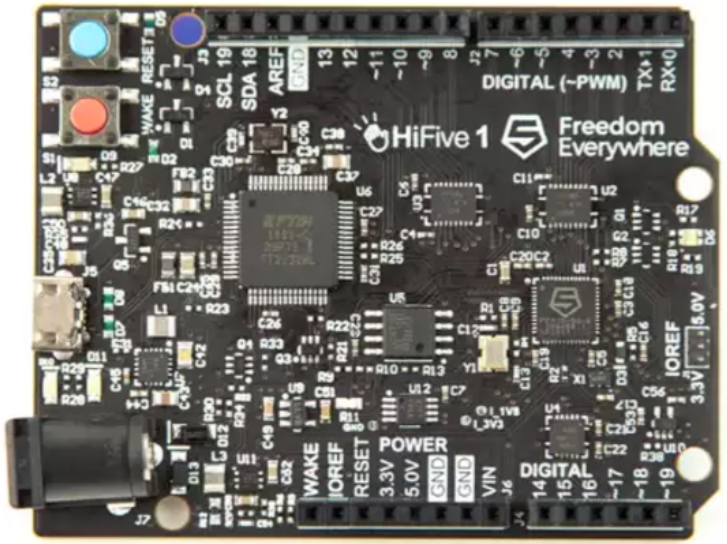
1680 GRVI Phalanx Cores

http://fpga.org/2017/01/12/grvi-phalanx-joins-the-kilocore-club/

Open Hardware?

Home Updates 4 Backers History

"Custom Chips For Under $100K"



01:25 HD

HiFive1: Open-Source, Arduino-Compatible RISC-V Dev Kit

https://www.sifive.com/blog/2016/12/20/custom-chips-for-under-100k/

Agile Development?

Fast design cycle to encourage prototyping

Some Ideas ...

Use modern PL (python, ...)

Metaprogramming and DSLs

Fast place and route

JIT compilation

Better support for test and debug

Logistics

Intro Lectures

Introduction to hardware design and FPGAs

- **Assign1 - Verilog program**

Design a processor using Magma

- **Assign2 - Magma program**

Metaprogramming and DSLs

Tool chains

Project Course

Brainstorm projects

Papers

Panels

Guest speakers

Project

Develop open-source tools for the icestick

Background

No need to know much about hardware

- **Arduino-level a plus (EE40M)**
- **Will teach basics of digital logic design**

Diversity

- **Strong hardware background**
- **Strong software background**
- **Strong HCI: design, prototyping, ease-of-use**

OK to combine with RA projects

Axess is open for enrollment, hopefully can accommodate everyone