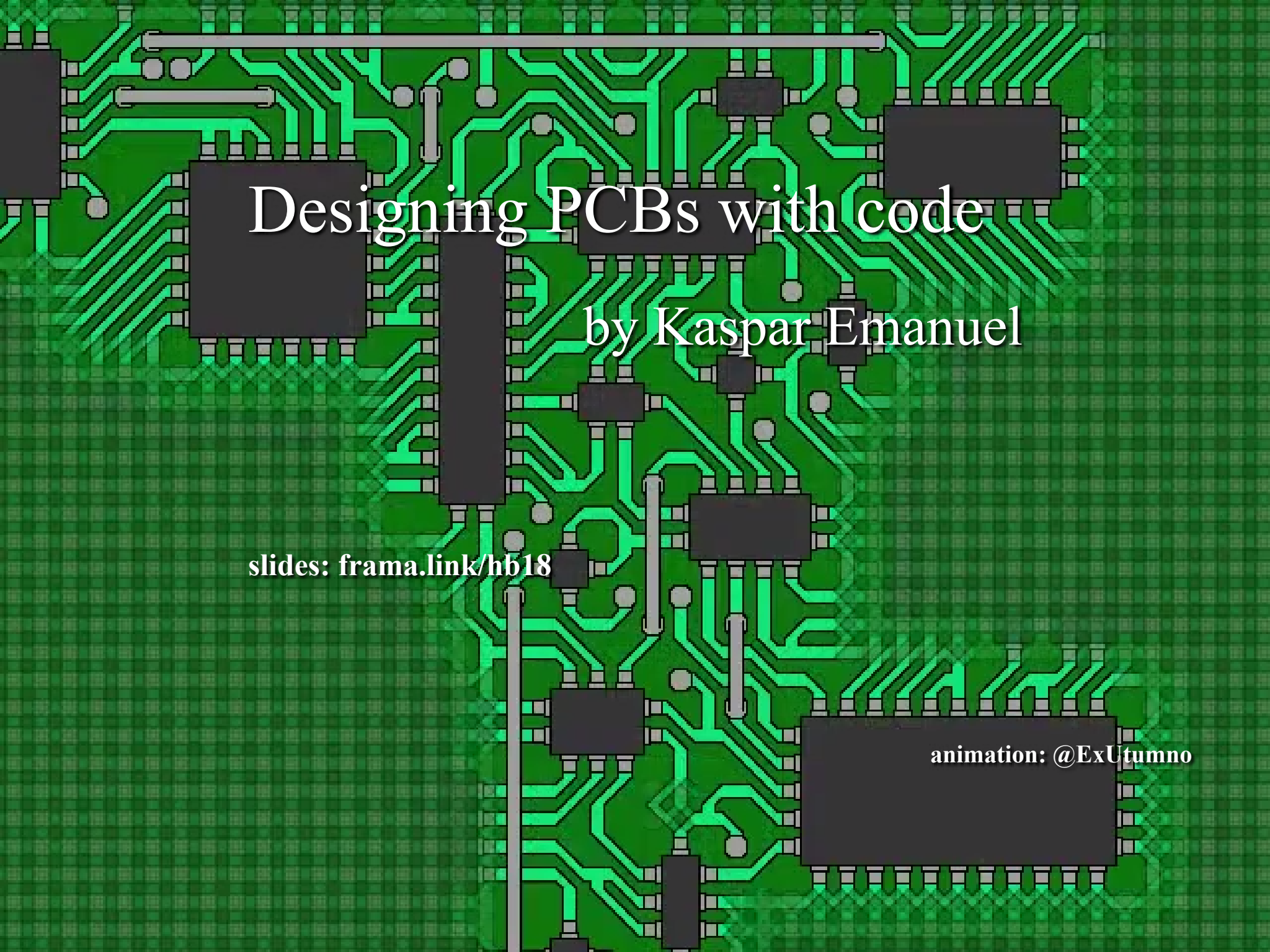
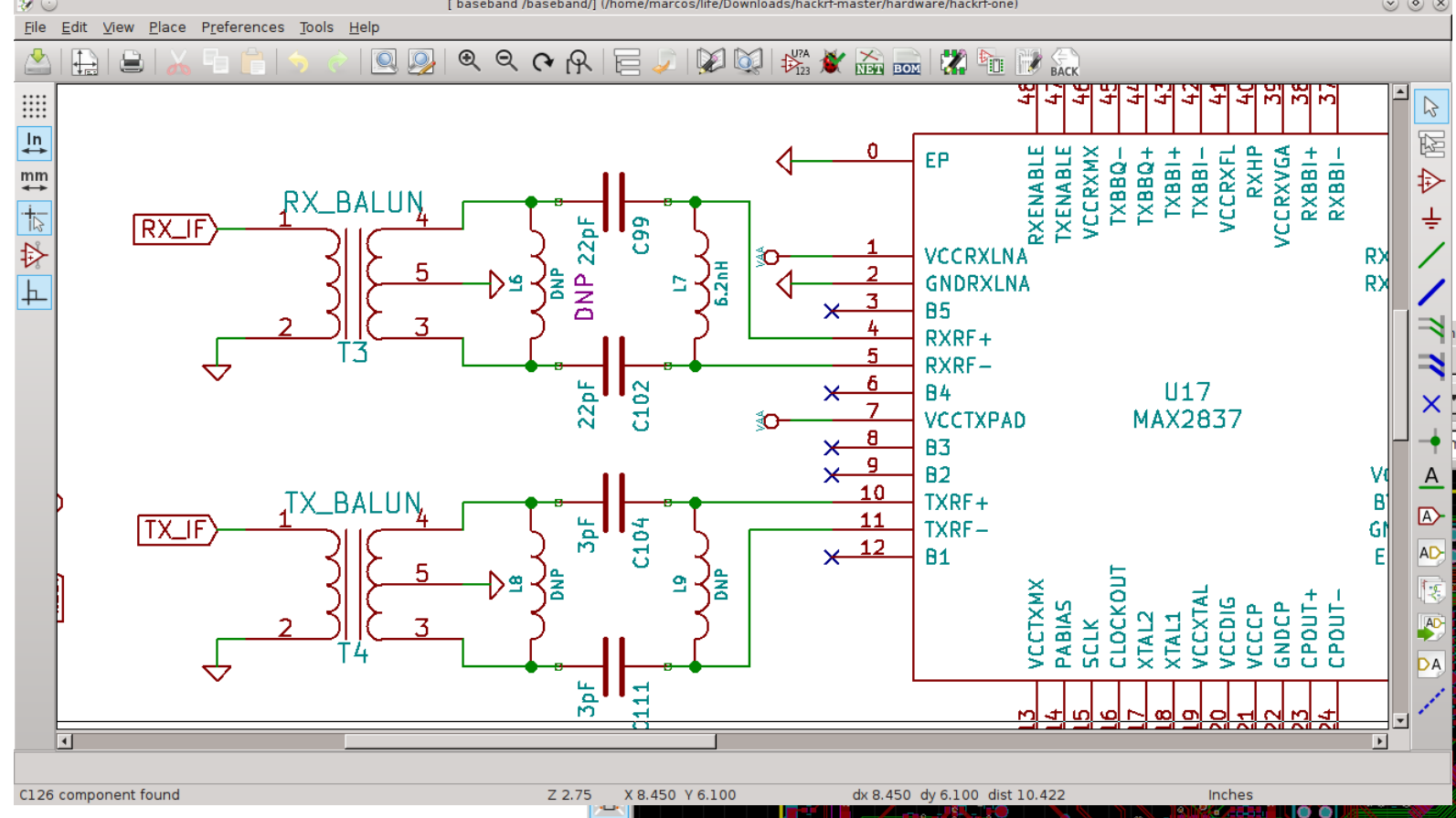
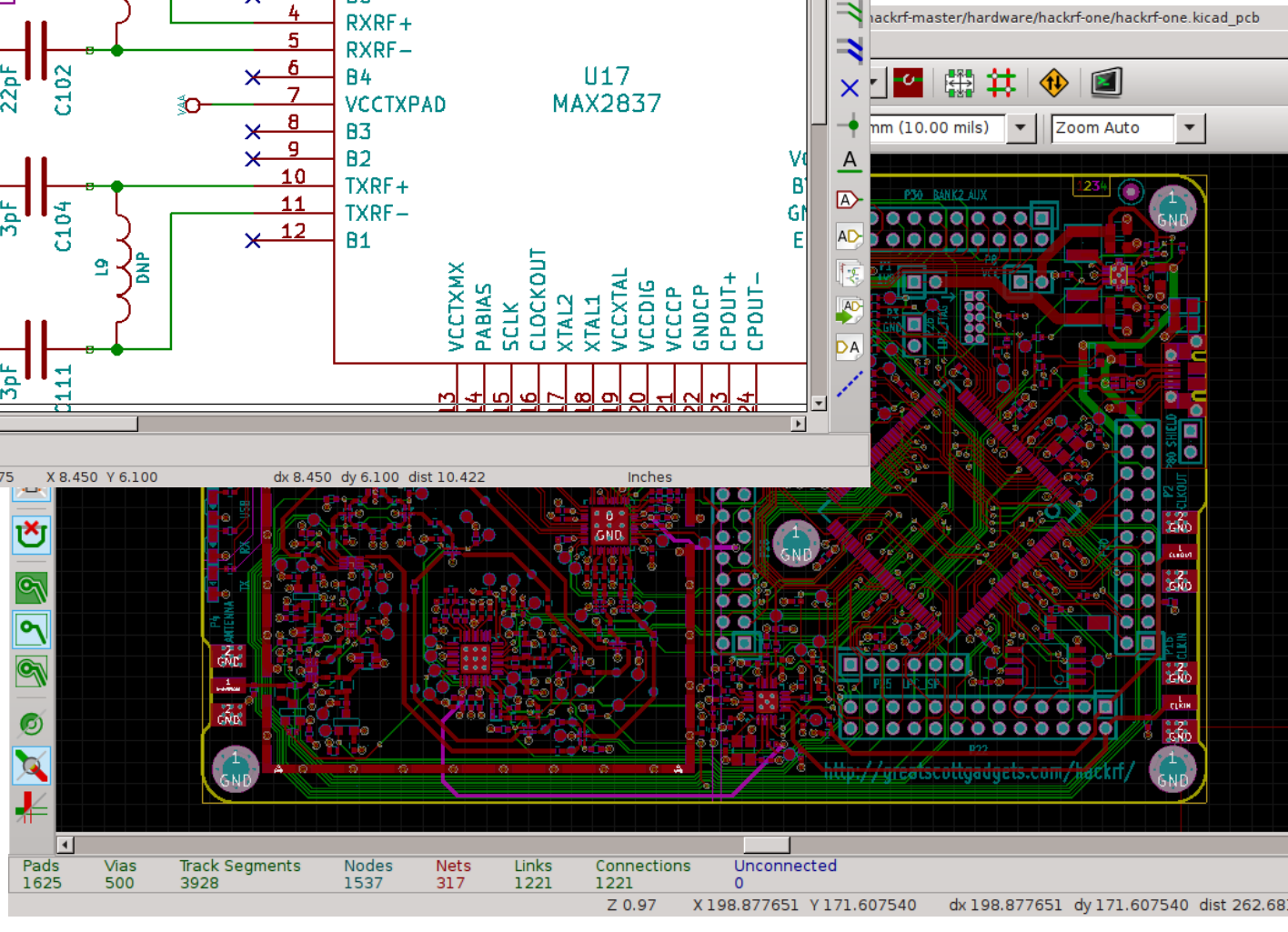
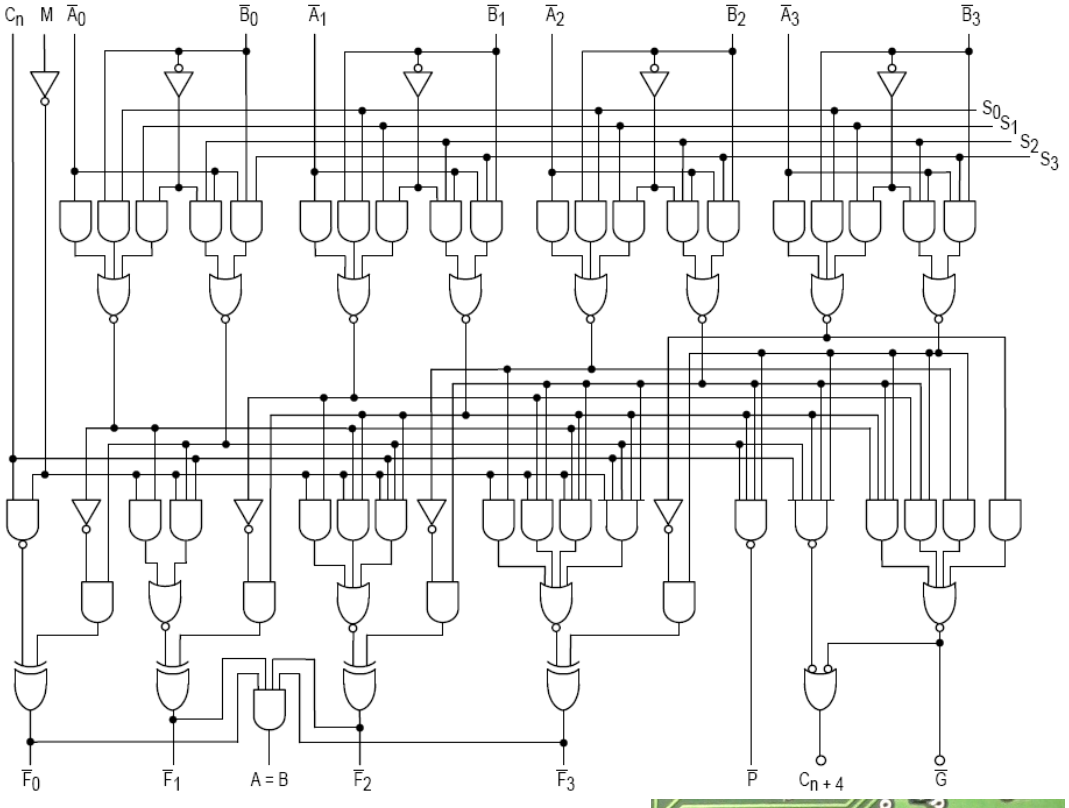

Designing PCBs with code
by Kaspar Emanuel
slides: frama.link/hb18
animation: @ExUtumno

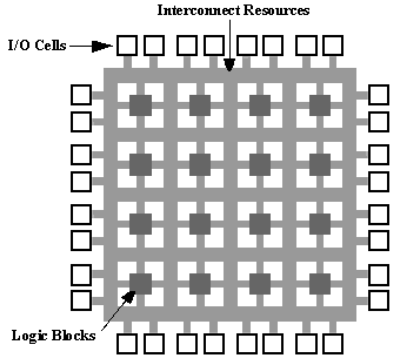
About Me

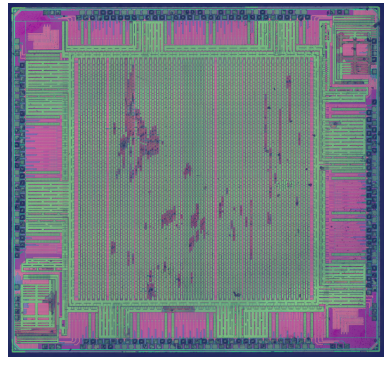
- Freelance electronic engineer and software developer
- Love writing code and love open source (especially electronics)
- github.com/kasbah | hackaday.io/kasbah


[baseband /baseband/] (/home/marcos/life/Downloads/hackrf-master/hardware/hackrf-one)
File Edit View Place Preferences Tools Help
RX_BALUN
RX_IF
T3
L6
DNP
DNP 22pF
C99
L7
6.2nH
22pF
C102
TX_BALUN
TX_IF
T4
L8
DNP
3pF
C104
L9
DNP
3pF
C111
U17
MAX2837
EP
VCCRXLNA
GNDRXLNA
B5
RXRF+
RXRF−
B4
VCCTXPAD
B3
B2
TXRF+
TXRF−
B1
RXENABLE
TXENABLE
VCCRXMX
TXBBQ−
TXBBQ+
TXBBI+
TXBBI−
VCCRXFL
RXHP
VCCRXVGA
RXBBI+
RXBBI−
VCCTXMX
PABIAS
SCLK
CLOCKOUT
XTAL2
XTAL1
VCCXTAL
VCCDIG
VCCCP
GNDCP
CPOUT+
CPOUT−
C126 component found
Z 2.75
X 8.450 Y 6.100
dx 8.450 dy 6.100 dist 10.422
Inches

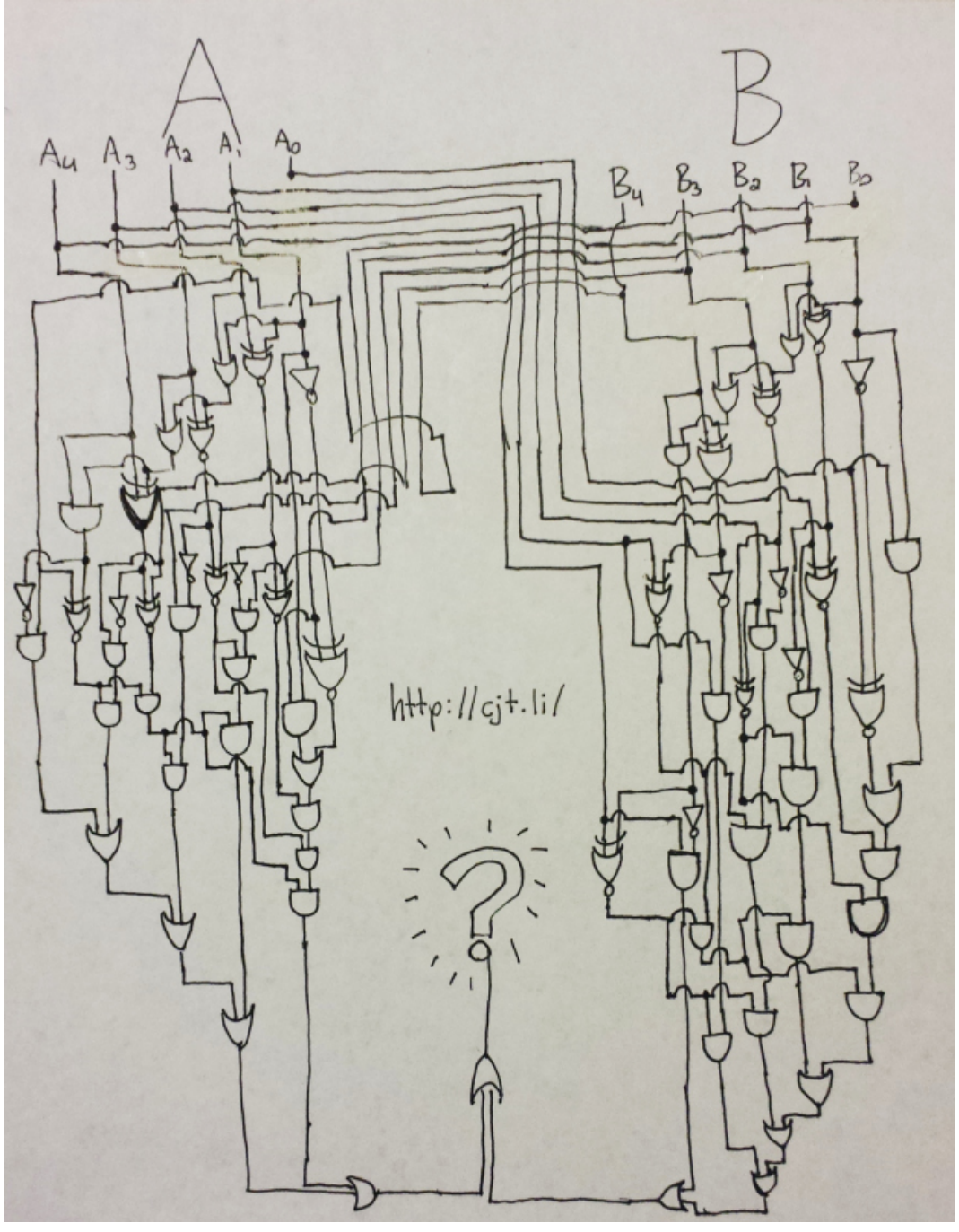

hackrf-master/hardware/hackrf-one/hackrf-one.kicad_pcb
mm (10.00 mils)
Zoom Auto
http://greatscottgadgets.com/hackrf/
Pads 1625
Vias 500
Track Segments 3928
Nodes 1537
Nets 317
Links 1221
Connections 1221
Unconnected 0
Z 0.97
X 198.877651 Y 171.607540
dx 198.877651 dy 171.607540 dist 262.68


Cn
M
A0
B0
A1
B1
A2
B2
A3
B3
S0
S1
S2
S3
F0
F1
A = B
F2
F3
P
Cn + 4
G


Interconnect Resources
I/O Cells
Logic Blocks

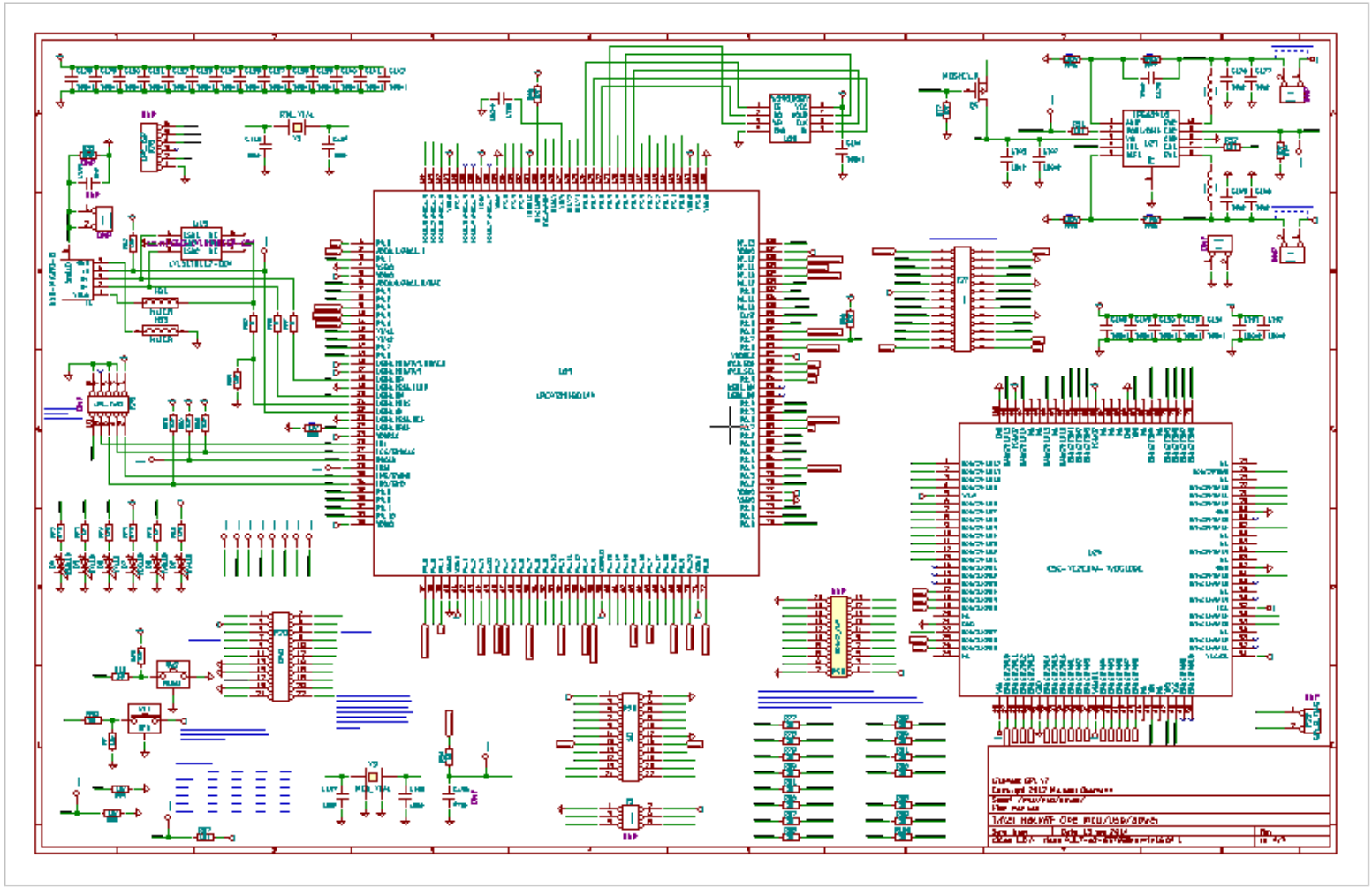



XILINX
SPARTAN


A
B
A4 A3 A2 A1 A0
B4 B3 B2 B1 B0
http://cjt.li/
?

```
-- (this is a VHDL comment)

-- import std_logic from the IEEE library
library IEEE;
use IEEE.std_logic_1164.all;

-- this is the entity
entity ANDGATE is
  port (
    I1 : in std_logic;
    I2 : in std_logic;
    O  : out std_logic);
end entity ANDGATE;

-- this is the architecture
architecture RTL of ANDGATE is
begin
  O <= I1 and I2;
end architecture RTL;
```

Preferences Tools Help



Z 0.55 X 229.87 Y 134.00 dx 229.87 dy 134.00 dist 266.08 mm

Can we use HDLs for analog circuits?

```
//Verilog-A (1993) and Verilog-AMS (2000)
`include "constants.vams"
`include "disciplines.vams"
// Simple ADC model
module adc_simple(clk, dout, vref, vin);
    // Parameters
    parameter integer bits = 4 from[1:24]; // Number of bits
    parameter integer td = 1 from[0:inf);  // Processing delay of the ADC
    // Define input/output
    input clk, vin, vref;
    output [bits-1:0] dout;
    electrical vref, vin;
    logic clk;
    reg [bits-1:0] dout;
    // Internal variables
    real ref, sample;
    integer i;
  ...
```

```
-- VHDL-AMS
-- IEEE 1076.1-1999

library IEEE;
use IEEE.math_real.all;
use IEEE.electrical_systems.all;

entity DIODE is
   generic (iss : current := 1.0e-14;  -- Saturation current
            af  : real    := 1.0;      -- Flicker noise coefficient
            kf  : real    := 0.0);     -- Flicker noise exponent
   port (terminal anode, cathode : electrical);
end entity DIODE;

architecture IDEAL of DIODE is
  quantity v across i through anode to cathode;
  constant vt : voltage := 0.0258;     -- Thermal voltage at 300 K
begin

  i == iss * (exp(v/vt) - 1.0);

end architecture IDEAL;
```

- Also: "Circuit Description Language" (Taku Noda, IPST 1999)

SPICE

```
AMP2.CIR - CASCADED OPAMPS
*
VS    1    0    AC    1    SIN(0 1 10KHZ)
*
R1    1    2    5K
R2    2    3    10K
XOP1    0 2    3    OPAMP1
R3    4    0    10K
R4    4    5    10K
XOP2    3 4    5    OPAMP1
*
* SINGLE-POLE OPERATIONAL AMPLIFIER MACRO-MODEL
* connections:      non-inverting input
*                   |   inverting input
*                   |   |   output
*                   |   |   |
.SUBCKT OPAMP1      1   2   6
* INPUT IMPEDANCE
RIN     1          2          10MEG
* DC GAIN (100K) AND POLE 1 (100HZ)
EP1    3 0    1 2    100K
RP1    3    4    1K
CP1    4    0    1.5915UF
* OUTPUT BUFFER AND RESISTANCE
EOUT    5 0    4 0    1
ROUT    5    6    10
.ENDS
*
* ANALYSIS
.TRAN    0.01MS  0.2MS
* VIEW RESULTS
.PLOT TRAN V(1) V(5)
```

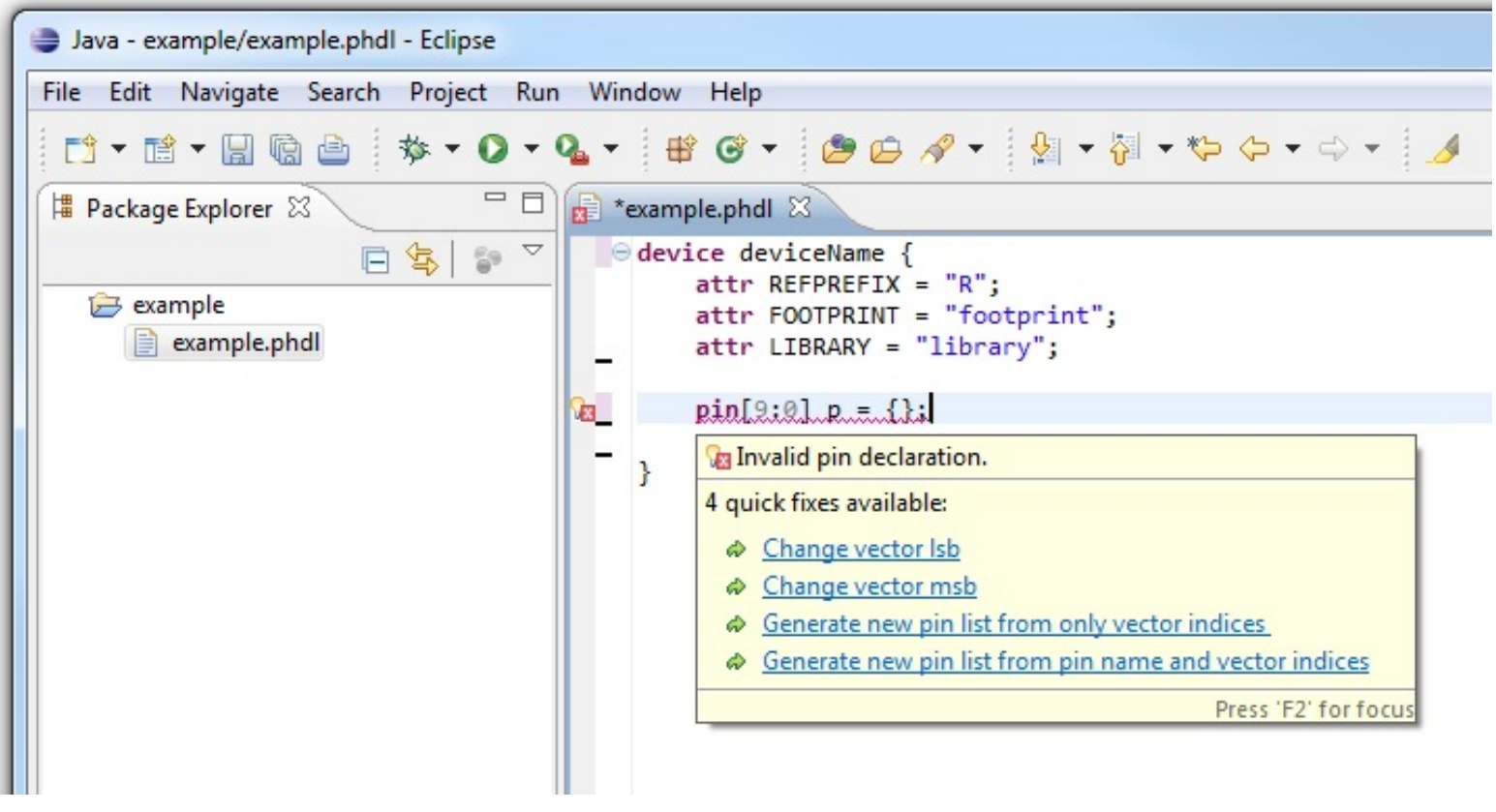
What do people want when they want to make hardware more like software?

1. Fast build/test iteration cycles
2. Use programming constructs and tools for a faster/better design process
3. Modularity and re-usability

```
/* PHDL - PCB hardware description language (Brent Nelson, 2011)*/


// A surface mount resistor
device resistor {
  attr REFPREFIX = "R";
  attr FOOTPRINT = "R0805";
  attr LIBRARY = "rcl-smd";
  attr VALUE = "1k";
  pin a = {1};
  pin b = {2};
}

design top {
  net vcc, vout, gnd;

  inst r1 of resistor {
    a = vcc;
    b = vout;
  }

  inst r2 of resistor {
    a = vout;
    b = gnd;
  }
}
```

PHDL

- From 2011
- New language
- Java based compiler & Eclipse IDE plugin
- Outputs Eagle and Orcad netlists natively



SKiDL

- Created in 2016
- A library/language in Python
- Outputs KiCad netlists

```
from skidl import *

gnd = Net('gnd')  # Ground reference.
vin = Net('vin')   # Input voltage to the divider.
vout = Net('vout')  # Output voltage from the divider.
r1, r2 = 2 * Part('device', 'R', TEMPLATE)  # Create two resistors.
r1.value, r1.footprint = '1K',  'Resistors_SMD:R_0805'  # Set resistor values
r2.value, r2.footprint = '1K', 'Resistors_SMD:R_0805'  # and footprints.
r1[1] += vin      # Connect the input to the first resistor.
r2[2] += gnd      # Connect the second resistor to ground.
vout += r1[2], r2[1]  # Output comes from the connection of the two resistors.

generate_netlist()
```

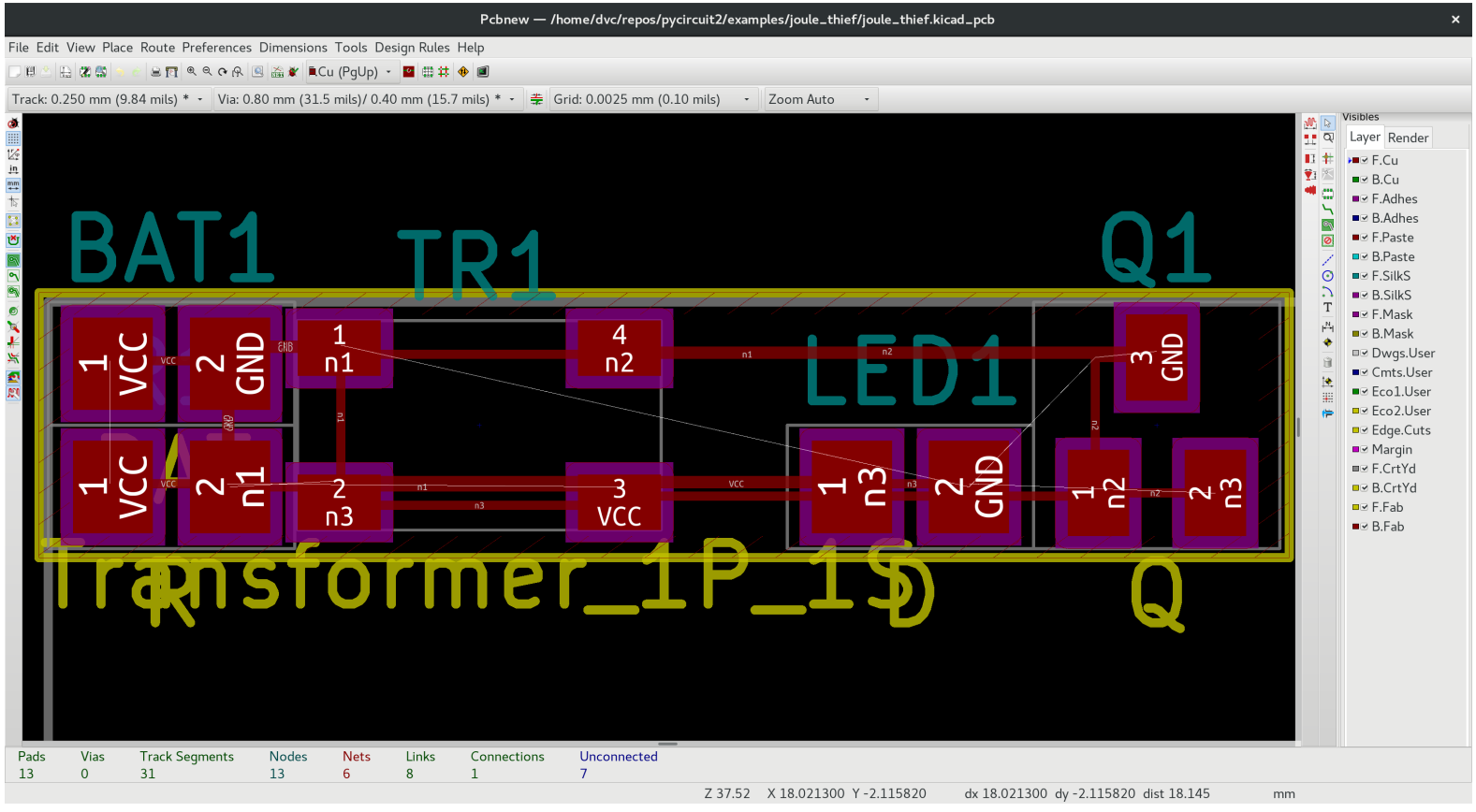
```
from skidl import *

# Define the voltage divider module. The @subcircuit decorator
# handles some skidl housekeeping that needs to be done.
@subcircuit
def vdiv(inp, outp):
    """Divide inp voltage by 3 and place it on outp net."""
    rup = Part('device', 'R', value='1K', footprint='Resistors_SMD:R_0805')
    rlo = Part('device','R', value='500', footprint='Resistors_SMD:R_0805')
    rup[1,2] += inp, outp
    rlo[1,2] += outp, gnd

gnd = Net('GND')         # GLobal ground net.
input_net = Net('IN')    # Net with the voltage to be divided.
output_net = Net('OUT')  # Net with the divided voltage.

# Instantiate the voltage divider and connect it to the input & output nets.
vdiv(input_net, output_net)

generate_netlist()
```

pycircuit

- Created in 2017
- A library/language in Python
- Outputs KiCad PCB files and Yosys netlists

```
@circuit('Voltage Divider', 'gnd', None, 'vin', 'vout')
def voltage_divider(self, gnd, vin, vout):
    Inst('R', '1k 0805')['~', '~'] = vin, vout
    Inst('R', '1k 0805')['~', '~'] = vout, gnd

@circuit('Top')
def top(self):
    vin, vout, gnd = nets('vin vout gnd')
    SubInst(voltage_divider())['vin', 'vout', 'gnd'] = vin, vout, gnd
```

```
# pycircuit for place and route

def place(filein, fileout):
    placer = Placer()
    placer.place(filein, fileout)

def route(filein, fileout):
    router = Router()
    router.route(filein, fileout)

Builder(joule_thief.top(), oshpark_4layer,
    place=place, route=route).build()
```

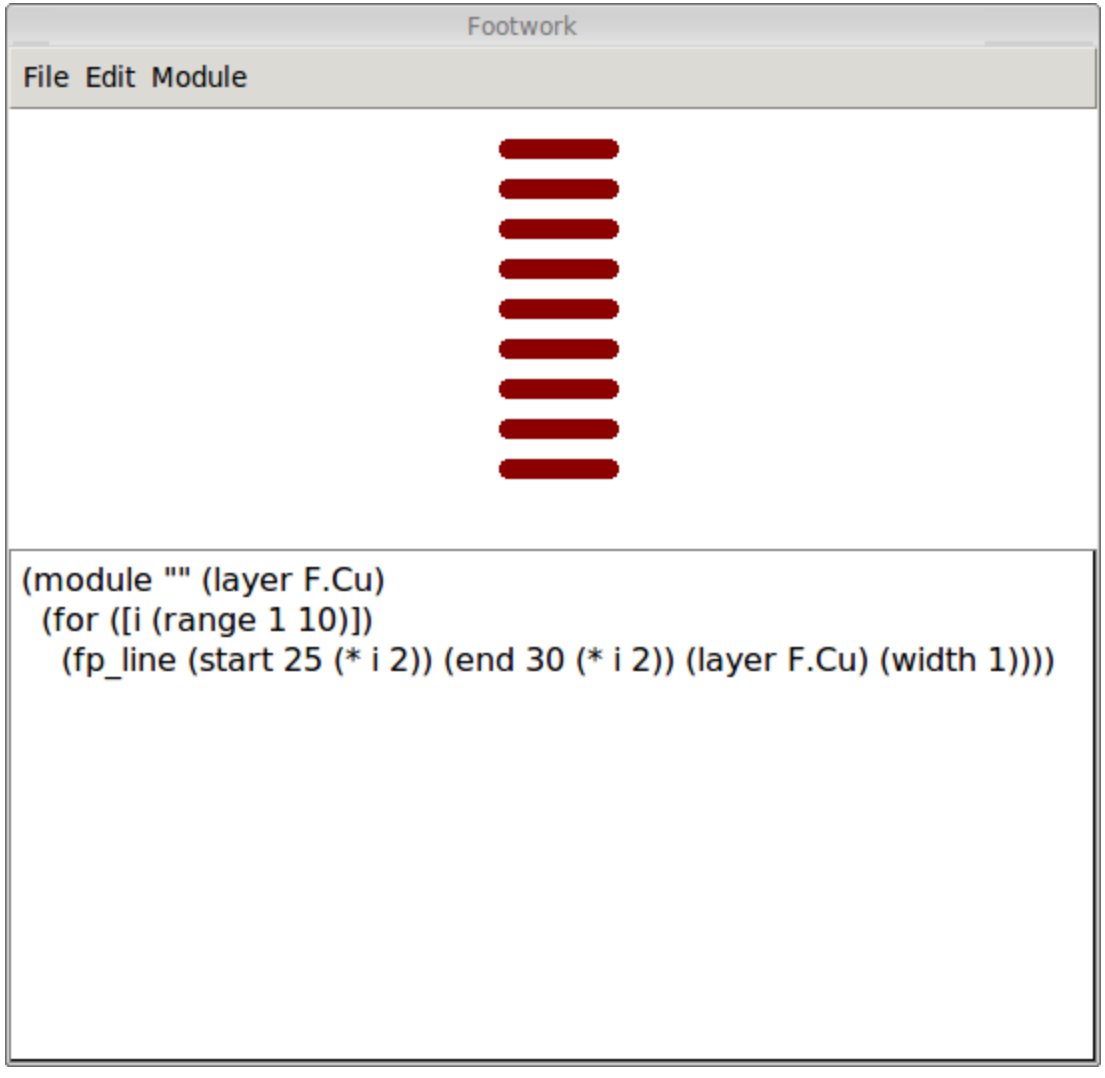


Footprints

- KicadModTree: a Python DSL for KiCad footprints
- qeda: A Coffeescript/Javascript utility for schematic symbol and footprint generation

```
name: D-MMA
description: MiniMelf Diode (MMA)
keywords: Diode
schematic:
  symbol: diode
housing:
  pattern: MELF
  options: polarized
  bodyDiameter: 1.3-1.4
  bodyLength: 3.4-3.6
  leadLength: 0.65-0.85
```

Footwork


Footwork
File Edit Module
(module "" (layer F.Cu)
(for ([i (range 1 10)])
(fp_line (start 25 (* i 2)) (end 30 (* i 2)) (layer F.Cu) (width 1))))

```
// replicad

var {Resistor, Power, Ground, Output} = require('replicad')

function resistorDivider(value1, value2) {
  var r1 = new Resistor(value1)
  var r2 = new Resistor(value2)

  var vcc = new Power()
  var gnd = new Ground()

  var vout = new Output()

  var circuit = new Circuit()
  circuit.chain(vcc, r1, vout, r2, gnd);
  return circuit
}


var circuit = resistorDivider('1k', '500 ohm')
```

- A domain specific language in Javascript
- Work in progress!

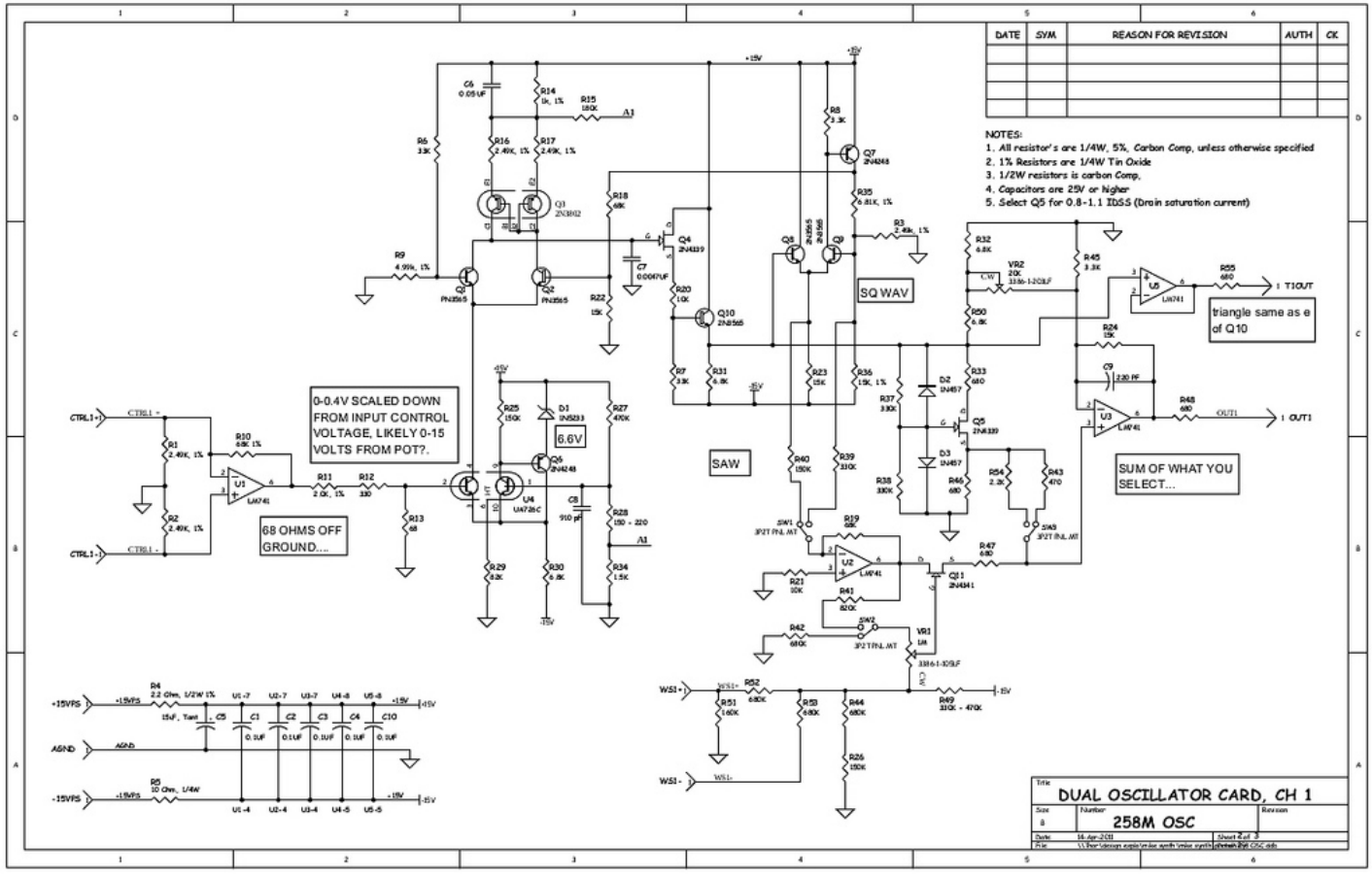
replicad

- Goals:
  - Initial goal is to offer netlist/schematic entry only
  - Make it easier to design and reason about circuits
  - Confirm Atwood's Law
  - Static analysis to make it very hard to create bugs
  - Encourage design re-use
- Implementation details:
  - Variable names are used as schematic references
  - Circuit objects are explicitly modified by adding connections
  - Function arguments are parameters not inputs/outputs

PCB hardware description languages

- Pros:
  - Define once and re-use
  - Use for-loops, slice notation, etc.
- Issues:
  - It's hard to visualize
  - Circuit definition can still be very tedious
  - Debugging could easily become a nightmare

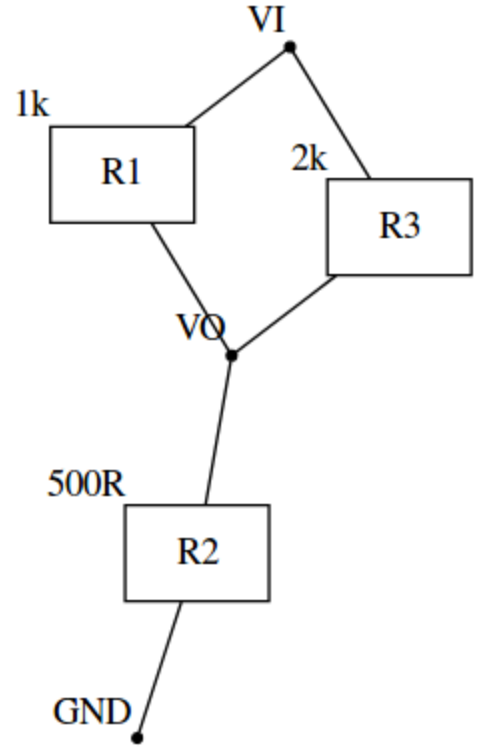
Visualization



SKiDL

```
vin += r3[1]
r3[2] += vout
vout += r1[2], r2[1]  # Output com

generate_graph()
```

Out[1]:

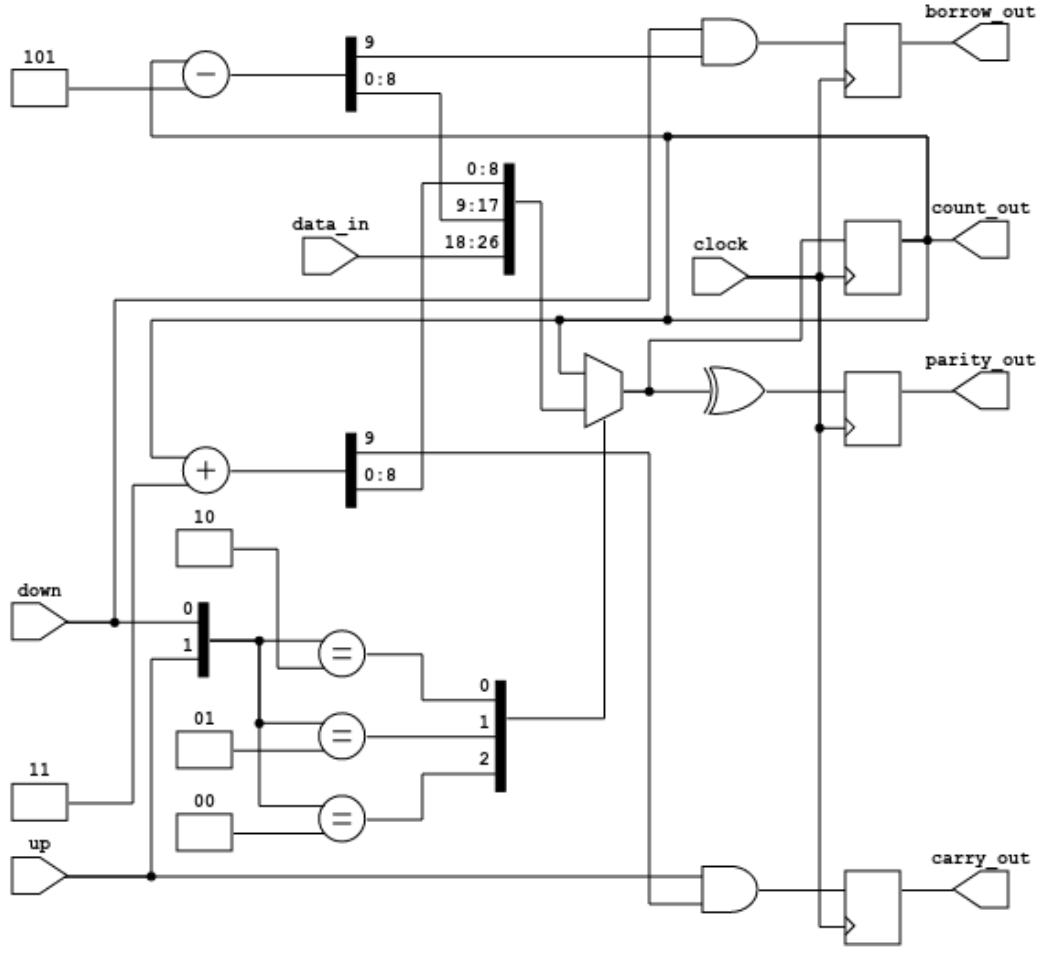


netlistsvg

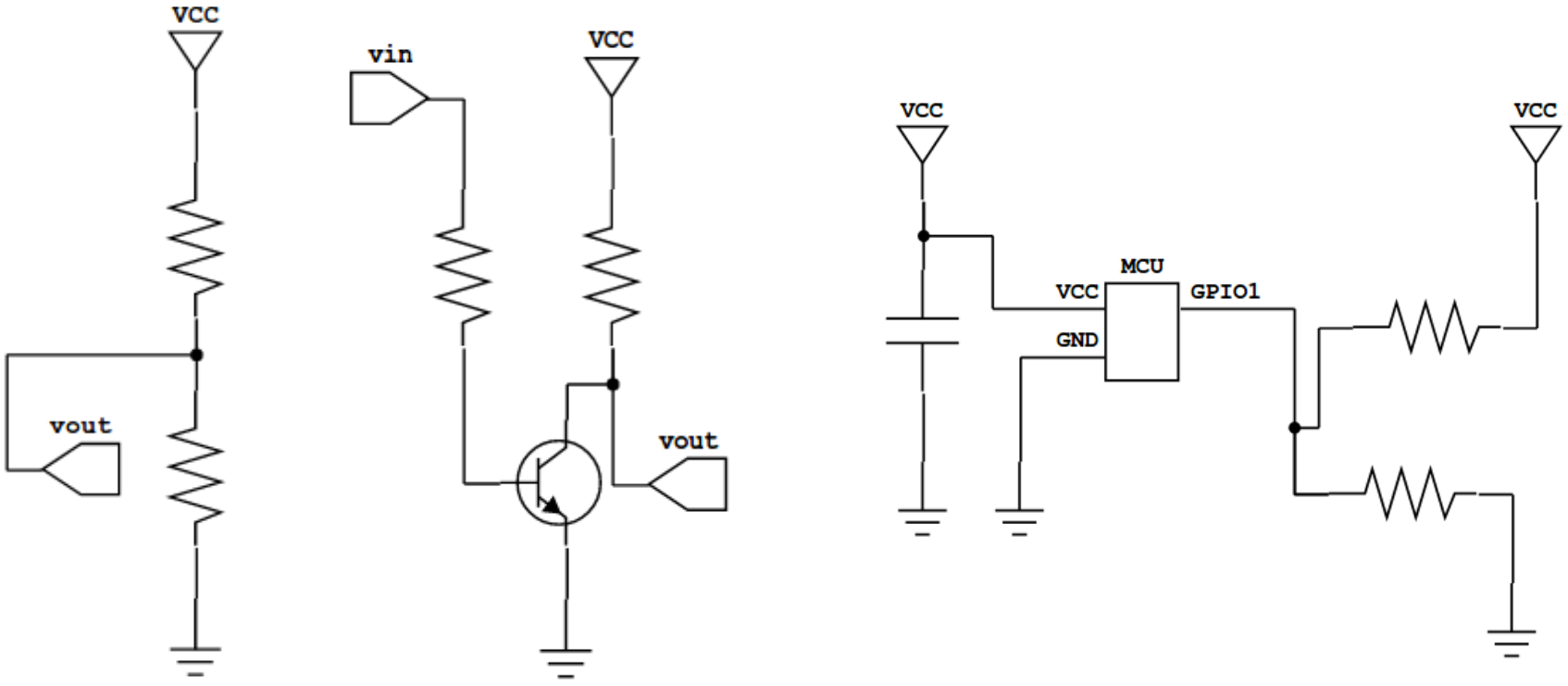
draws an SVG schematic from a yosys JSON netlist. It uses elkjs for layout.

You can see an online demo here

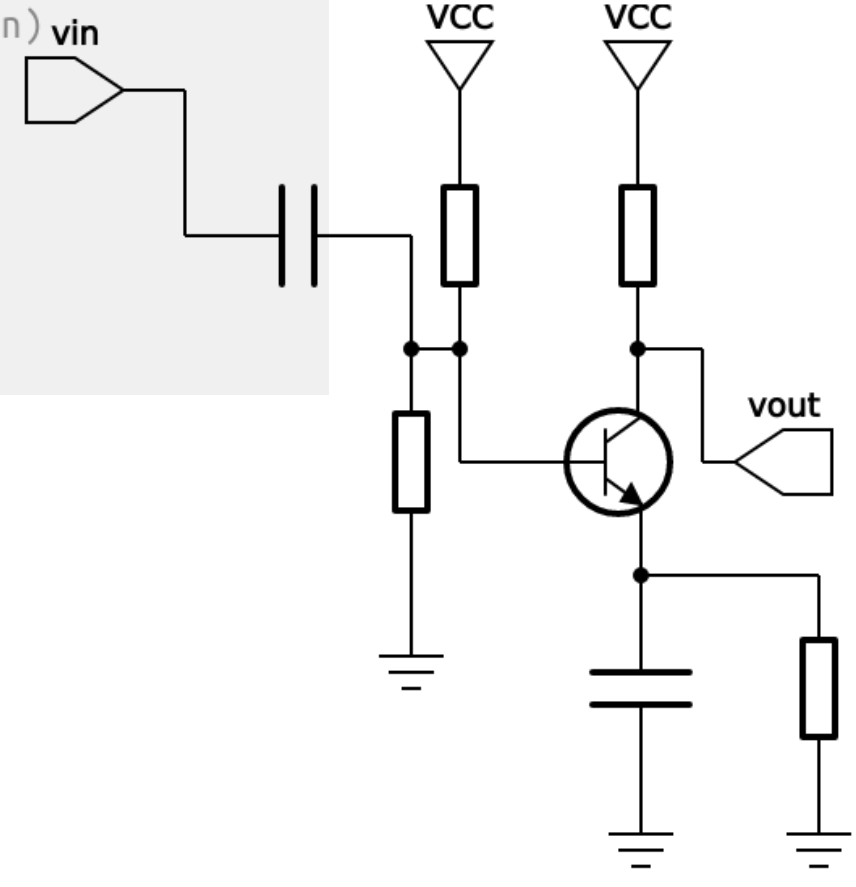
▸ JSON Source



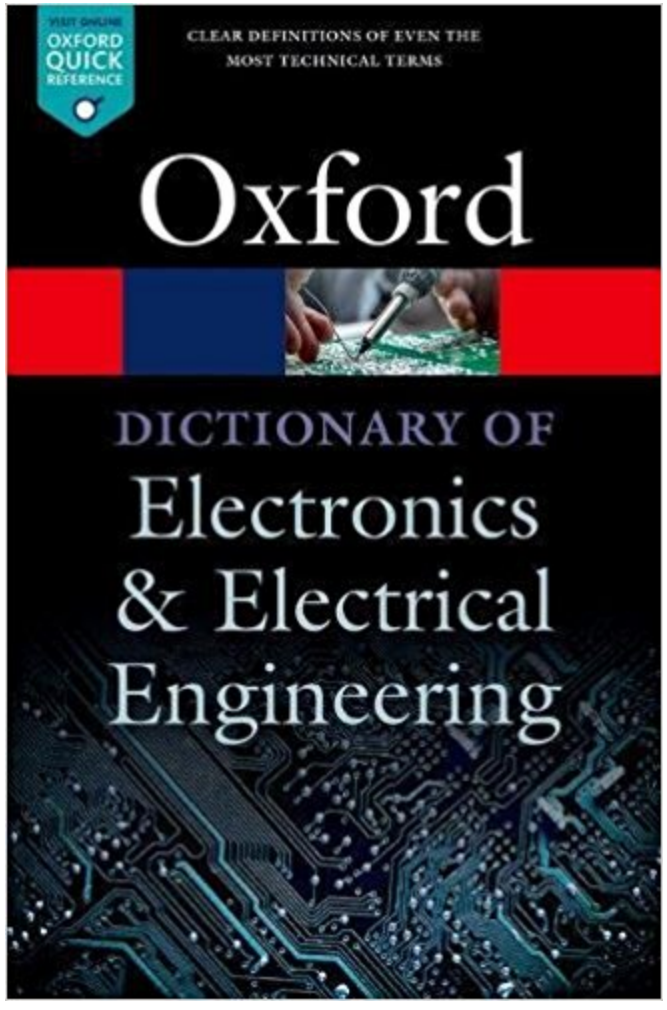
Using "analog" skin with netlistsvg



```
def common_emitter_amplifer(self, gnd, vcc, vin, vout):
    nb, ne = nets('nb ne')
    Inst('Q', 'npn sot23')['B', 'C', 'E'] = nb, vout, ne

    # Current limiting resistor
    Inst('R', '1.2k')['~', '~'] = vcc, vout

    # Thermal stabilization (leads to a gain reduction)
    Inst('R', '220')['~', '~'] = ne, gnd
    # Shorts Re for AC signal (increases gain)
    Inst('C', '10uF')['~', '~'] = ne, gnd

    # Biasing resistors
    Inst('R', '20k')['~', '~'] = vcc, nb
    Inst('R', '3.6k')['~', '~'] = nb, gnd
    # Decoupling capacitor
    Inst('C', '10uF')['~', '~'] = vin, nb
```



The language of electronics



Electro Grammar (demo)

```
> var {parse} = require('electro-grammar')

> parse('100nF 0603 C0G 10% 25V')
{ type: 'capacitor',
  capacitance: 1e-7,
  size: '0603',
  characteristic: 'C0G',
  tolerance: 10,
  voltage_rating: 25 }

> parse('1k 0805 5% 125mW')
{ type: 'resistor',
  resistance: 1000,
  size: '0805',
  tolerance: 5,
  power_rating: 0.125 }

> parse('green led 1206')
{ type: 'led', color: 'green', size: '1206' }
```

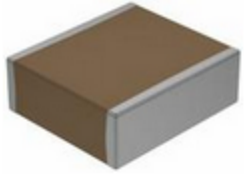
https://bom-builder.kitspace.org

Open Save Undo Redo Delete Remove All

References	Qty	Description	« » Part Number	...	Digikey	Mouser	RS
R1	1	1k 0603	Panasonic ERJ-3EKF1001V				
C1	1	1uF 0805	Yageo CC0805KKX7R8BB105				
U1	1		Texas Instruments				

Selected

Yageo CC0805KKX7R8BB105 1/2



Multilayer Ceramic Capacitor, CC Series, 1 - F, -10%, X7R, 25 V, 0805 [2012 Metric]

Datasheet

Capacitance	1.00 µF
Case/Package	0805
Dielectric Characteristic	X7R
Capacitance Tolerance	±10%

Newark

Powered by Octopart

...

Electro Grammar v1

- JavaScript only
- Capacitors, resistors and LEDs (SMD only)
- Lax parser only (any-order, ignores invalid input)

Electro Grammar v2

- Work in progress!
- Uses Antlr4: JavaScript, Python, Java, C (& C++), Go
- Capacitors, resistors, LEDs, diodes, transistors (SMD & through-hole)
- Strict and lax parser

Should you use it?

- PHDL: alpha (and some bitrot since 2012)
- SKiDL: alpha (probably your best right now)
- pycircuit: experimental (work stopped?)
- replicad: vaporware?

What do people want when they want to make hardware more like software?

1. Fast build/test iteration cycles
2. Use programming constructs and tools for a faster/better design process
3. Modularity and re-usability

Kitspace.org

Share and build open hardware electronics

Neococat

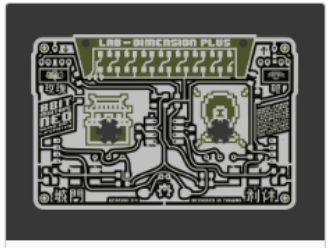
github.com / kimitobo

Board modification of 8Bitmixtape NEO synthesizer

NextLevelEdition

github.com / 8BitMixtape

8-BitMixtape NextLevelEdition



8BitmixtapeNEO_ShenzhenRe...

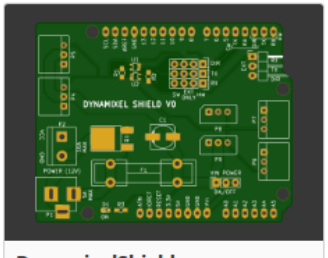
github.com / 8BitMixtape

The key feature of the new 8Bit Mixtape is the easiness of uploading new codes using ...

sympetrum-v2

github.com / borgel

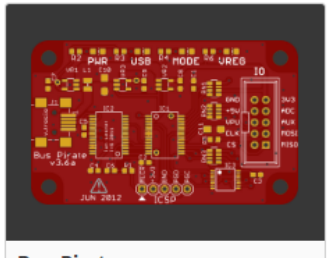
A communicative electronic conference badge in the shape of a dragonfly



DynamixelShield

github.com / descampsa

Arduino shield to connect Dynamixel servomotors



Bus_Pirate

github.com / kitspace-forks

The Bus Pirate is an open source hacker multi-tool that talks to electronic stuff.

- Buy components from Digikey, Mouser, RS, Newark, Farnell with a single click

- Free and open source at github.com/monostable/kitspace

- Add your own open source hardware projects!

Questions?

- github.com/kasbah | hackaday.io/kasbah
- gitter.im/monostable/electro-grammar

Thanks to:

- Brent Nelson and co (PHDL),
- Dave Vandenbout (SKiDL),
- David Craven (pycircuit),
- Neil Turley (netlistsvg)
- All contributors to:
  - Graphviz
  - ELK and ELKJS
  - Nearley parser generator
  - Antlr4 parser generator