Villa Ritter
Biel/Bienne

2024

COSIN

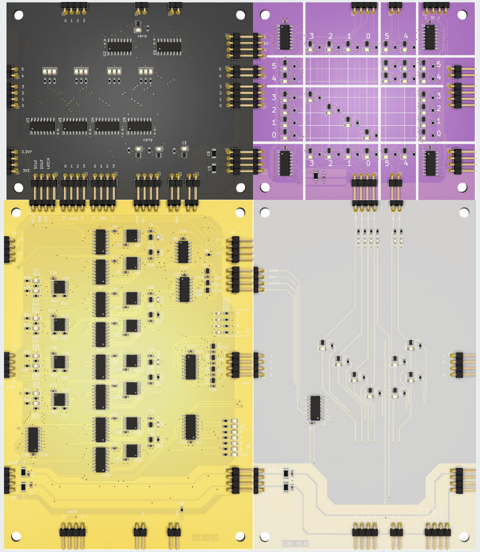
28. - 30. Juni 2024

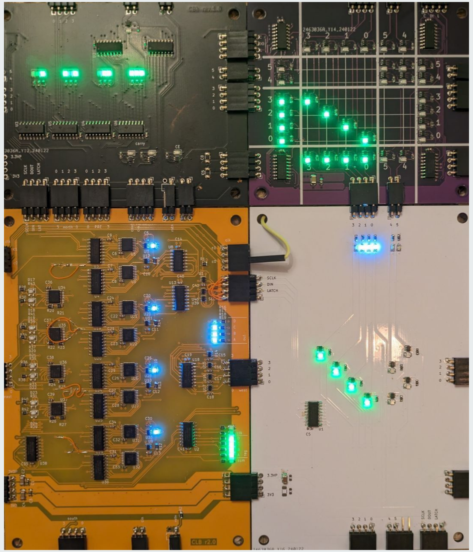
How to build your own FPGA

with 7400-Logic

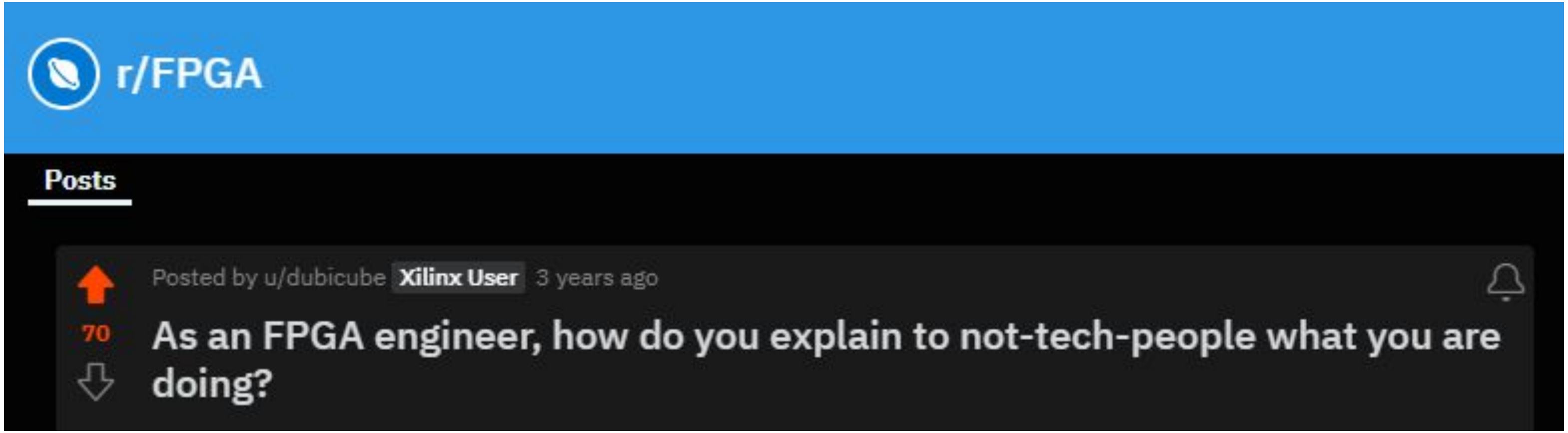
Simon Burkhardt

mnemocron.github.io





What is an FPGA?



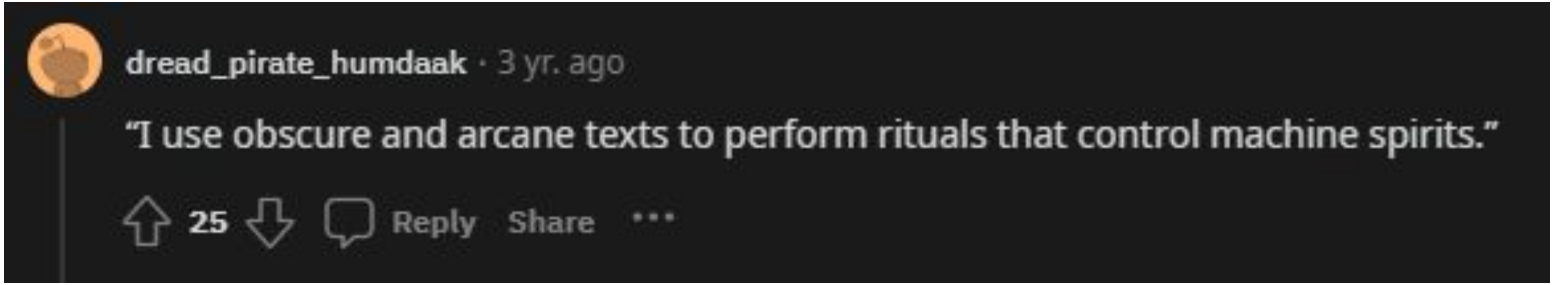
r/FPGA

Posts

Posted by u/dubicube Xilinx User 3 years ago

70

As an FPGA engineer, how do you explain to not-tech-people what you are doing?

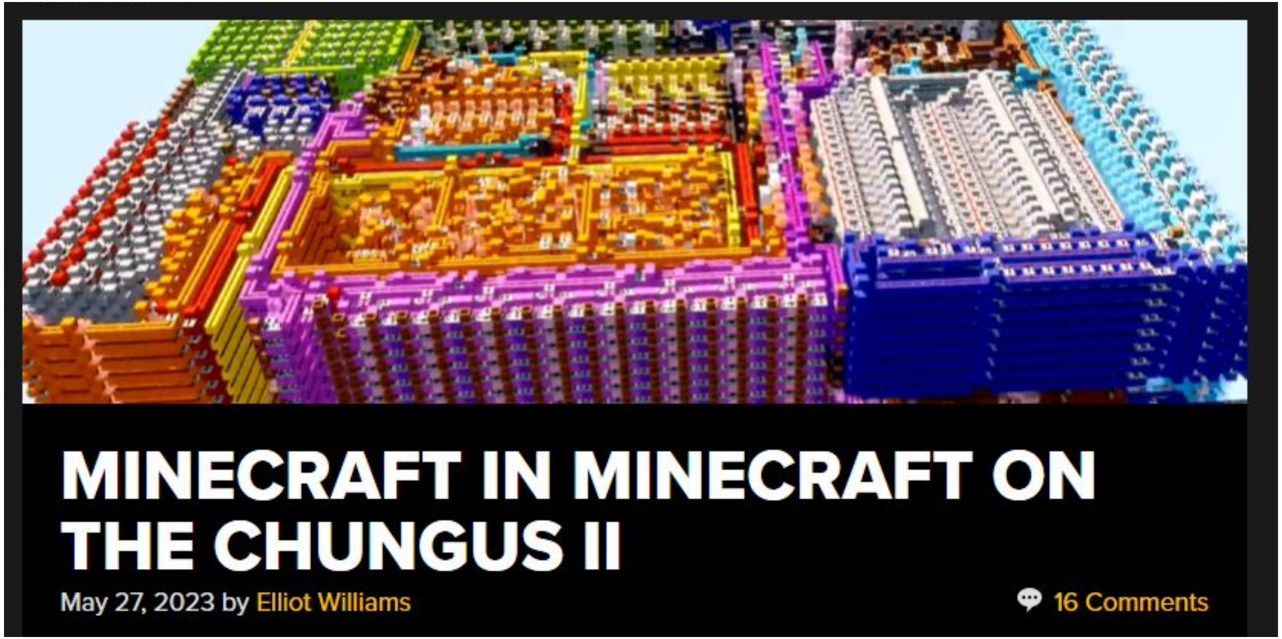


dread_pirate_humdaak · 3 yr. ago

"I use obscure and arcane texts to perform rituals that control machine spirits."

Why?

- Acquire an understanding of low-level FPGA hardware
- Teach people about FPGAs
- It is fun (?)



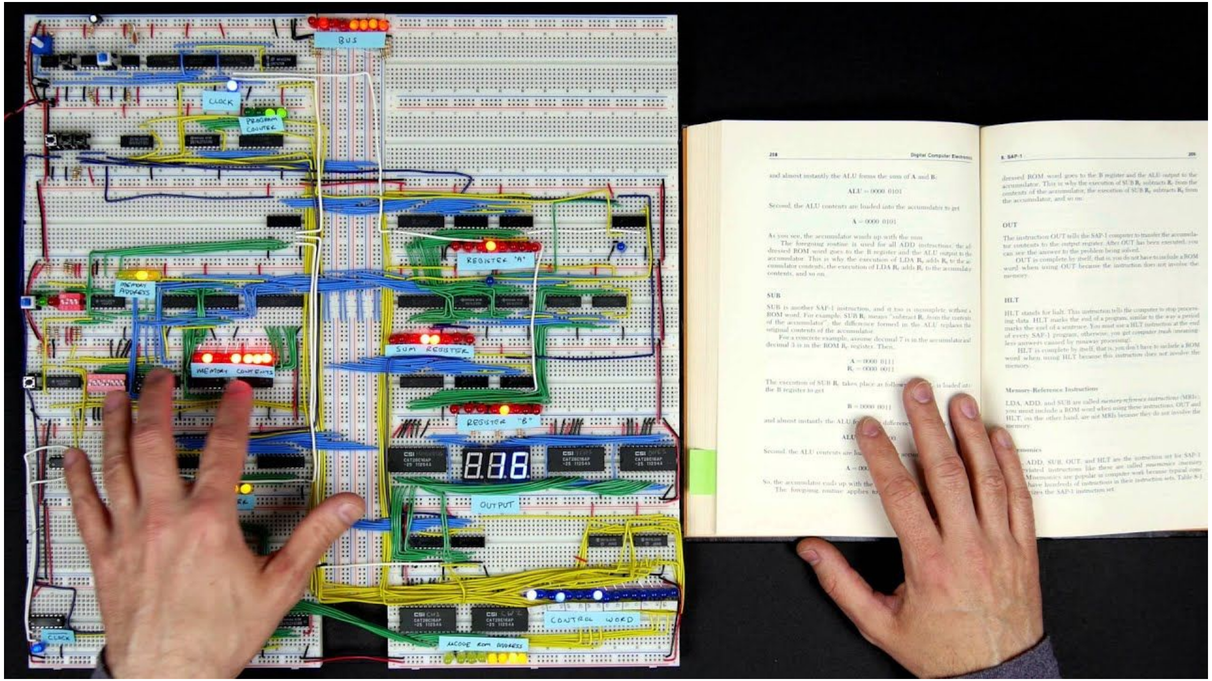
MINECRAFT IN MINECRAFT ON THE CHUNGUS II

May 27, 2023 by Elliot Williams

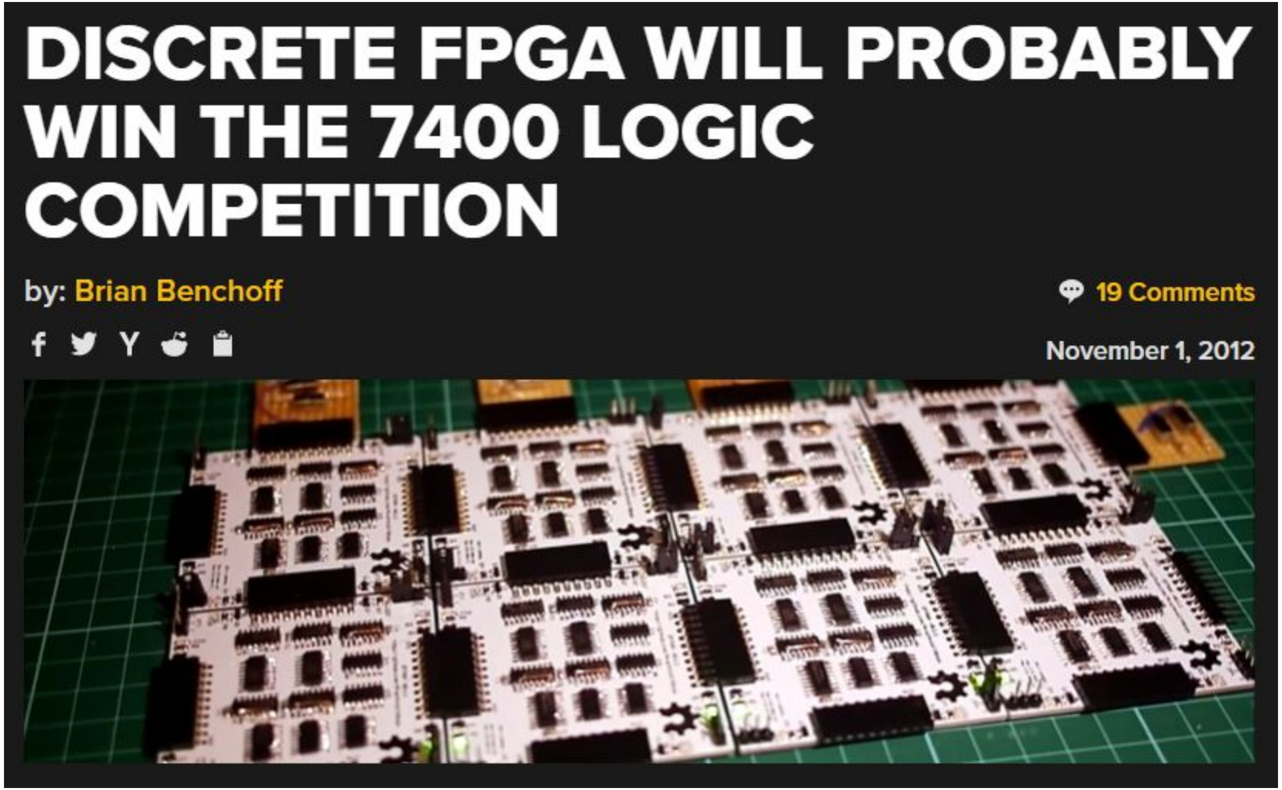
16 Comments

https://hackaday.com/tag/redstone/

8-bit CPU
by Ben Eater



https://www.youtube.com/watch?v=HyznrdDSSGM



DISCRETE FPGA WILL PROBABLY WIN THE 7400 LOGIC COMPETITION

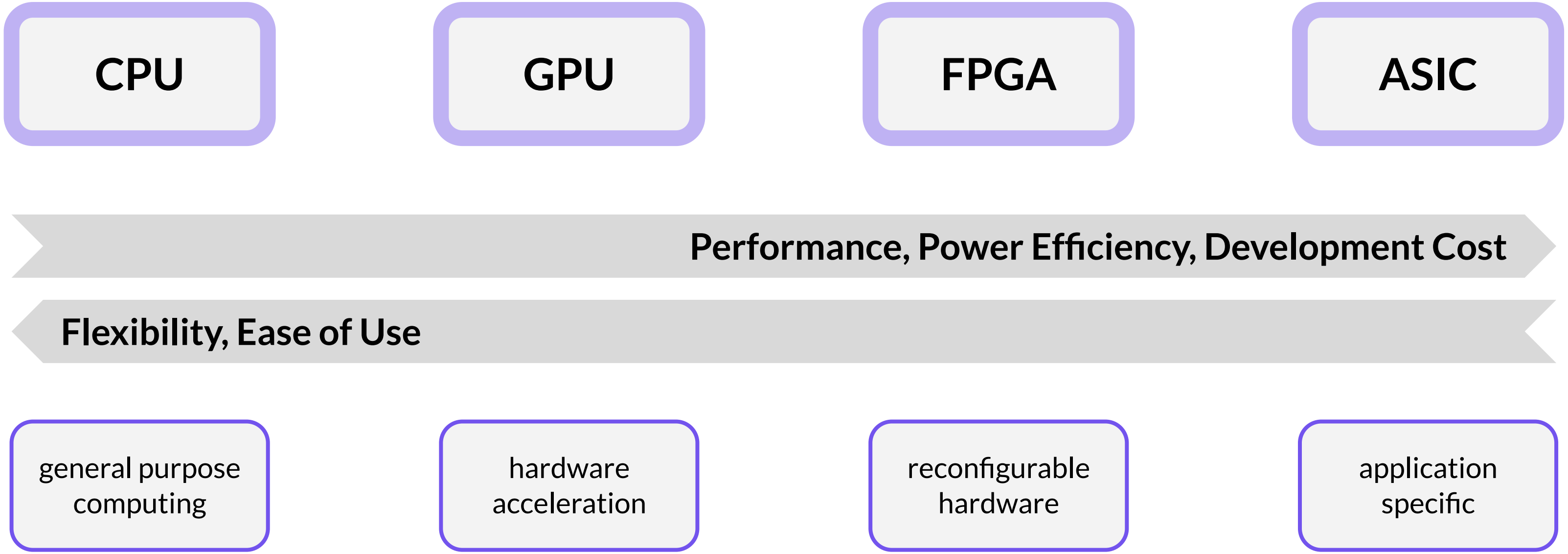
by: Brian Benchoff

19 Comments

November 1, 2012

https://hackaday.com/2012/11/01/discrete-fpga-will-probably-win-the-7400-logic-competition

Why do you need an FPGA?



CPU

GPU

FPGA

ASIC

Performance, Power Efficiency, Development Cost

Flexibility, Ease of Use

general purpose computing

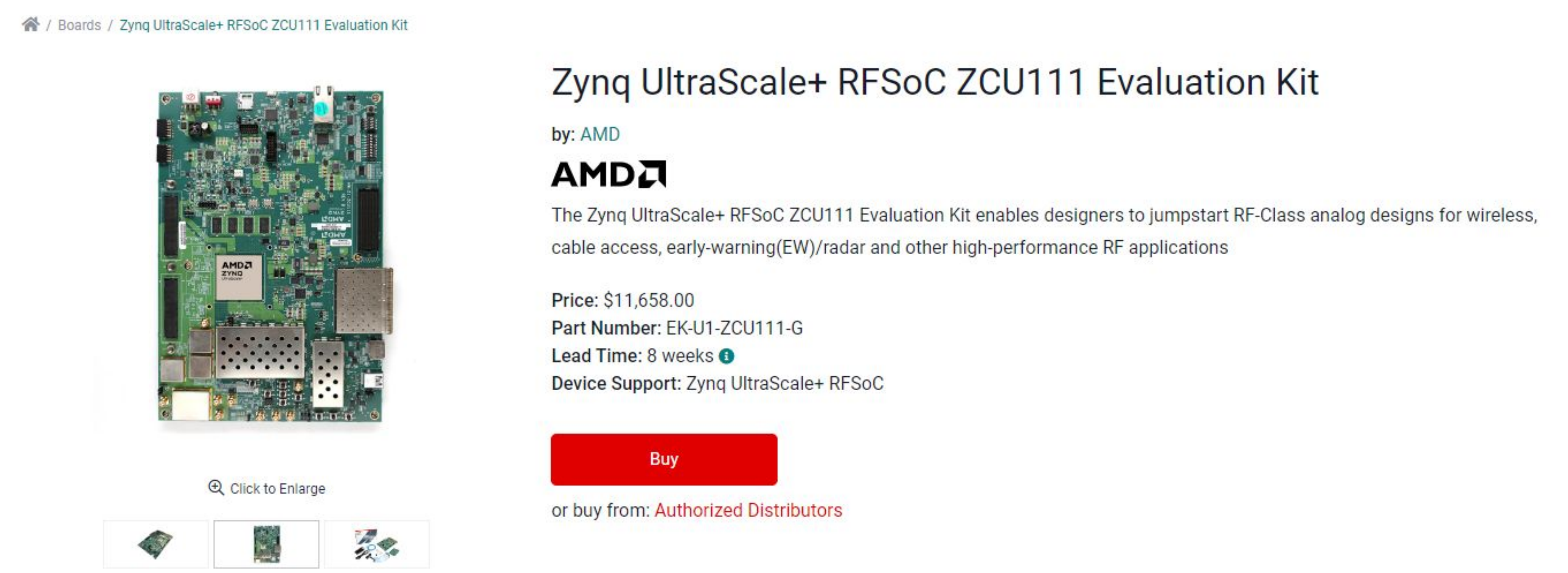
hardware acceleration

reconfigurable hardware

application specific

Applications	• Sensor Processing & Fusion • Motor Control • Low-cost Ultrasound • Traffic Engineering	• Flight Navigation • Missile & Munitions • Military Construction • Secure Solutions • Networking • Cloud Computing Security • Data Center • Machine Vision • Medical Endoscopy	• Situational Awareness • Surveillance/Reconnaissance • Smart Vision • Image Manipulation • Graphic Overlay • Human Machine Interface • Automotive ADAS • Video Processing • Interactive Display

https://docs.xilinx.com/v/u/en-US/ultrascale-plus-fpga-product-selection-guide
https://docs.xilinx.com/v/u/en-US/zynq-ultrascale-plus-product-selection-guide



🏠 / Boards / Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit

Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit

by: AMD

The Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit enables designers to jumpstart RF-Class analog designs for wireless, cable access, early-warning(EW)/radar and other high-performance RF applications

Price: $11,658.00
Part Number: EK-U1-ZCU111-G
Lead Time: 8 weeks ℹ
Device Support: Zynq UltraScale+ RFSoC

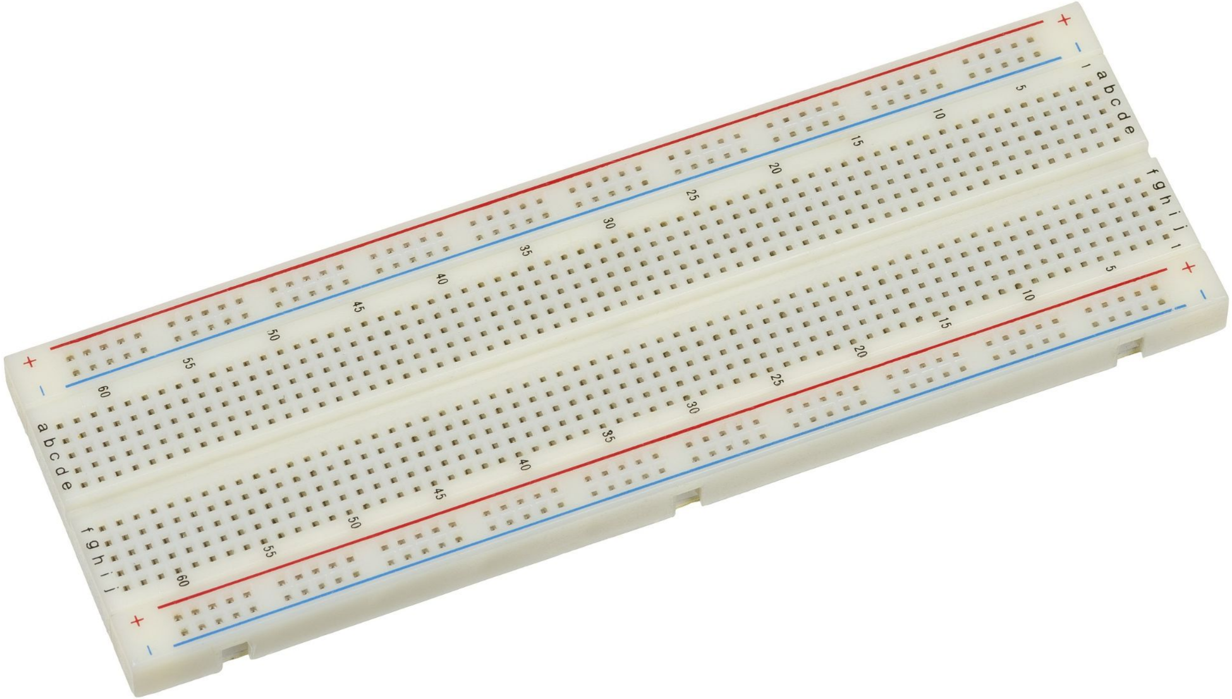
or buy from: Authorized Distributors

https://www.xilinx.com/products/boards-and-kits/zcu111.html

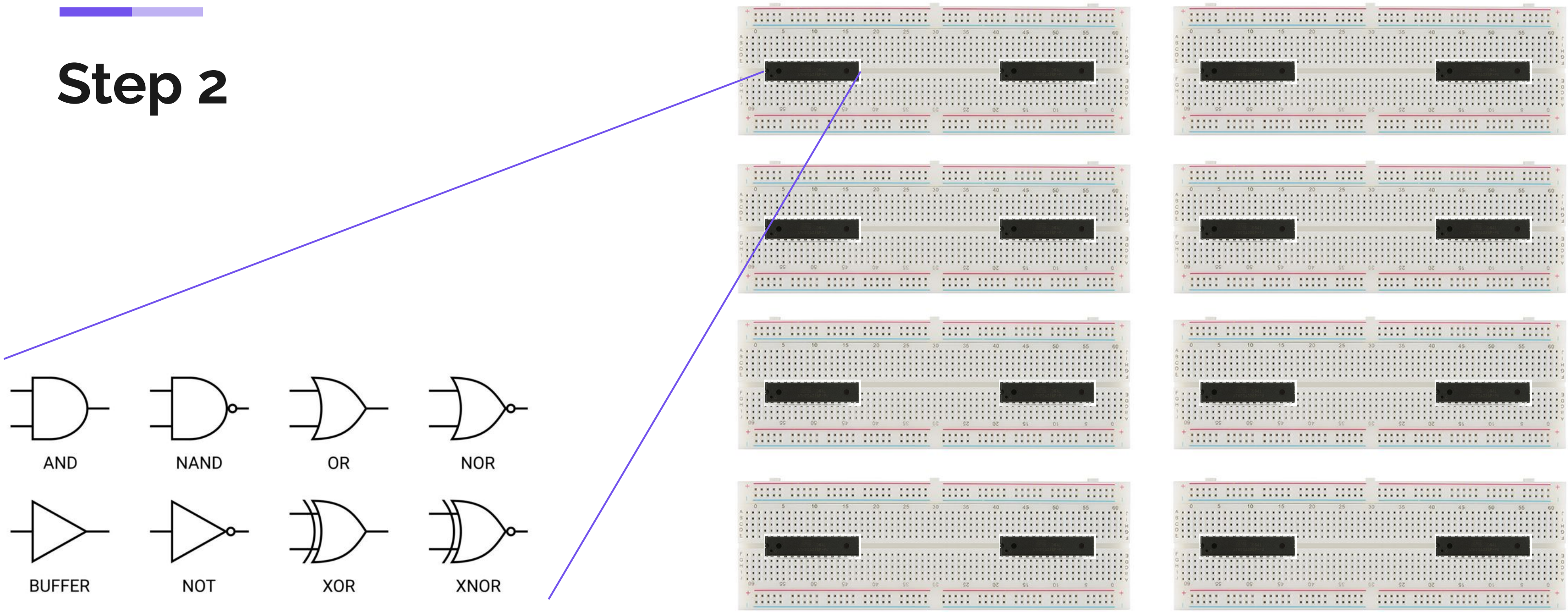
What is an FPGA?

How can we build any digital circuit?

Step 1



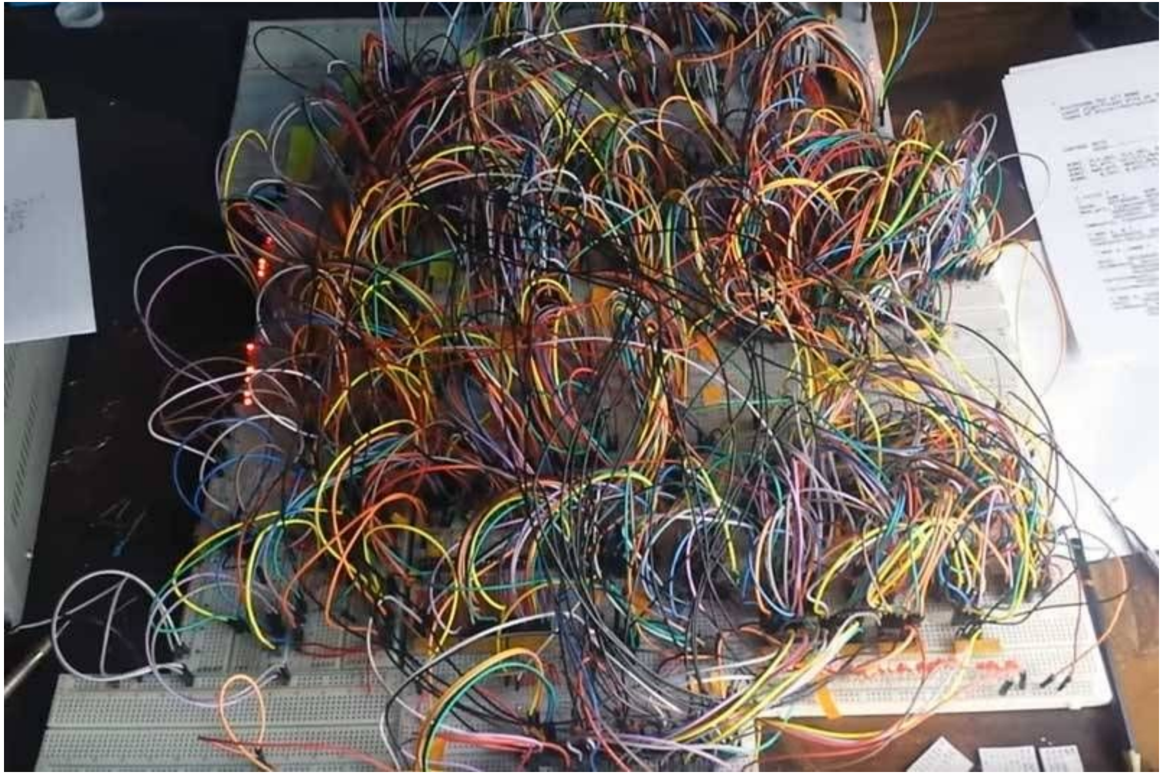
Step 2



...

Step 4

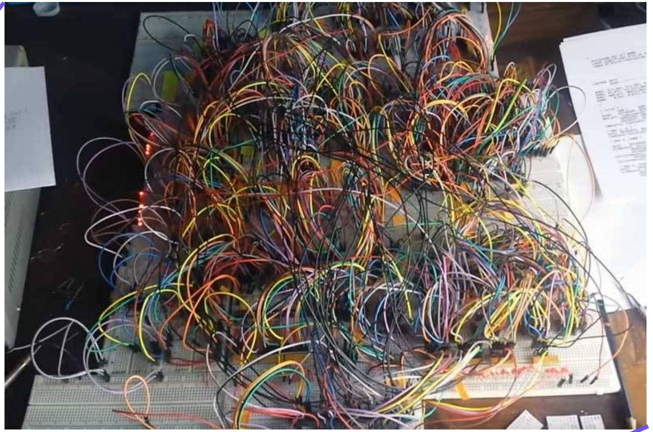
profit?

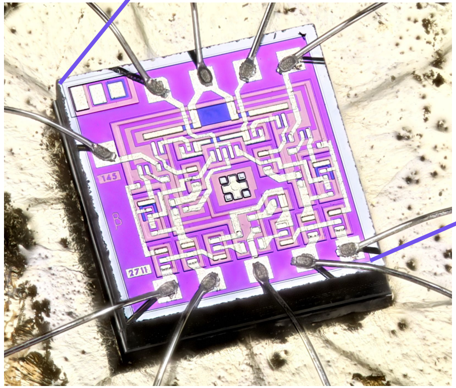


Problems of a breadboard?

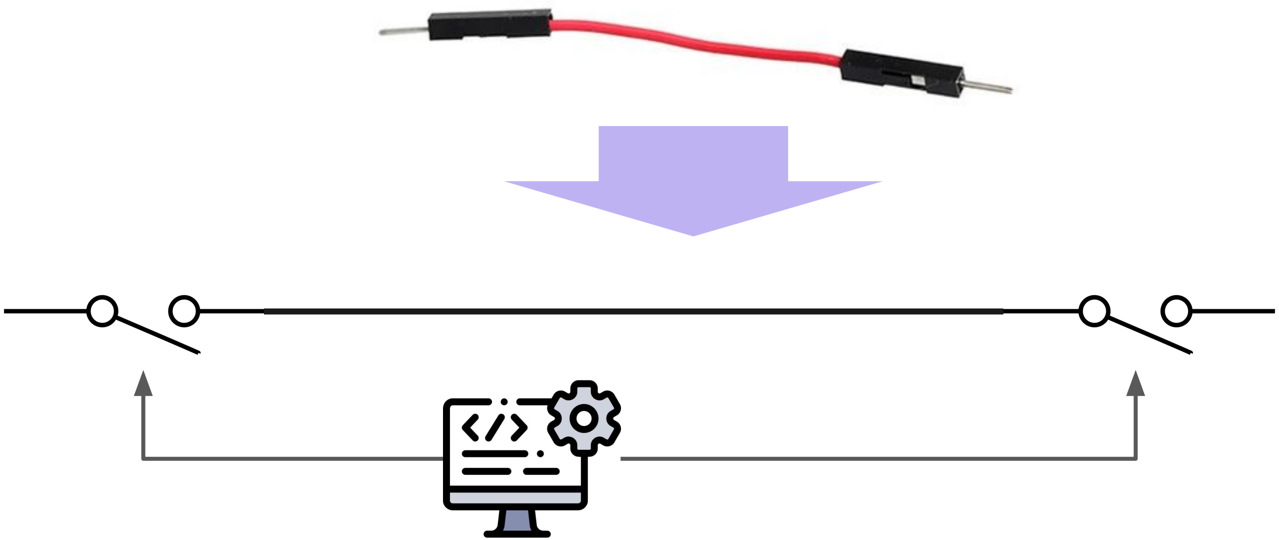
- BIG!
- you need wires (and hands)
- chaotic

too big?
→ shrink it



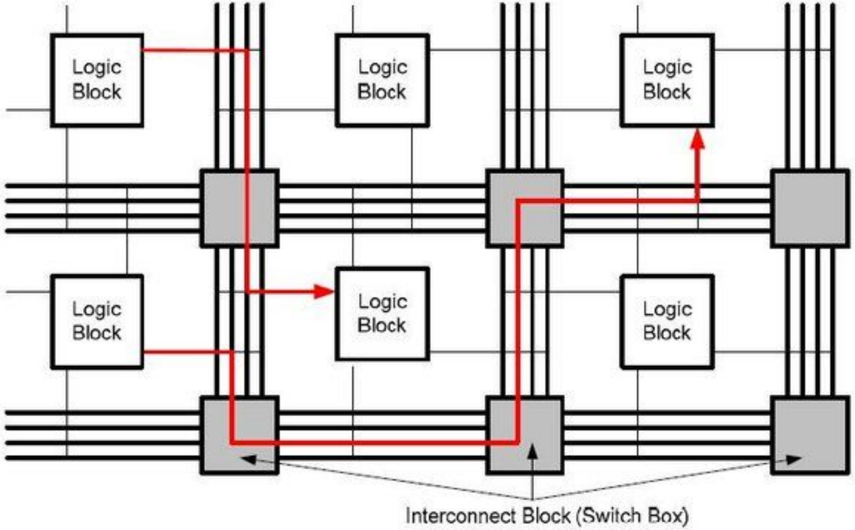


Wires?
→ make them programmable



chaotic?
→ define an interconnect

architecture for wires



https://www.researchgate.net/publication/331800773_Reconfigurable_hardware_acceleration_of_CNNs_on_FPGA-based_smart_cameras/figures

this is an FPGA

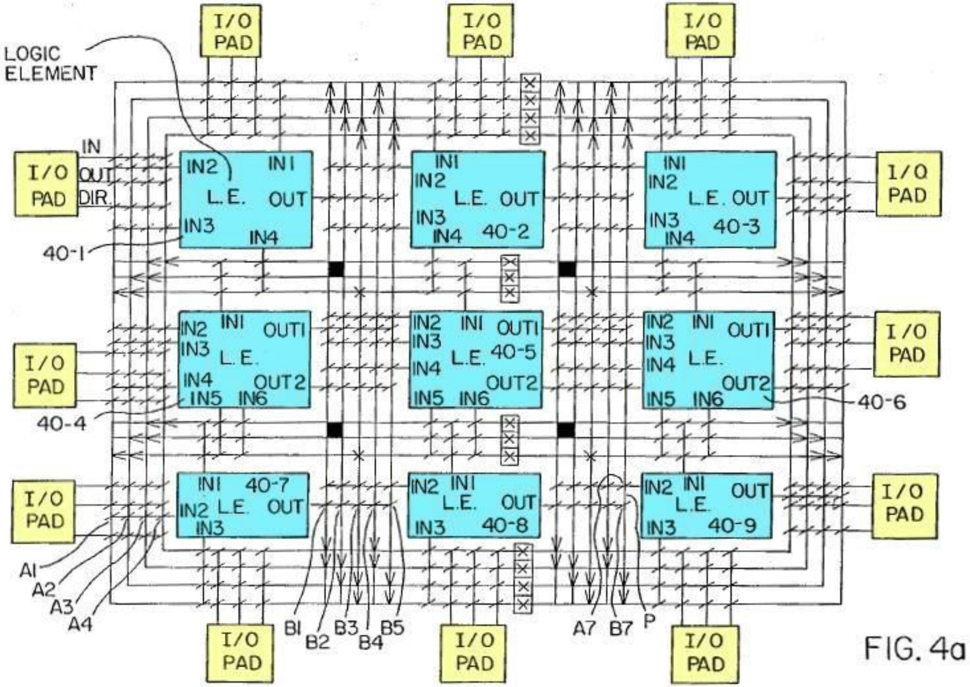
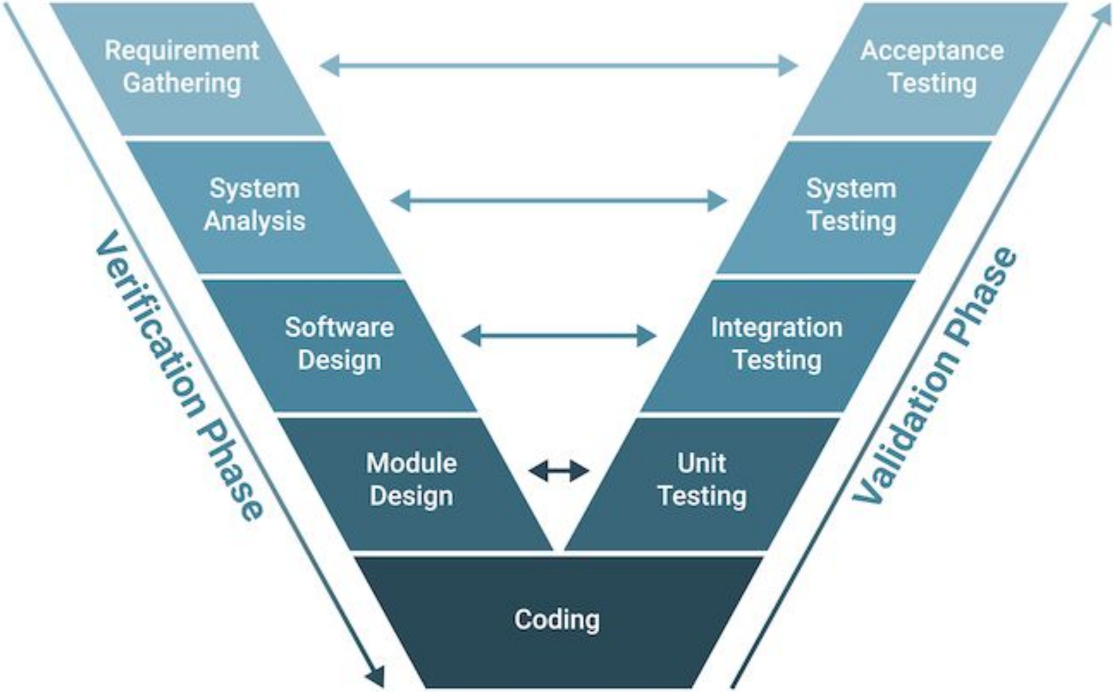


FIG. 4a

https://semiwiki.com/fpga/290990-reverse-engineering-the-first-fpga-chip-xilinx-xc2064/

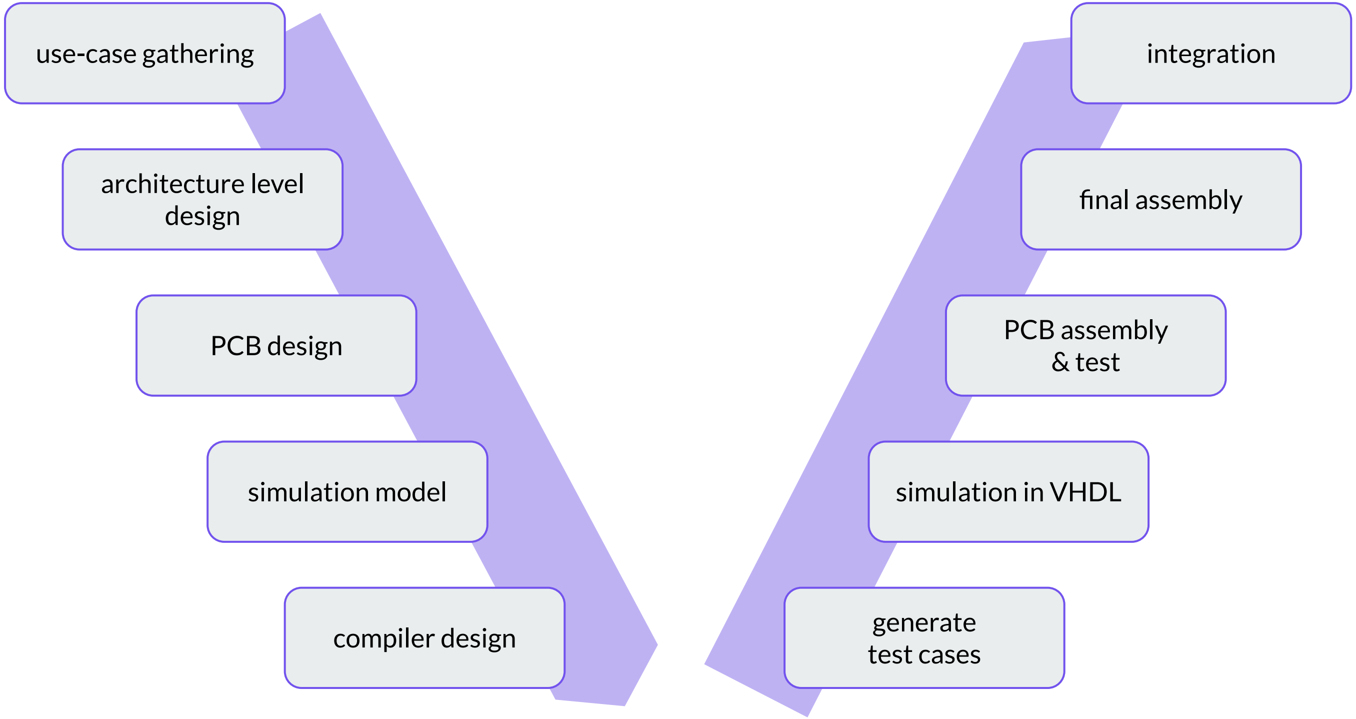
How do you build an FPGA?

V-Model



https://builtin.com/software-engineering-perspectives/v-model

V-Model



use-case gathering

architecture level design

PCB design

simulation model

compiler design

generate test cases

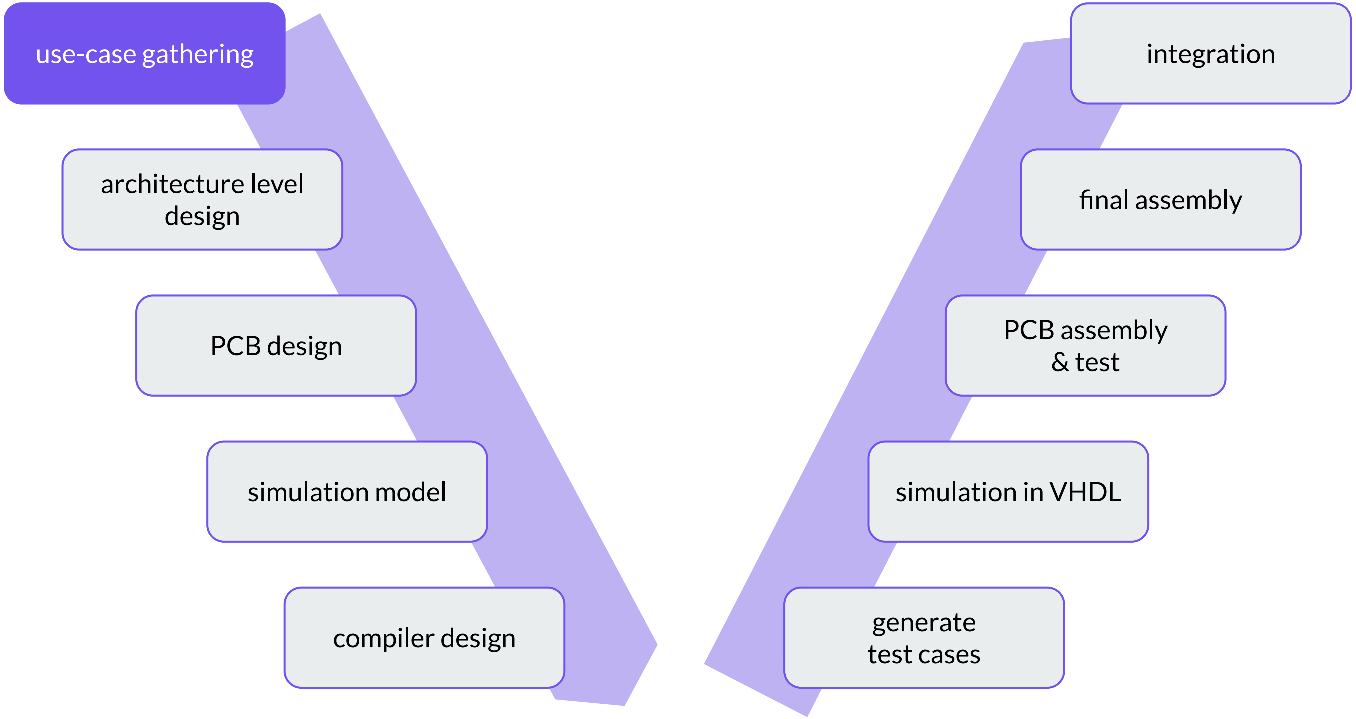
simulation in VHDL

PCB assembly & test

final assembly

integration

V-Model


use-case gathering

architecture level design

PCB design

simulation model

compiler design

generate test cases

simulation in VHDL

PCB assembly & test

final assembly

integration

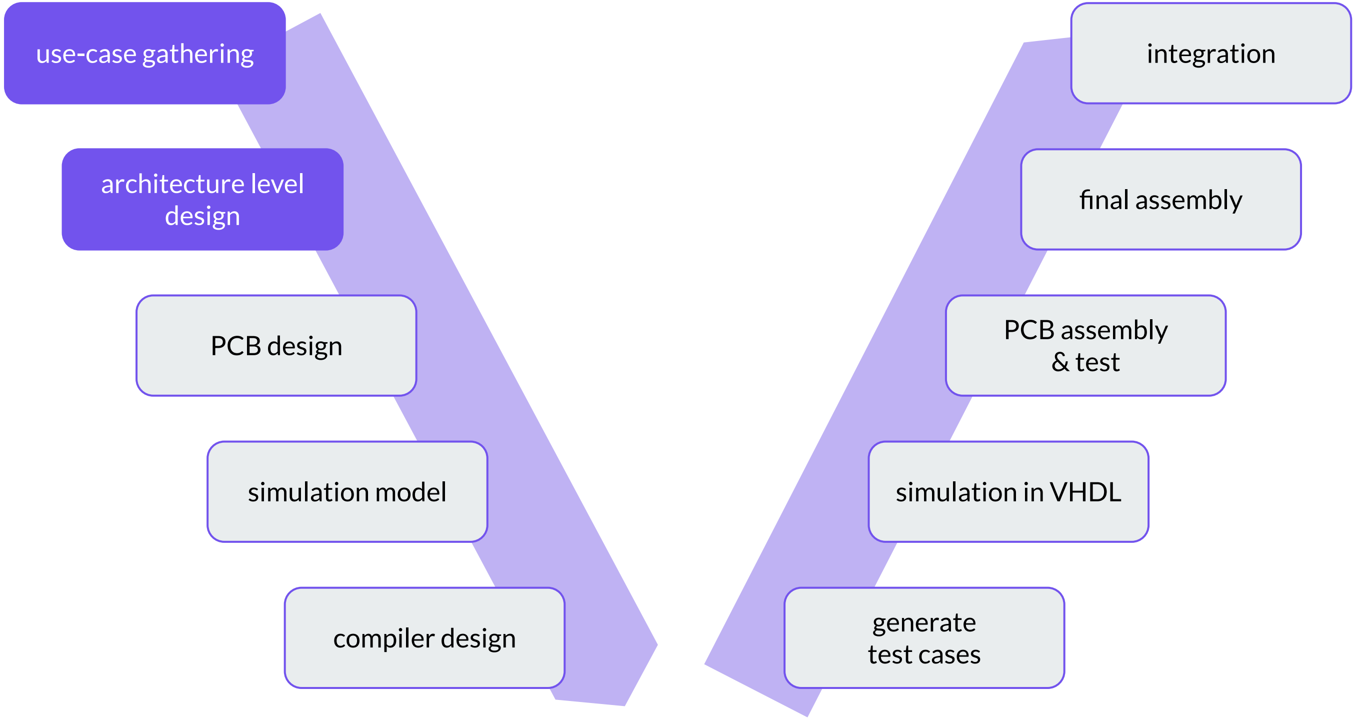
Which applications can I run on my FPGA?

- 4-bit counter → Yes
- 4-bit adder → Yes
- BCD to 7-segment decoder → Yes

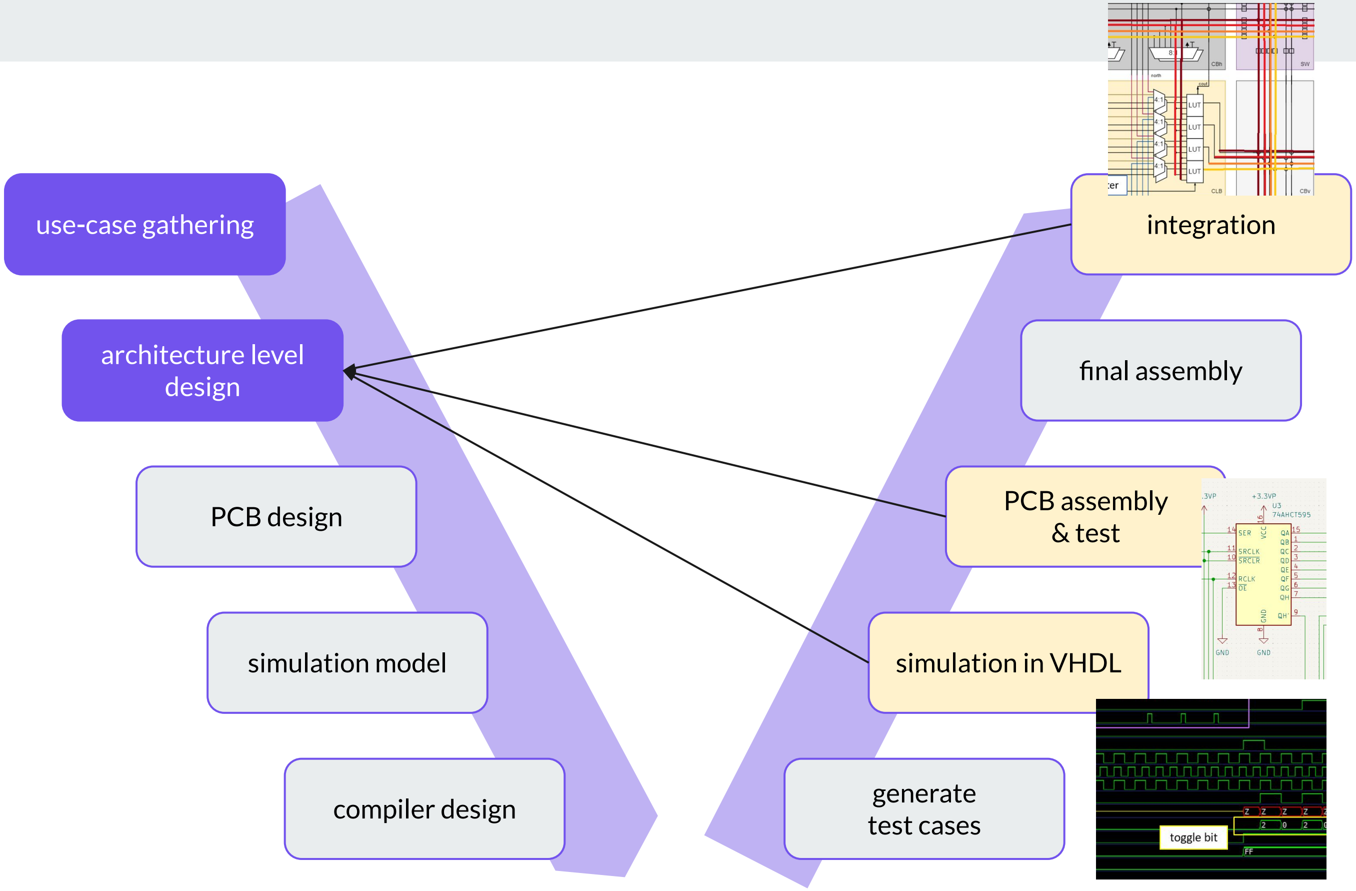
- clock-domain crossing → maybe
- digital “random” number generator (Game die) → maybe

- 8-bit CPU → **NO!**

V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
integration
final assembly
PCB assembly & test
simulation in VHDL
generate test cases

V-Model



use-case gathering

architecture level design

PCB design

simulation model

compiler design

generate test cases

simulation in VHDL

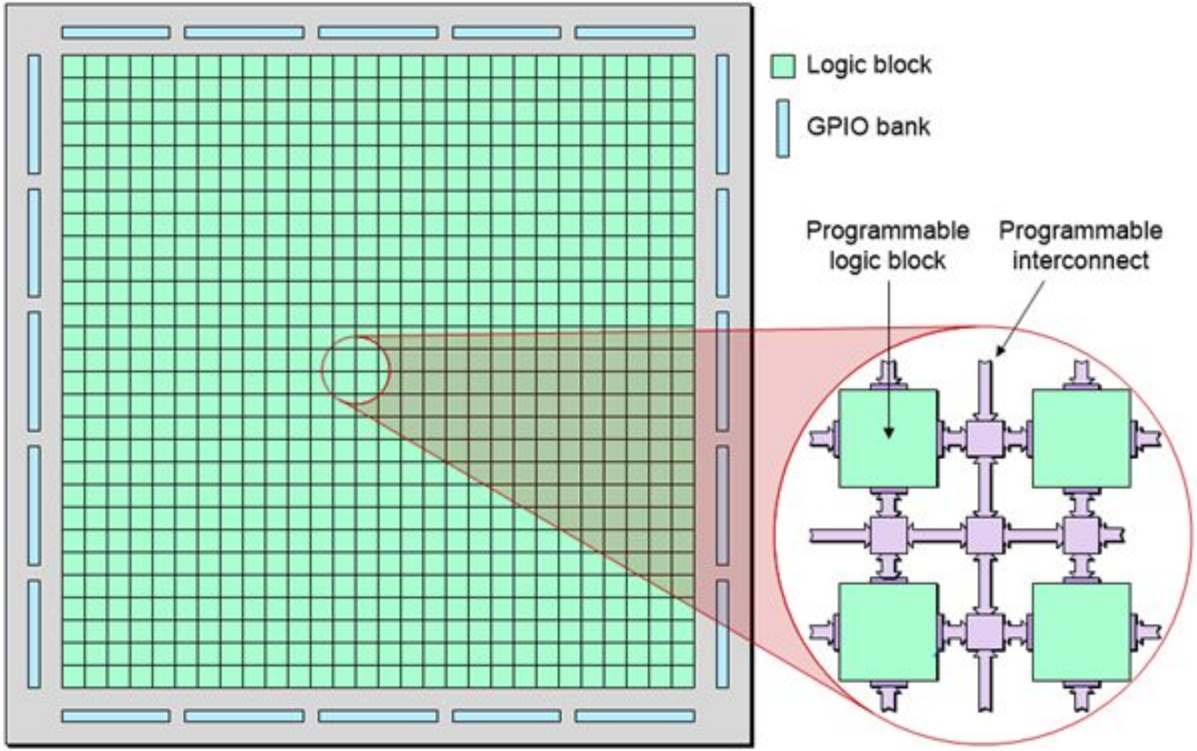
PCB assembly & test

final assembly

integration

Research Architectures

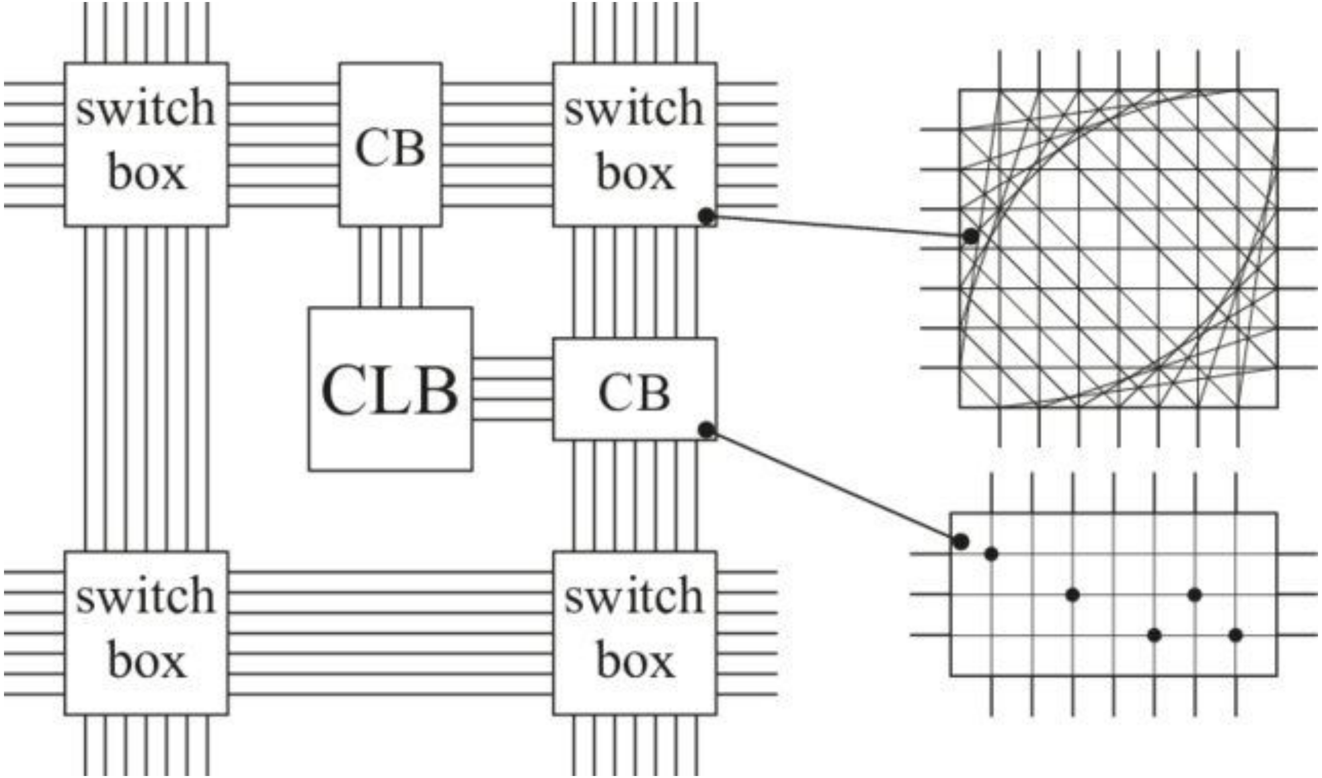
architecture level design



Bird's-eye view of FPGA

https://www.latticesemi.com/en/What-is-an-FPGA

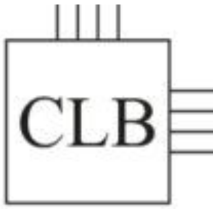
architecture level design



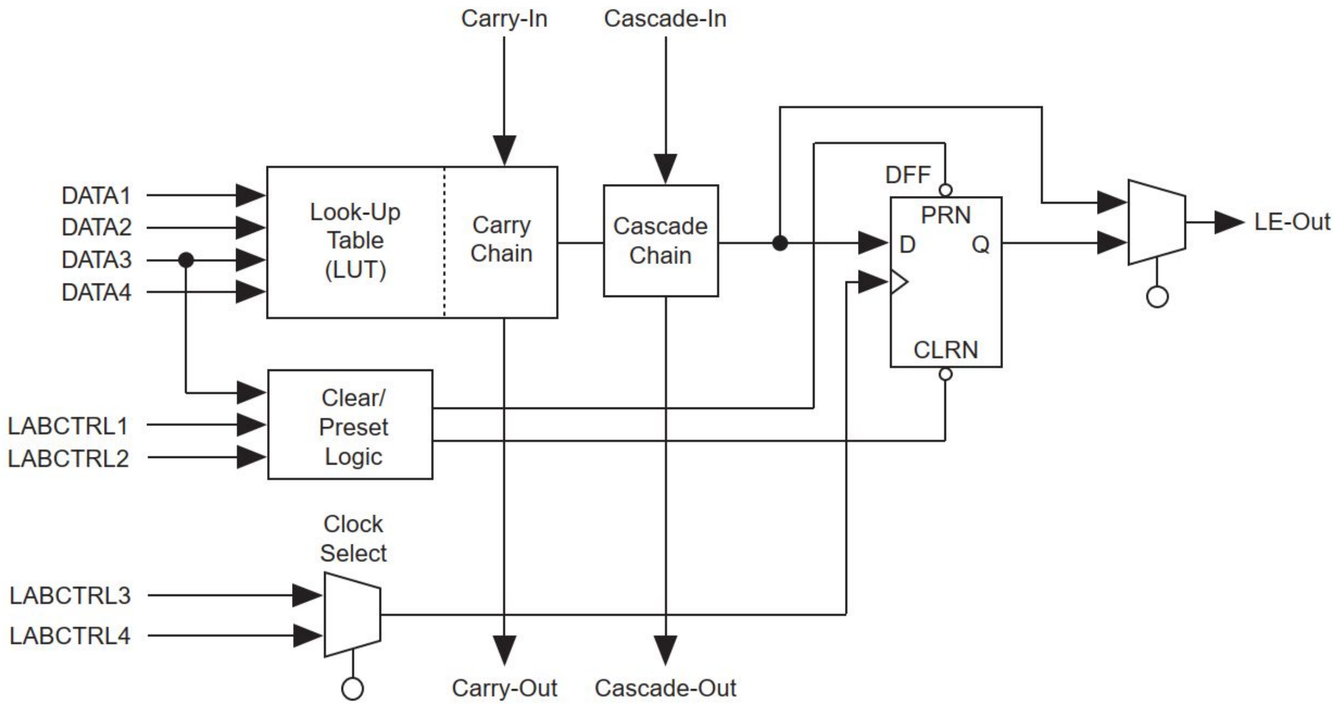
https://www.researchgate.net/publication/282769665_Adapting_Hardware_Systems_by_Means_of_Multi-Objective_Evolution/figures

architecture level design

The Configurable Logic Block (CLB)

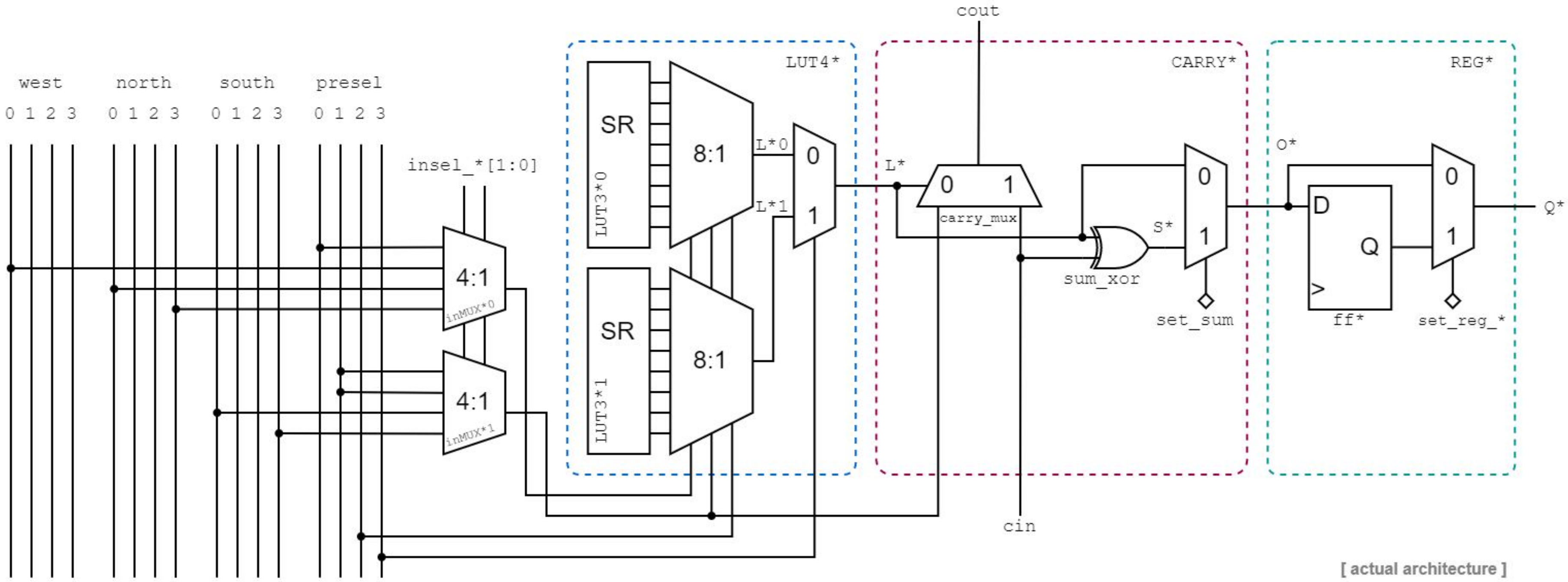


architecture level design



Altera Flex 8000: https://flex.phys.tohoku.ac.jp/riron/vhdl/up1/altera/ds/dsf8k.pdf

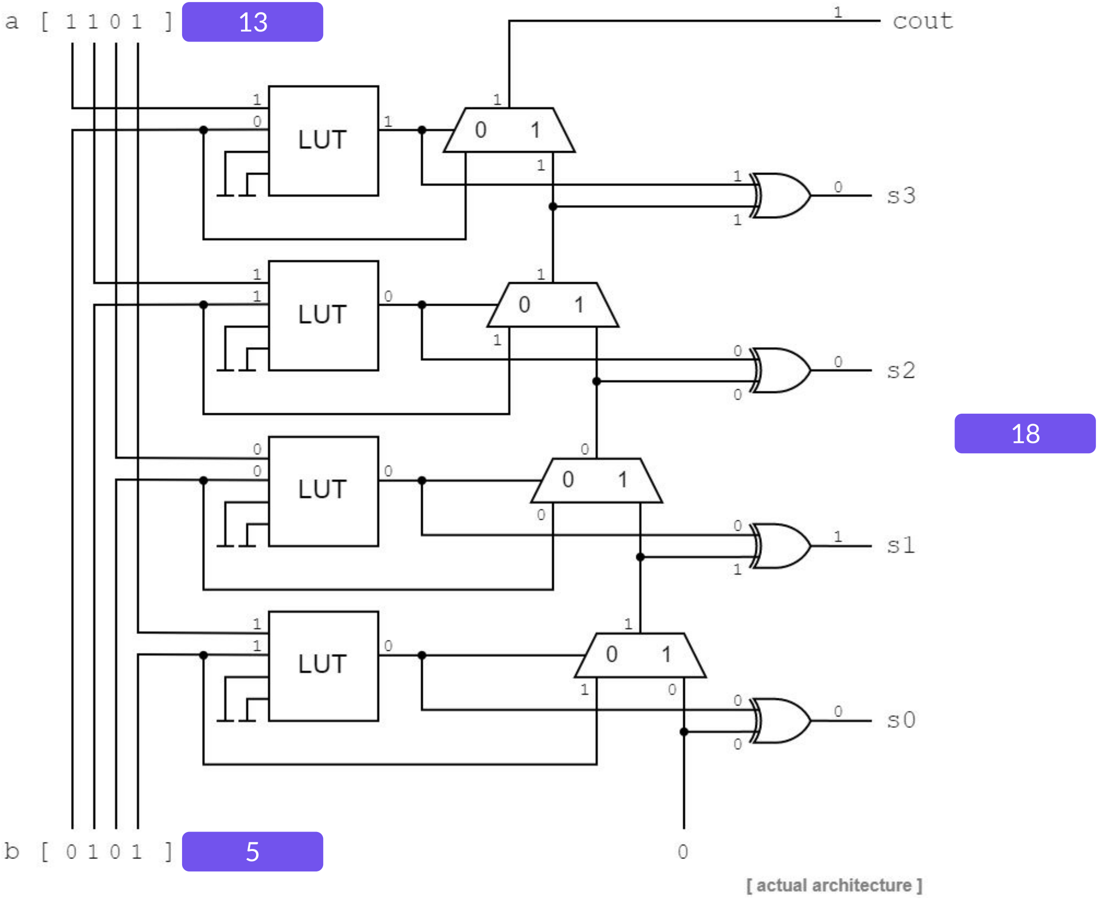
architecture level design



[actual architecture]

architecture level design

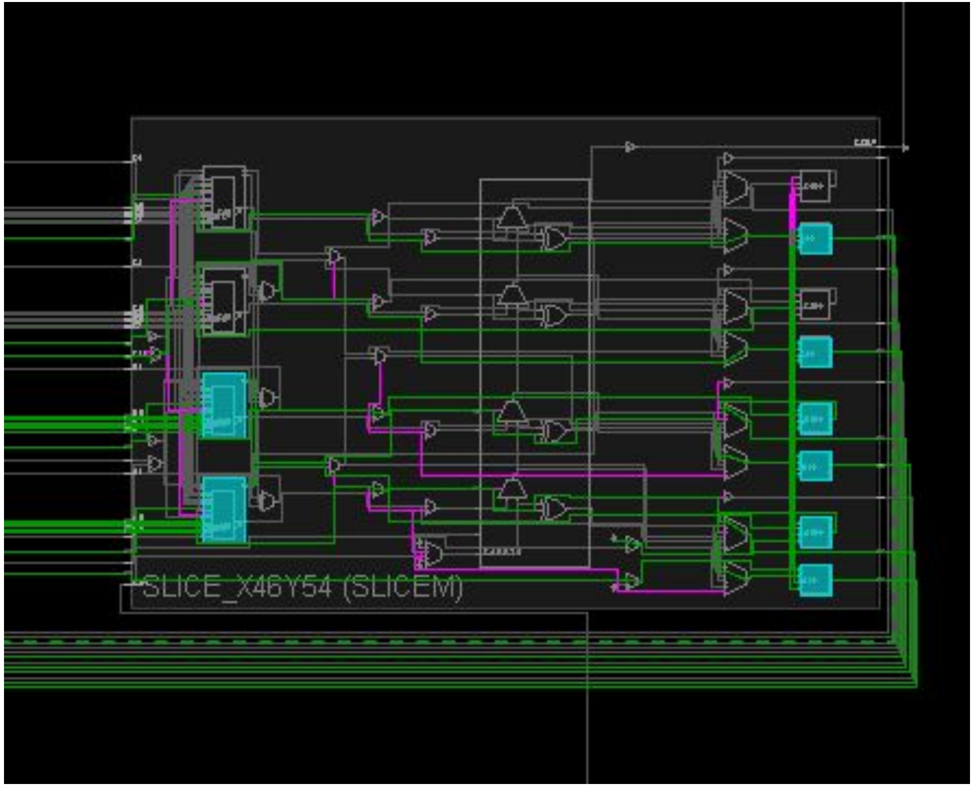
Carry chain



[actual architecture]

architecture level design

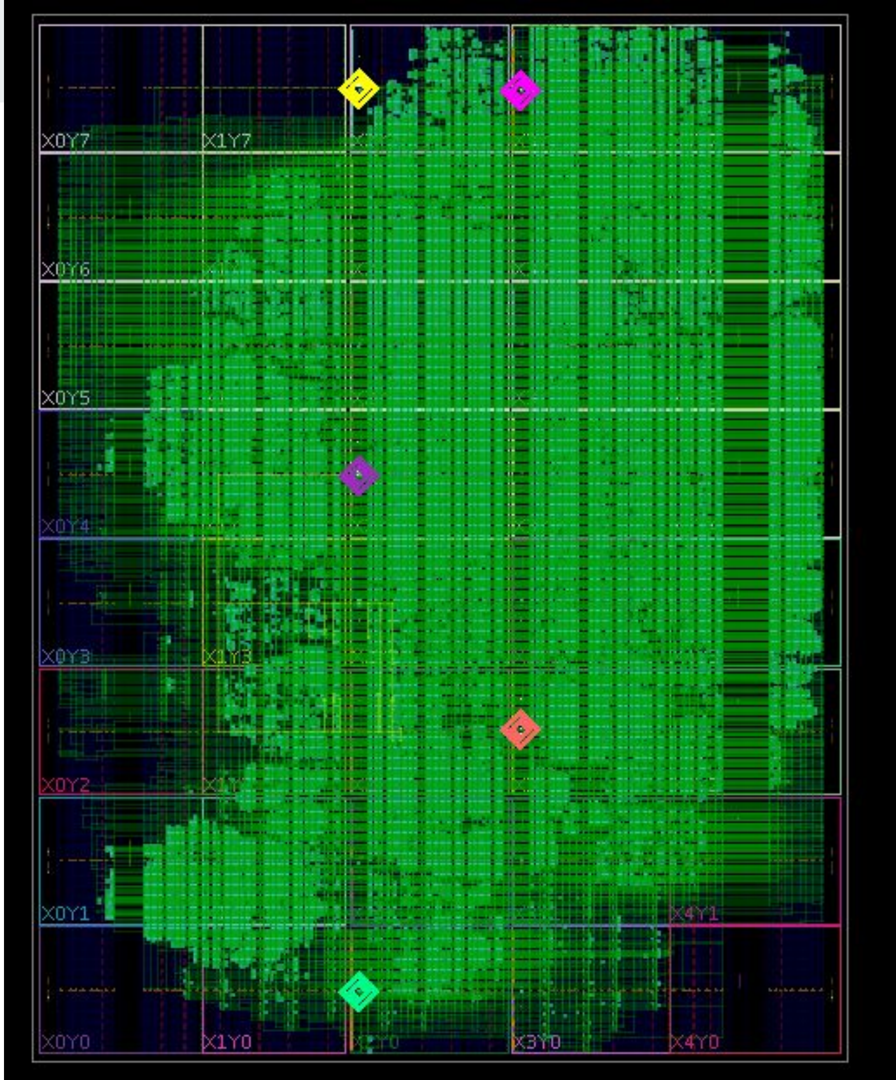
Vivado



https://support.xilinx.com/s/article/54683

architecture level design

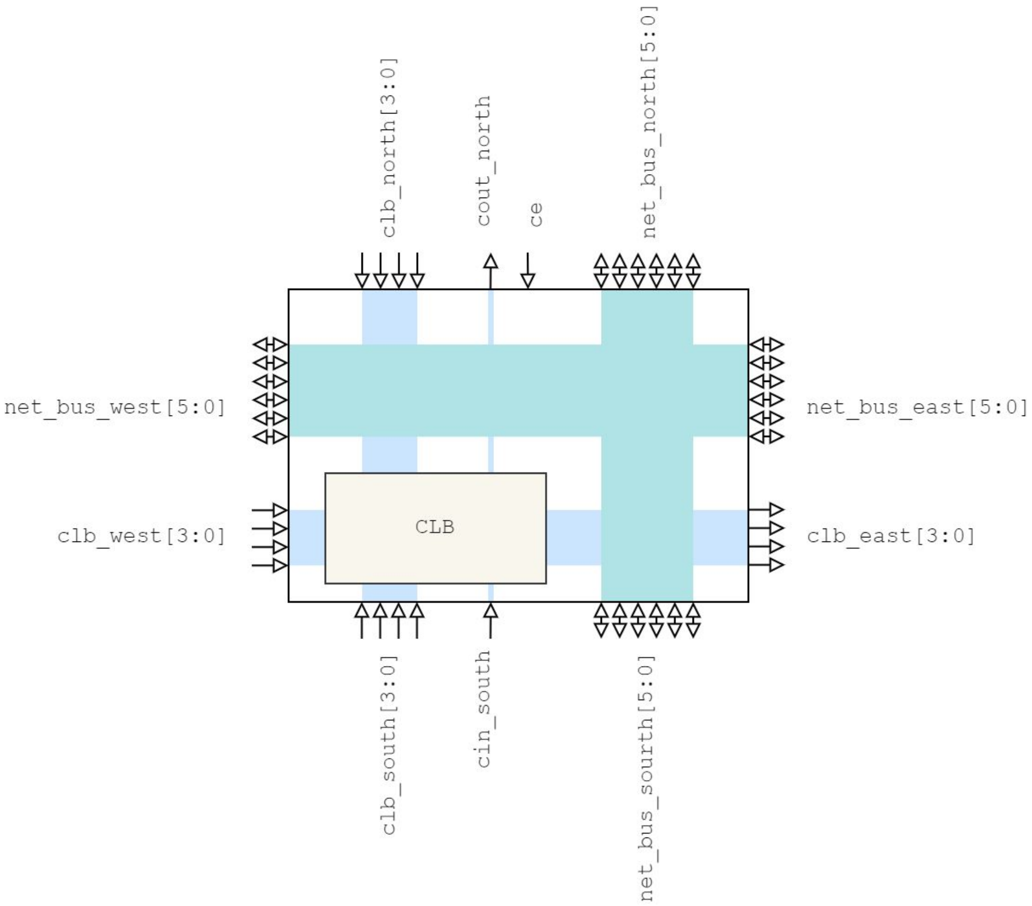
Vivado



https://support.xilinx.com/s/article/67384

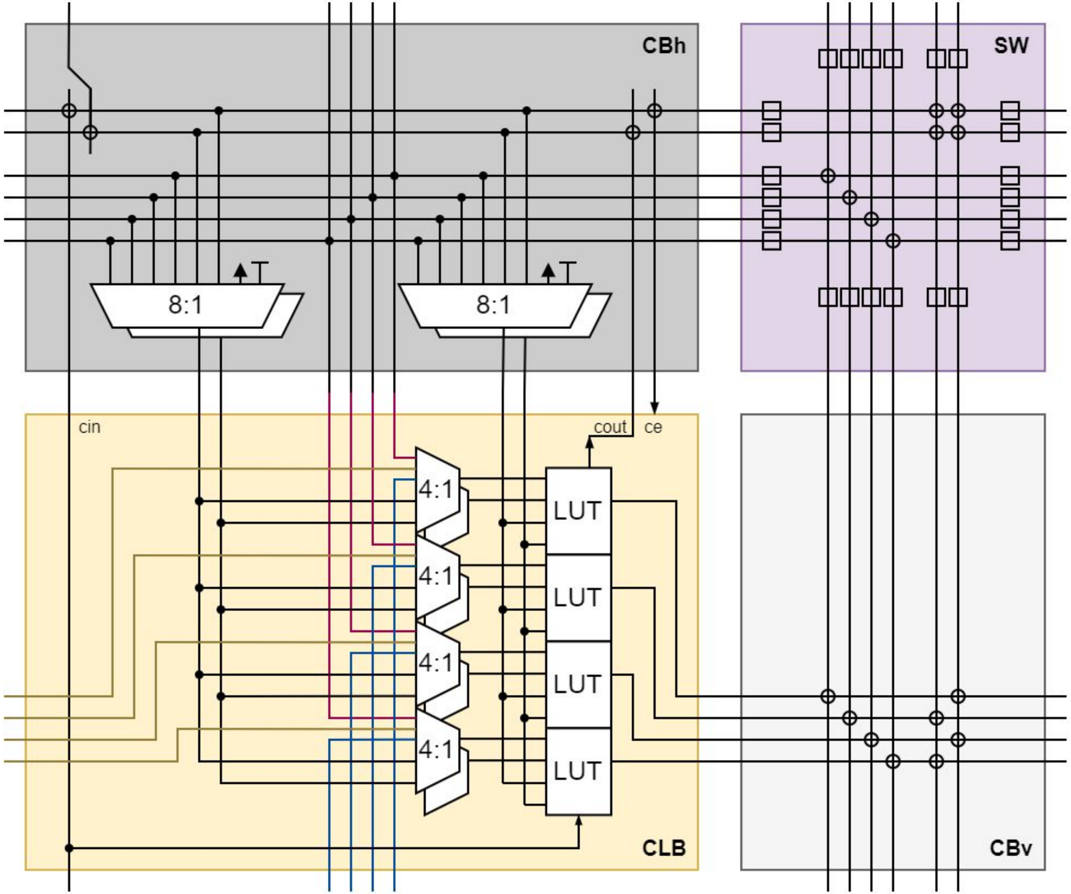
architecture level design

The Interconnect



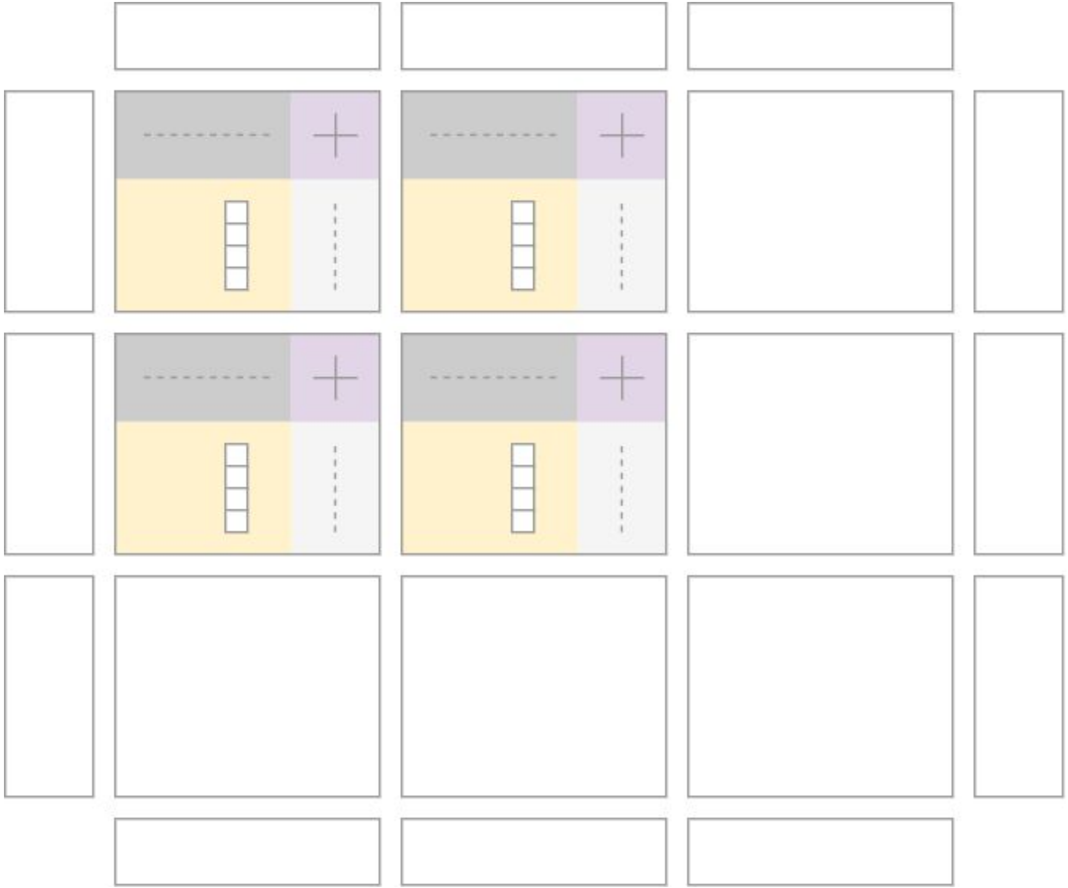
architecture level design

The Interconnect



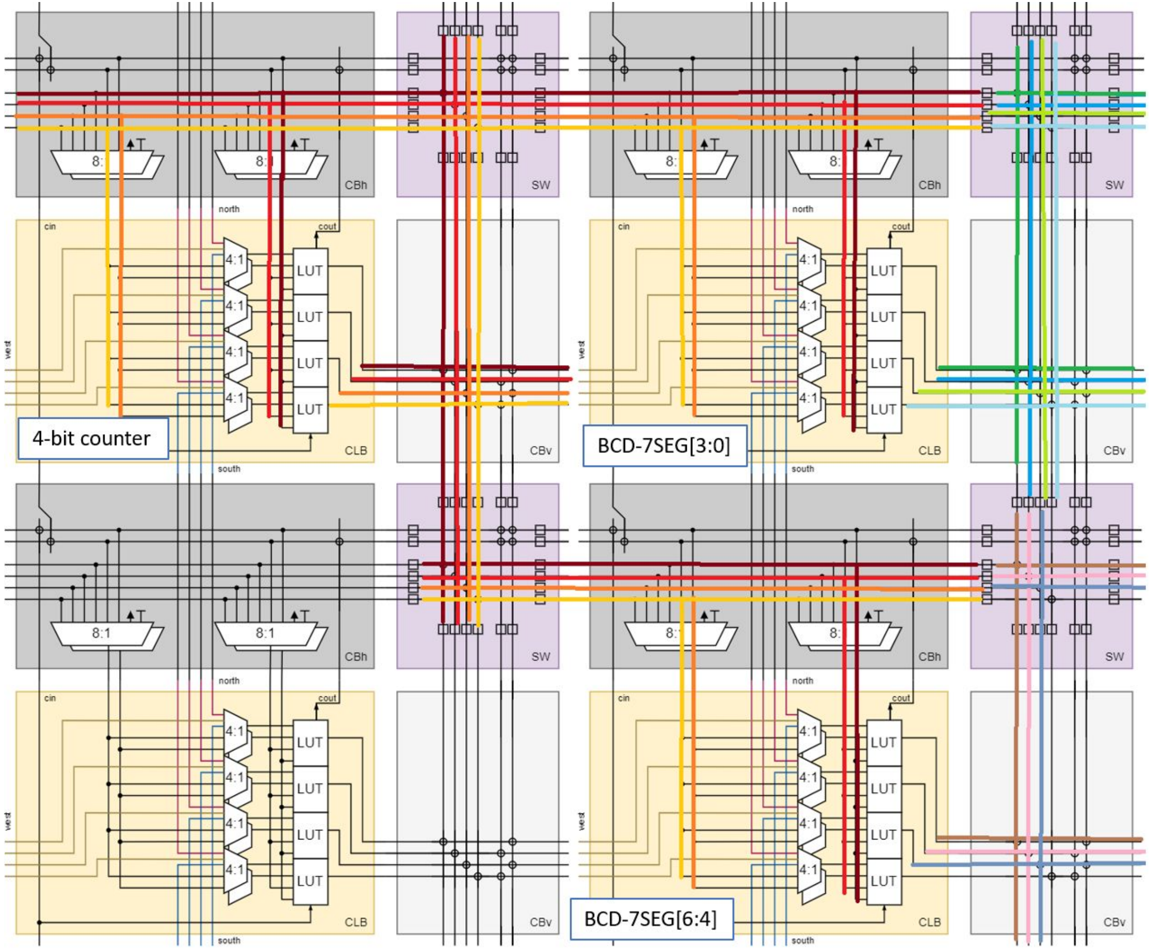
architecture level design

modular FPGA



architecture level design

Can I *place* and *route* my desired applications?

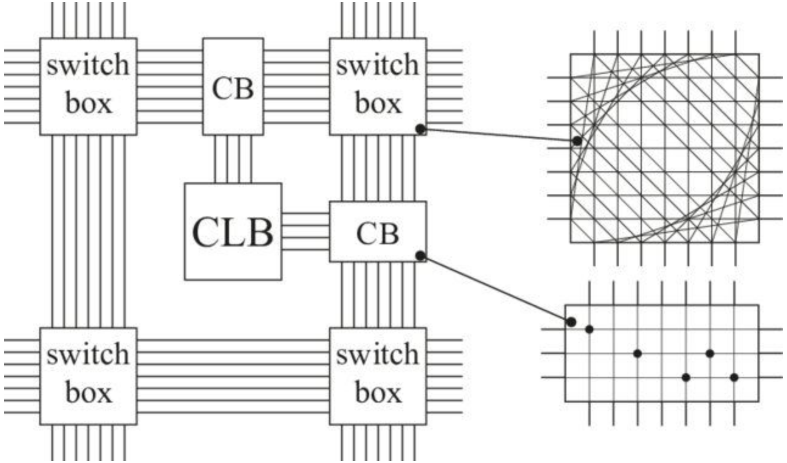


Flexibility vs. Complexity

… the greatest challenge of this project

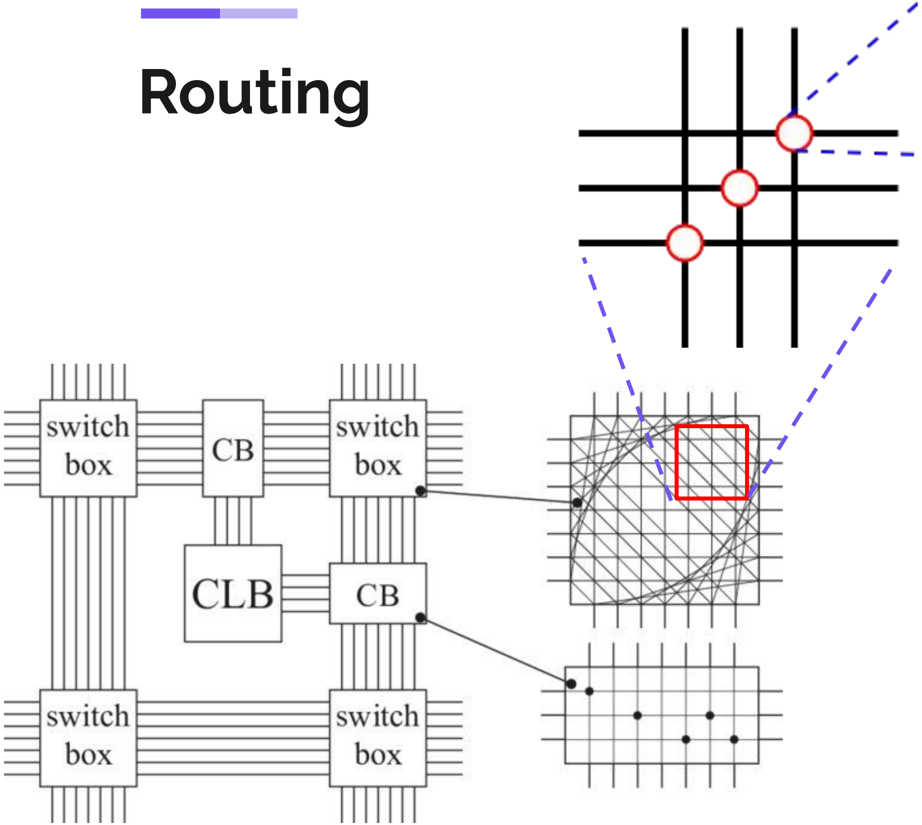
architecture level design

Routing



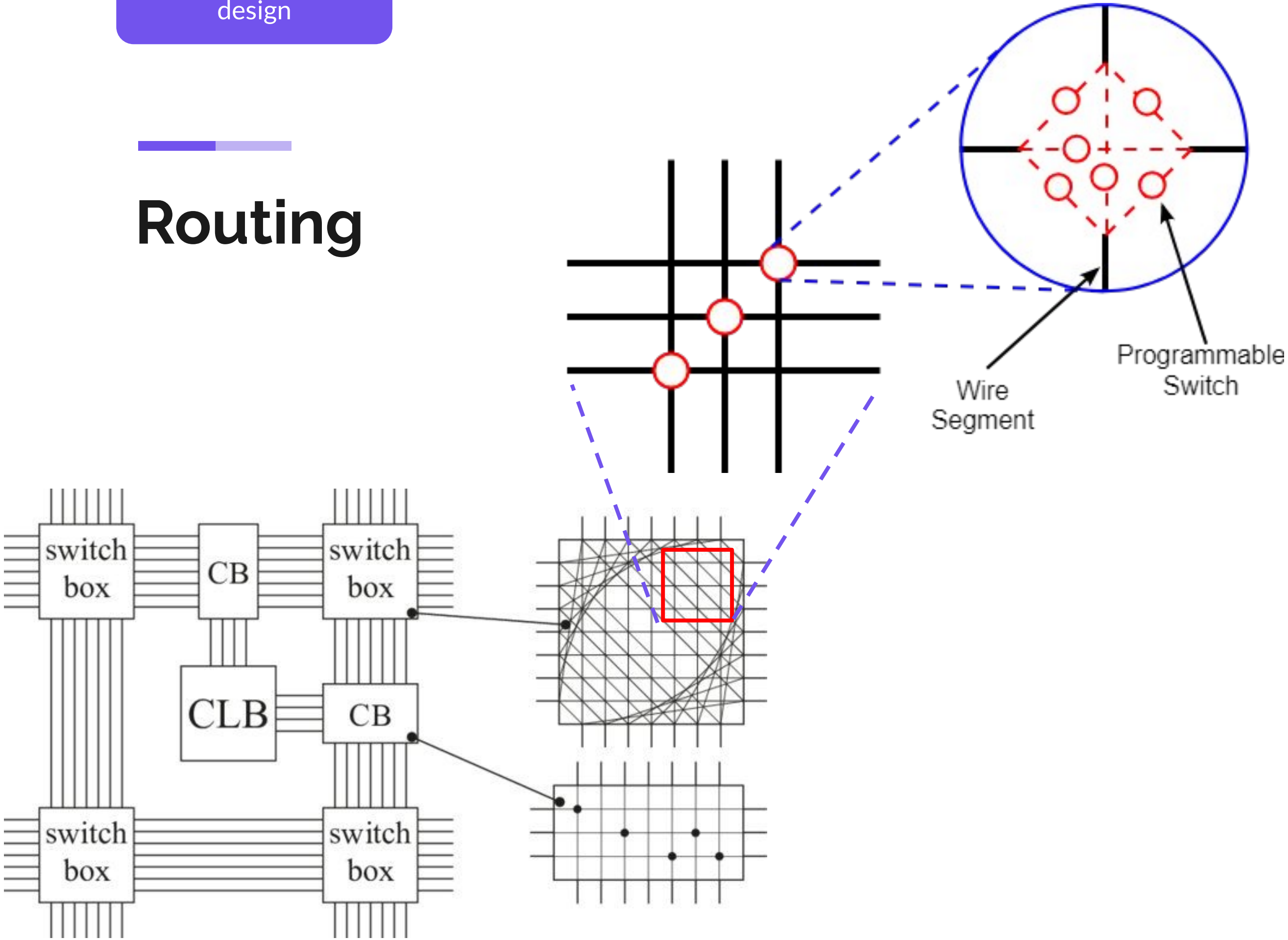
architecture level design

Routing



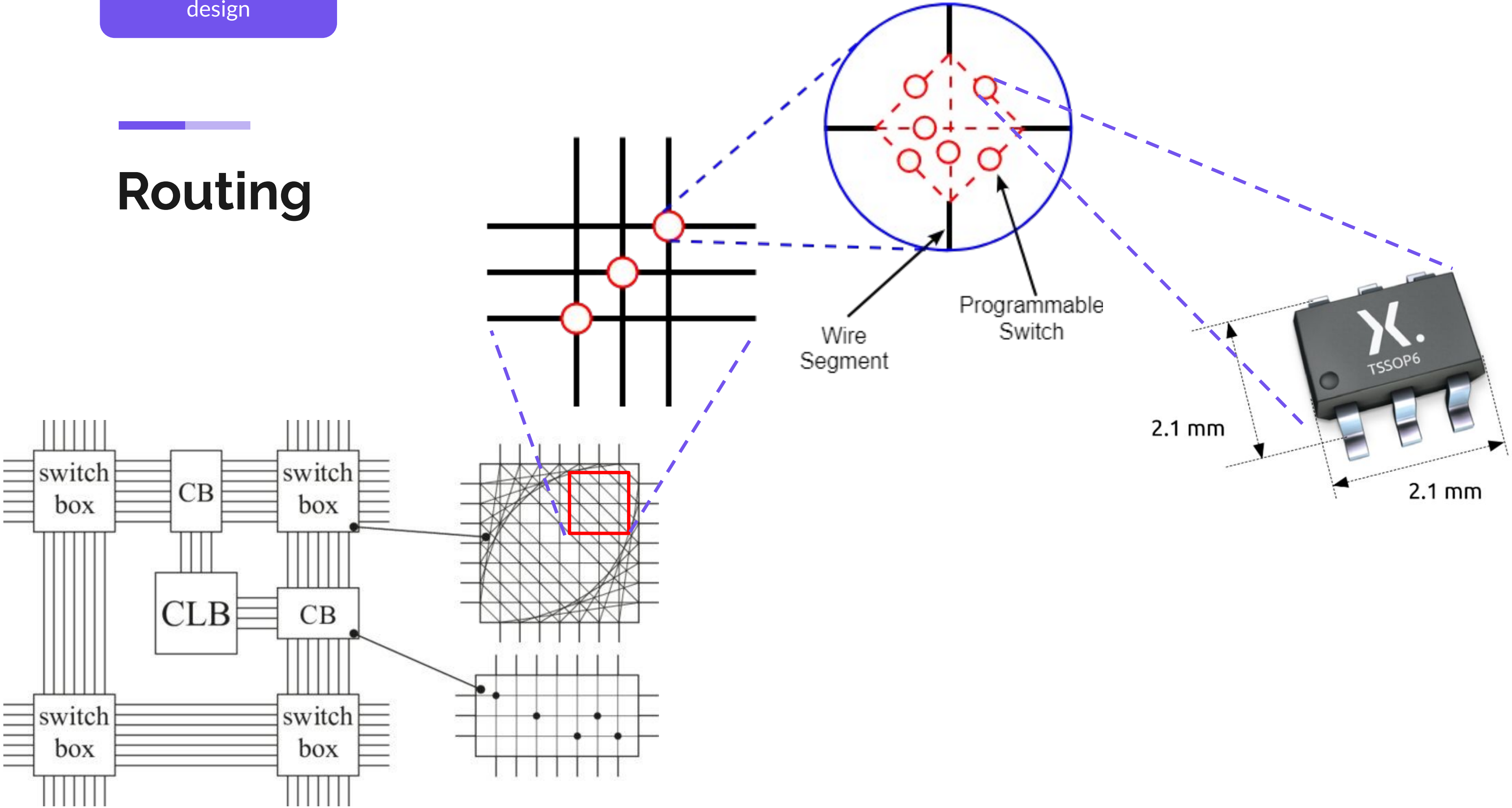
architecture level design

Routing



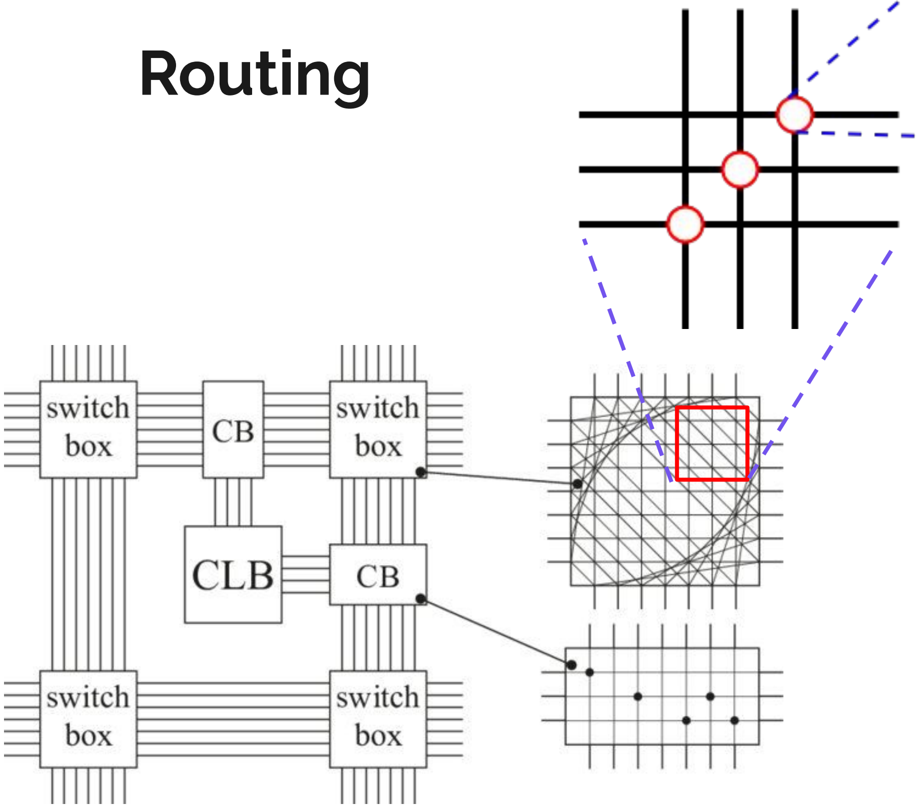
architecture level design

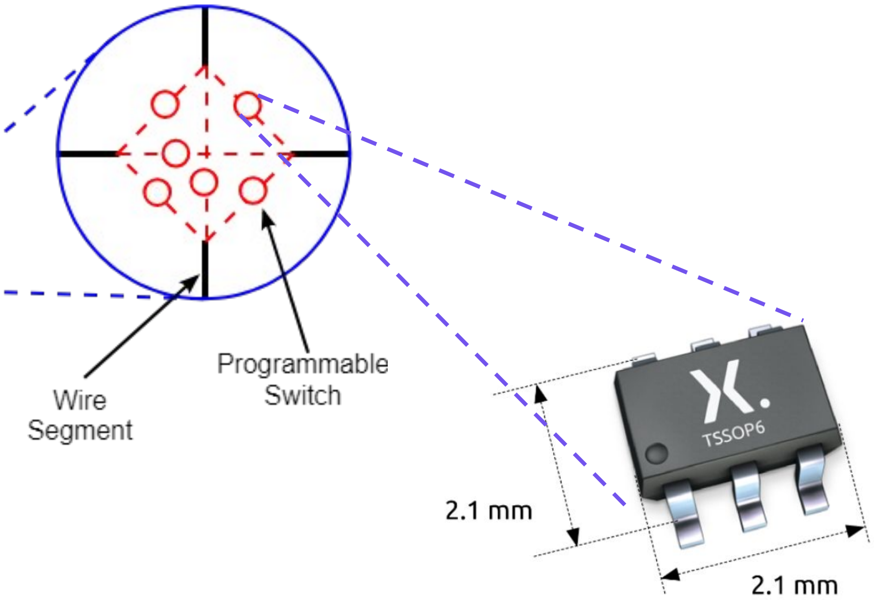
Routing



architecture level design

Routing

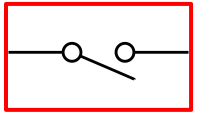


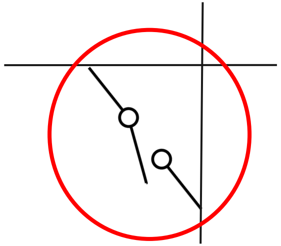


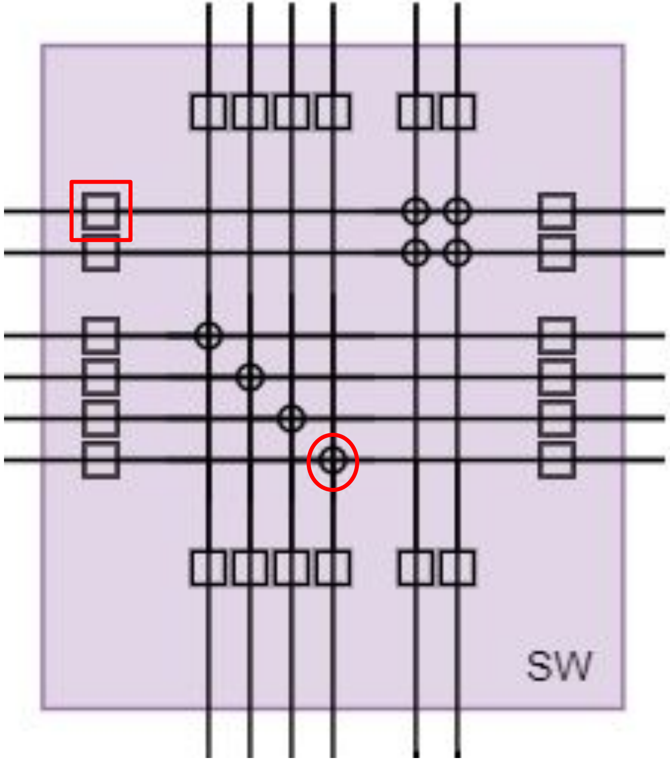
4 x 4 signals x 6 switches = 96
→ 96 ICs to layout on a PCB
→ 96 ICs to buy
→ 96 ICs to solder
→ 96 bits to manage in the bitstream

architecture level design

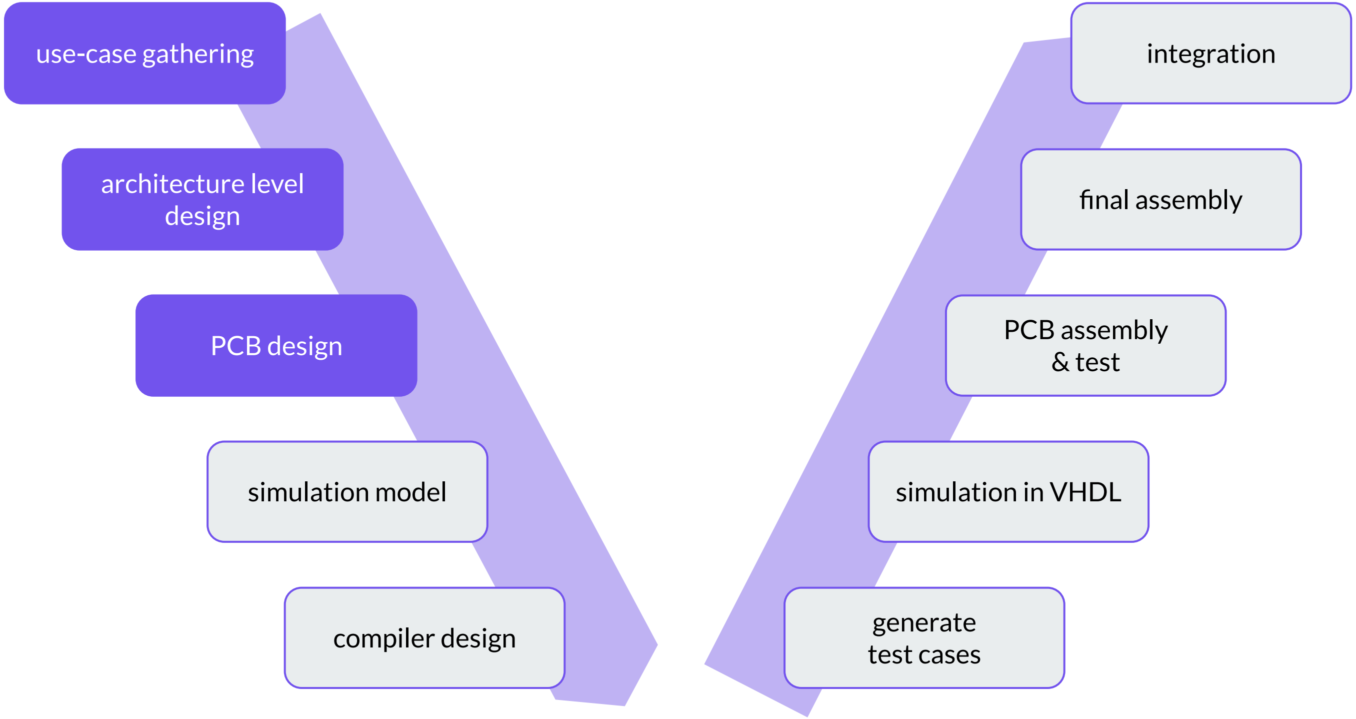
Switch Box







V-Model



use-case gathering

architecture level design

PCB design

simulation model

compiler design

integration

final assembly

PCB assembly & test

simulation in VHDL

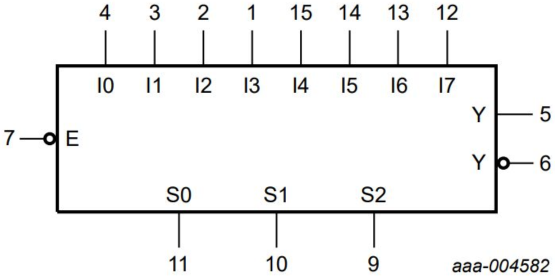
generate test cases

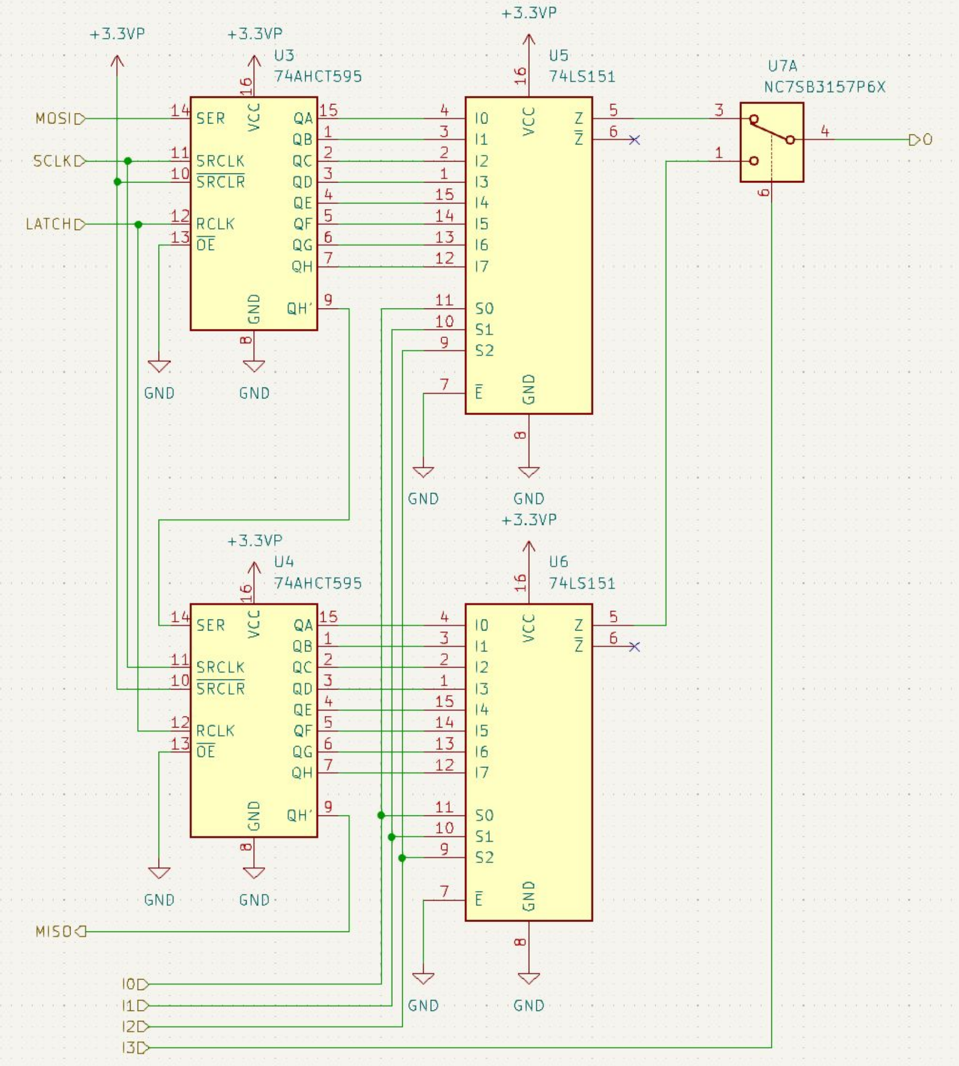
PCB design

LUT4

74HC151; 74HCT151

8-input multiplexer





PCB design

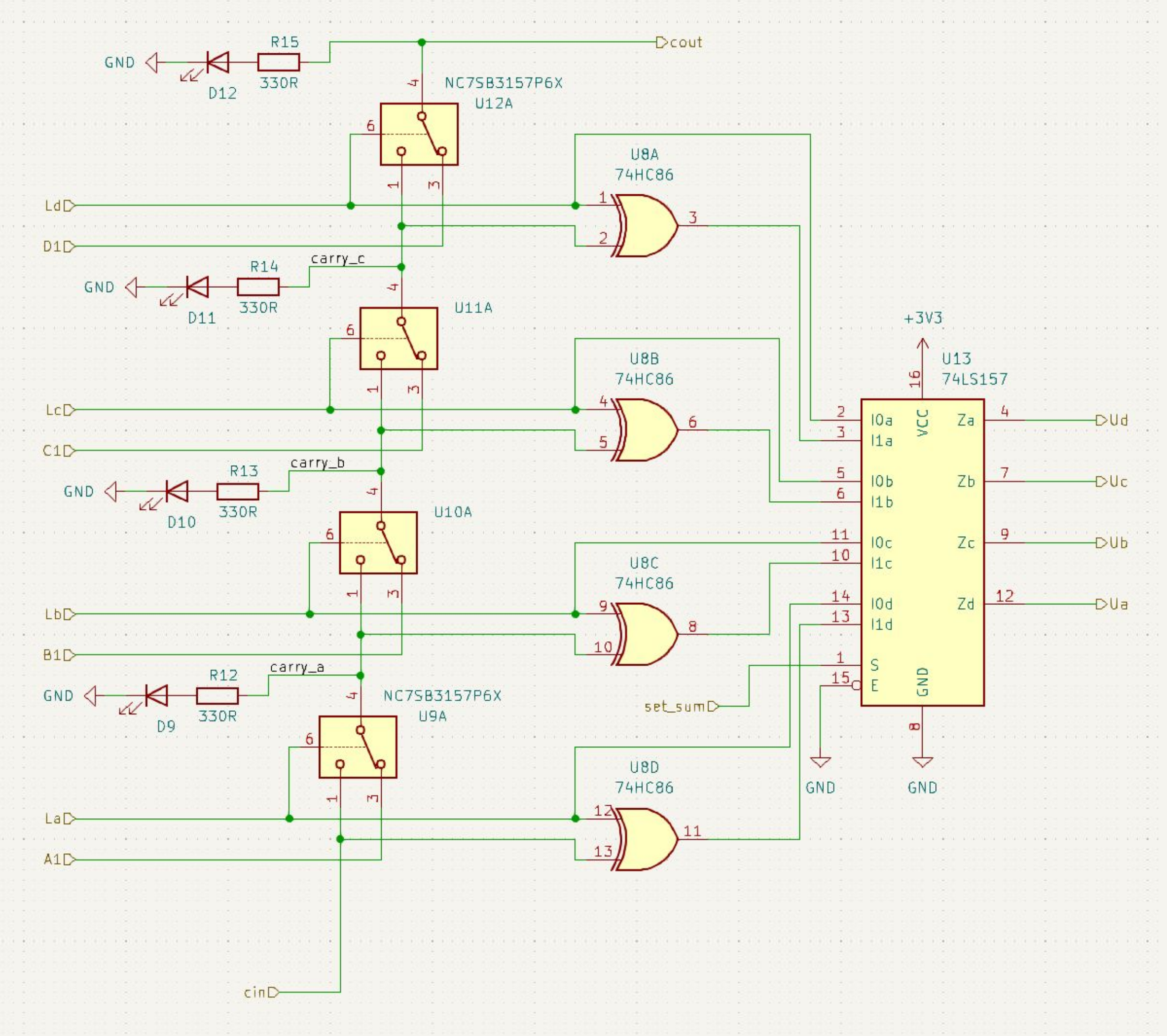
CARRY

74HC157; 74HCT157

Quad 2-input multiplexer

74HC86; 74HCT86

Quad 2-input EXCLUSIVE-OR gate

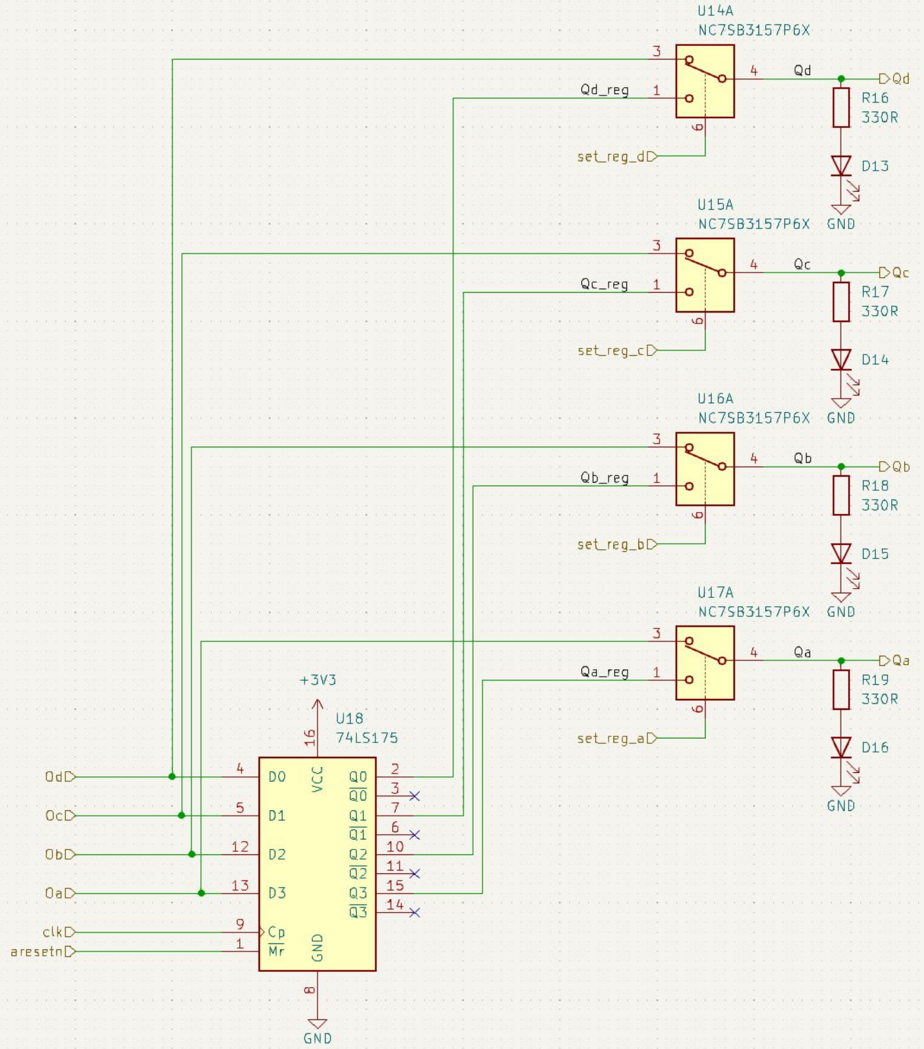


PCB design

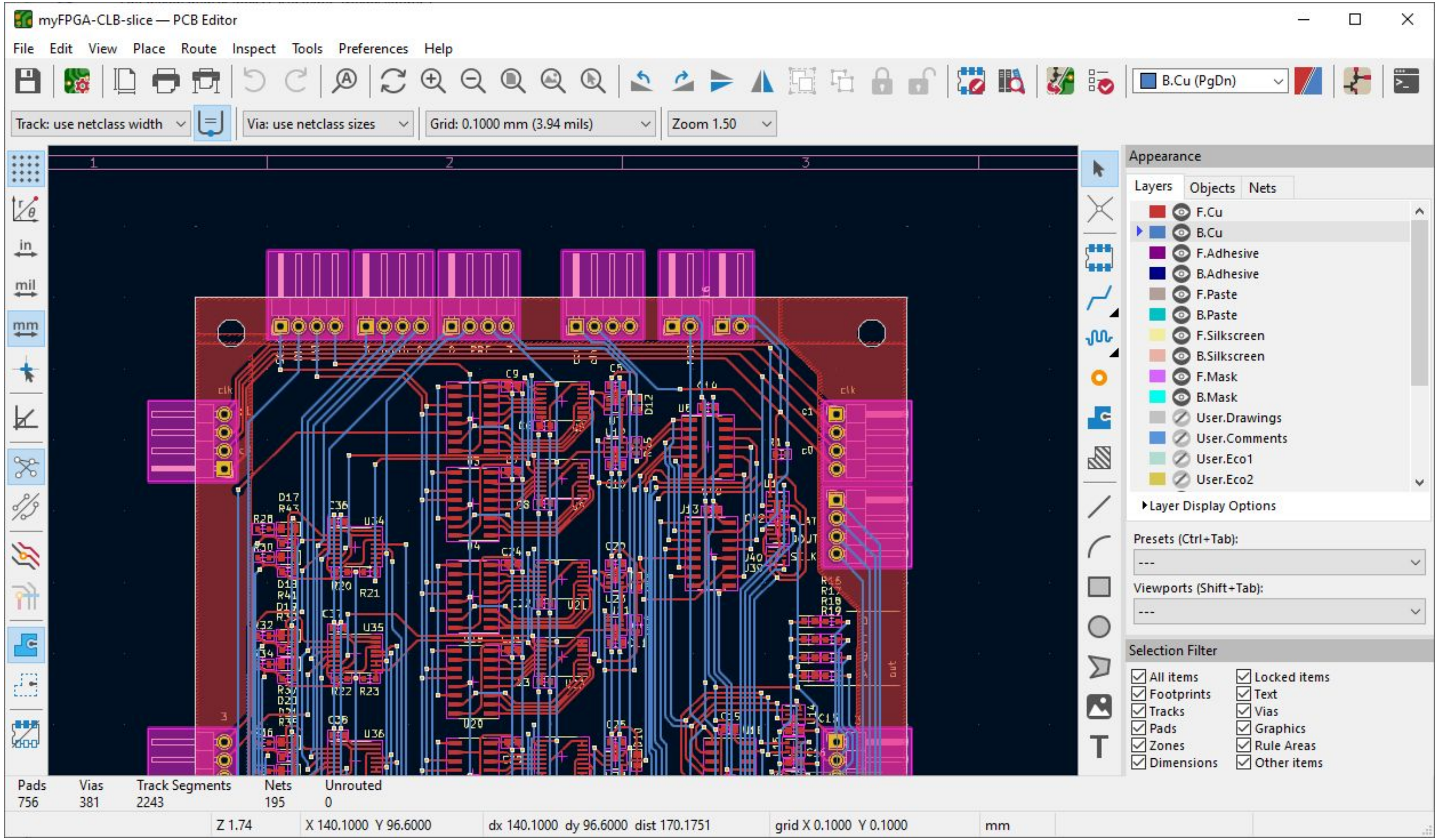
REGISTER

74HC175; 74HCT175

Quad D-type flip-flop with reset; positive-edge trigger



KiCad



myFPGA-CLB-slice — PCB Editor

Track: use netclass width

Via: use netclass sizes

Grid: 0.1000 mm (3.94 mils)

Zoom 1.50

B.Cu (PgDn)

Appearance

Layers Objects Nets

- F.Cu
- B.Cu
- F.Adhesive
- B.Adhesive
- F.Paste
- B.Paste
- F.Silkscreen
- B.Silkscreen
- F.Mask
- B.Mask
- User.Drawings
- User.Comments
- User.Eco1
- User.Eco2

Layer Display Options

Presents (Ctrl+Tab): ---

Viewports (Shift+Tab): ---

Selection Filter

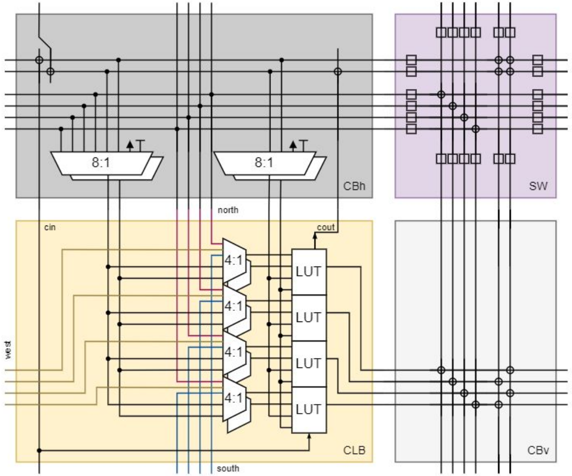
☑ All items ☑ Locked items
☑ Footprints ☑ Text
☑ Tracks ☑ Vias
☑ Pads ☑ Graphics
☑ Zones ☑ Rule Areas
☑ Dimensions ☑ Other items

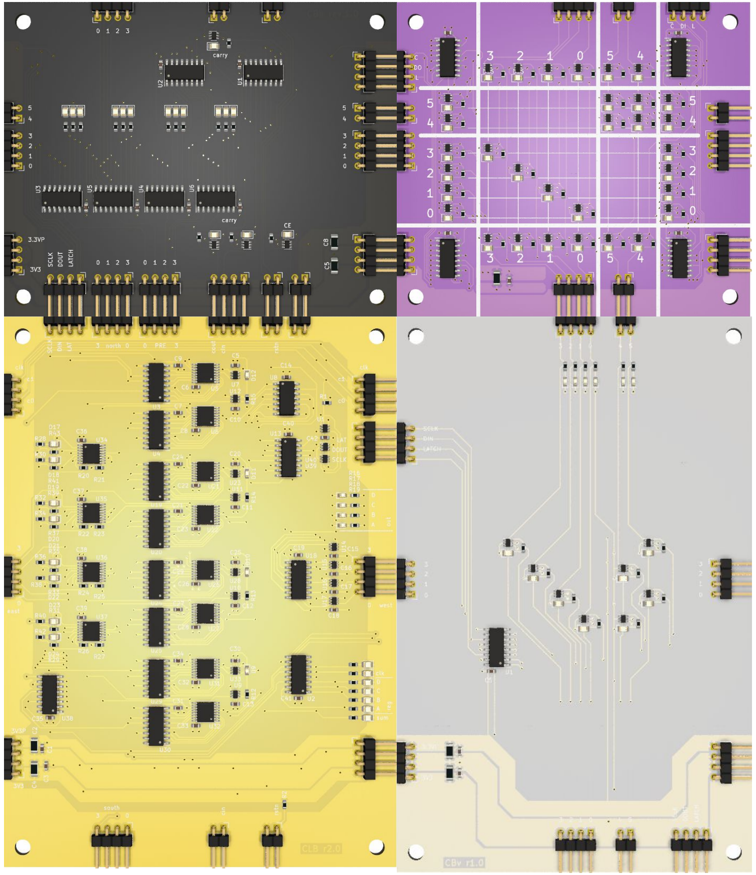
Pads	Vias	Track Segments	Nets	Unrouted
756	381	2243	195	0

Z 1.74 X 140.1000 Y 96.6000 dx 140.1000 dy 96.6000 dist 170.1751 grid X 0.1000 Y 0.1000 mm

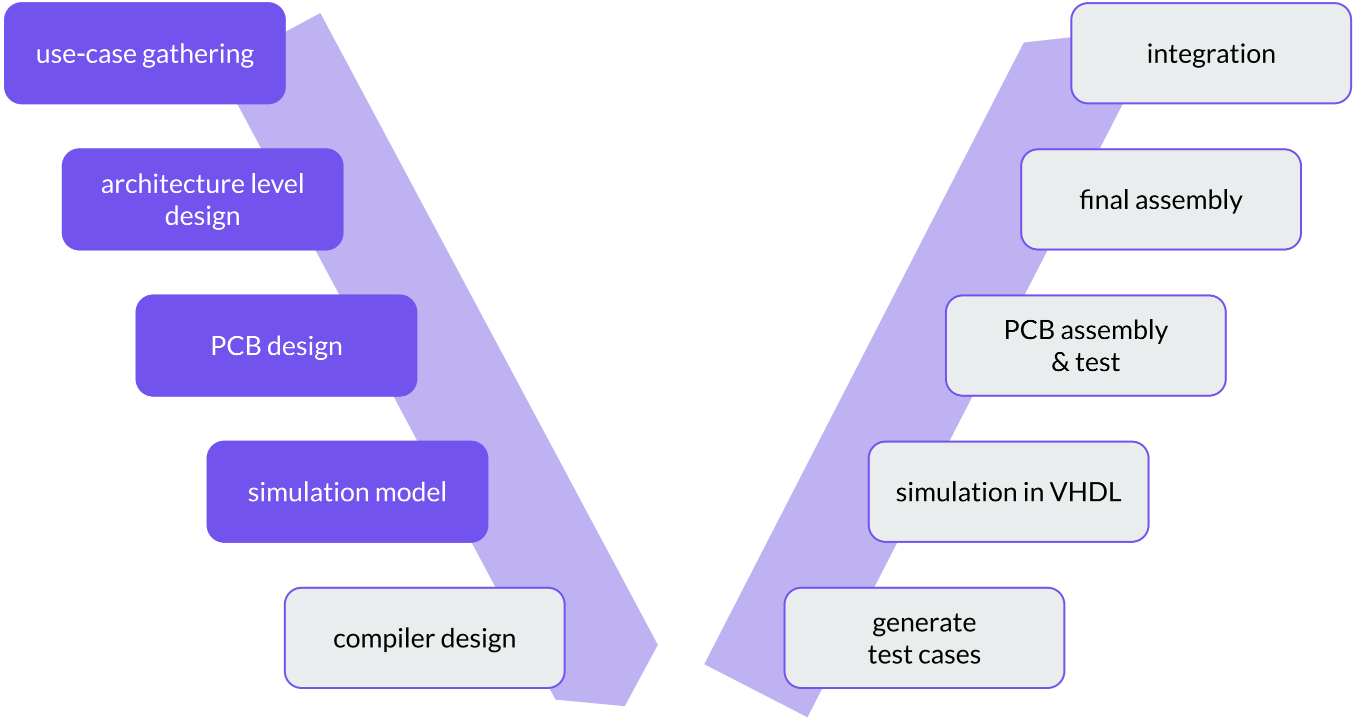
PCB design

Final PCB



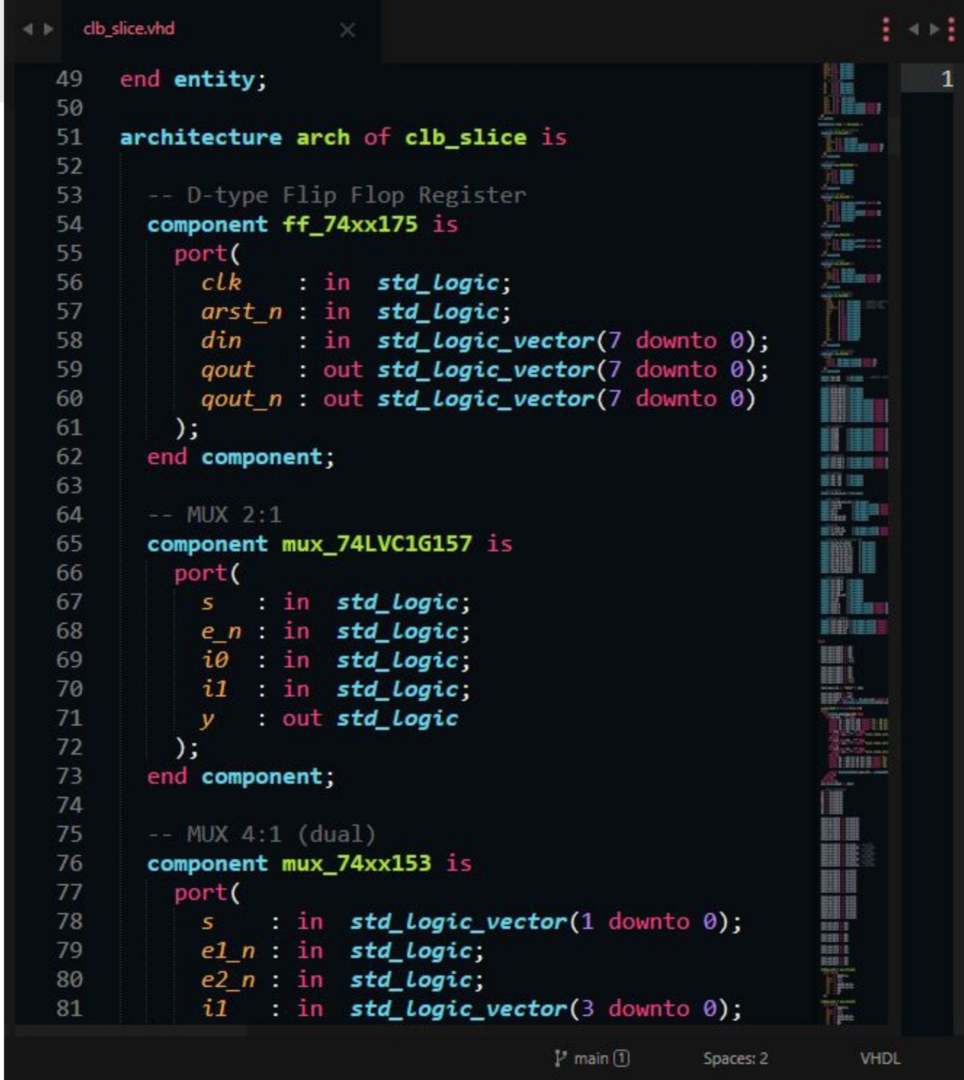


V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
integration
final assembly
PCB assembly & test
simulation in VHDL
generate test cases

VHDL Model

of every 7400 IC on the PCB

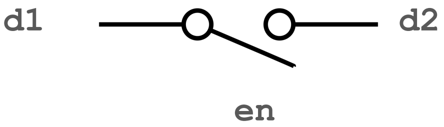


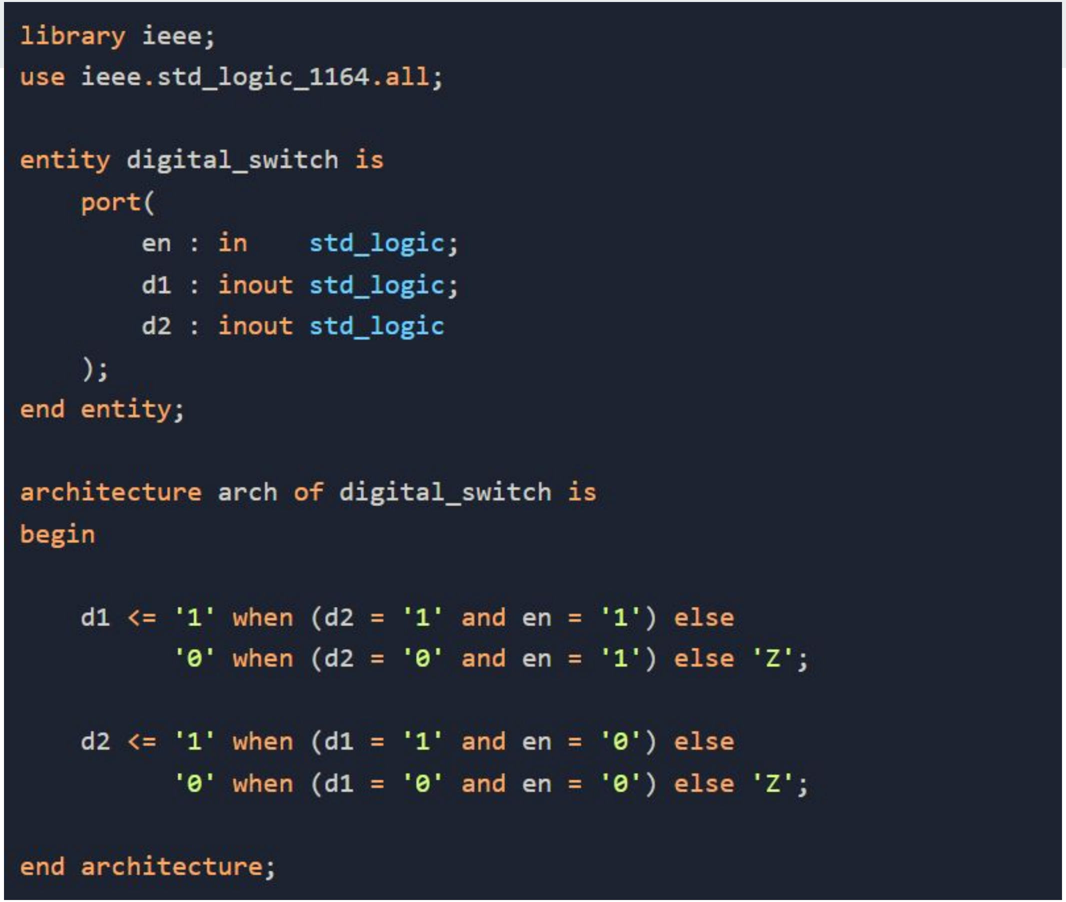
```
end entity;

architecture arch of clb_slice is

  -- D-type Flip Flop Register
  component ff_74xx175 is
    port(
      clk    : in  std_logic;
      arst_n : in  std_logic;
      din    : in  std_logic_vector(7 downto 0);
      qout   : out std_logic_vector(7 downto 0);
      qout_n : out std_logic_vector(7 downto 0)
    );
  end component;

  -- MUX 2:1
  component mux_74LVC1G157 is
    port(
      s   : in  std_logic;
      e_n : in  std_logic;
      i0  : in  std_logic;
      i1  : in  std_logic;
      y   : out std_logic
    );
  end component;

  -- MUX 4:1 (dual)
  component mux_74xx153 is
    port(
      s    : in  std_logic_vector(1 downto 0);
      e1_n : in  std_logic;
      e2_n : in  std_logic;
      i1   : in  std_logic_vector(3 downto 0);
```

simulation model

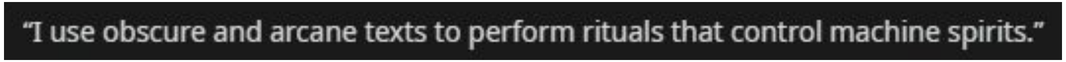
Transistor Level

with VHDL (!)



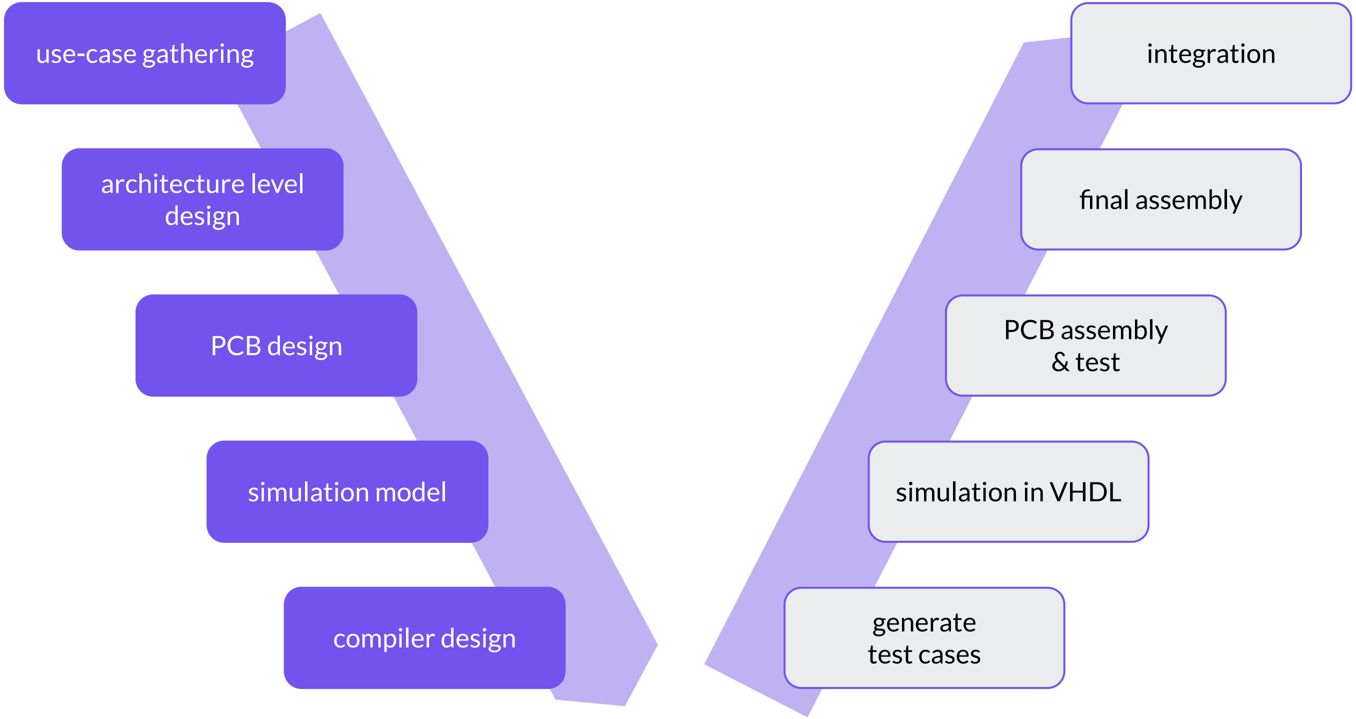


```
library ieee;
use ieee.std_logic_1164.all;

entity digital_switch is
    port(
        en : in    std_logic;
        d1 : inout std_logic;
        d2 : inout std_logic
    );
end entity;

architecture arch of digital_switch is
begin

    d1 <= '1' when (d2 = '1' and en = '1') else
          '0' when (d2 = '0' and en = '1') else 'Z';

    d2 <= '1' when (d1 = '1' and en = '0') else
          '0' when (d1 = '0' and en = '0') else 'Z';

end architecture;
```



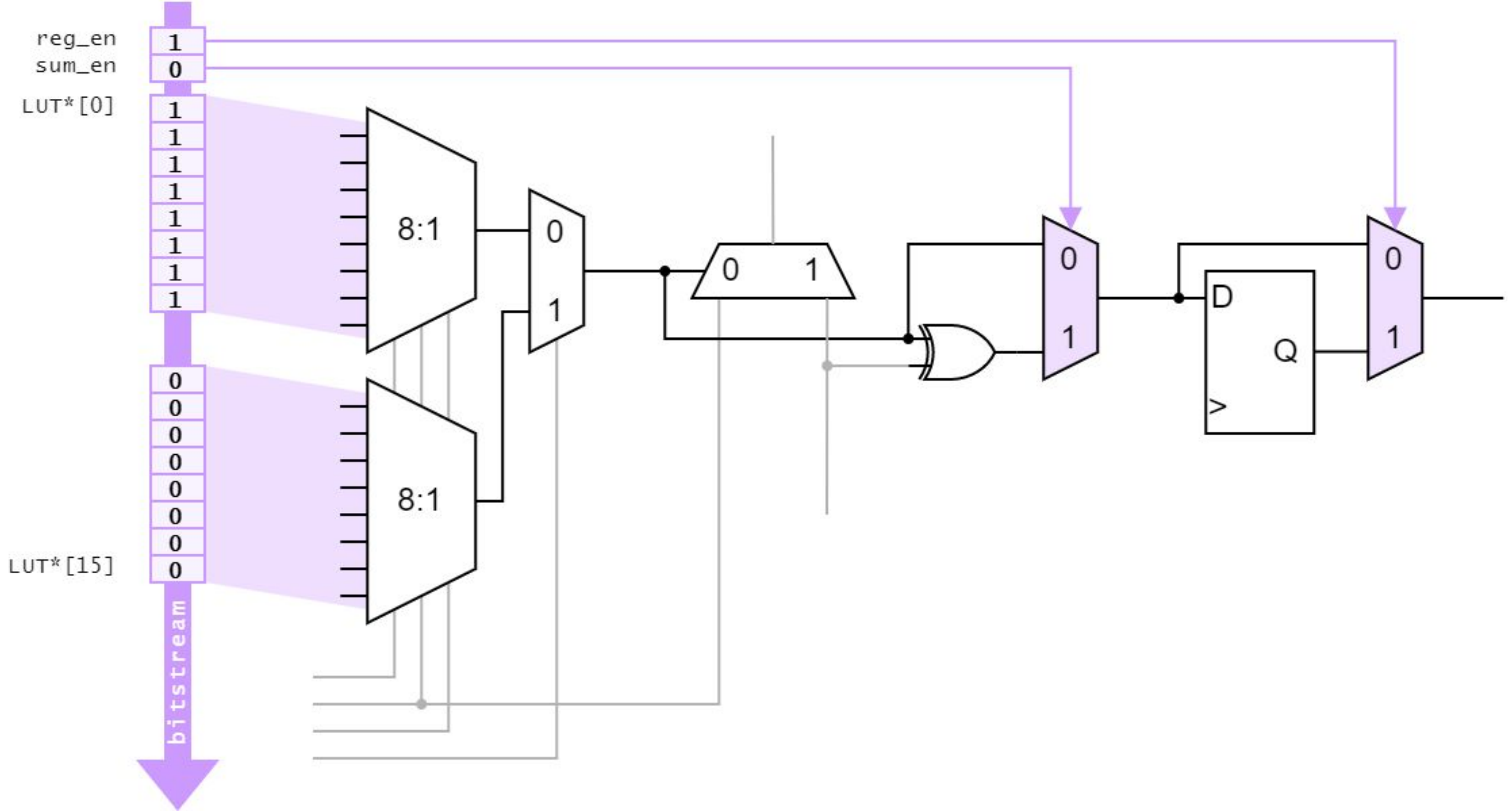
"I use obscure and arcane texts to perform rituals that control machine spirits."

V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
integration
final assembly
PCB assembly & test
simulation in VHDL
generate test cases

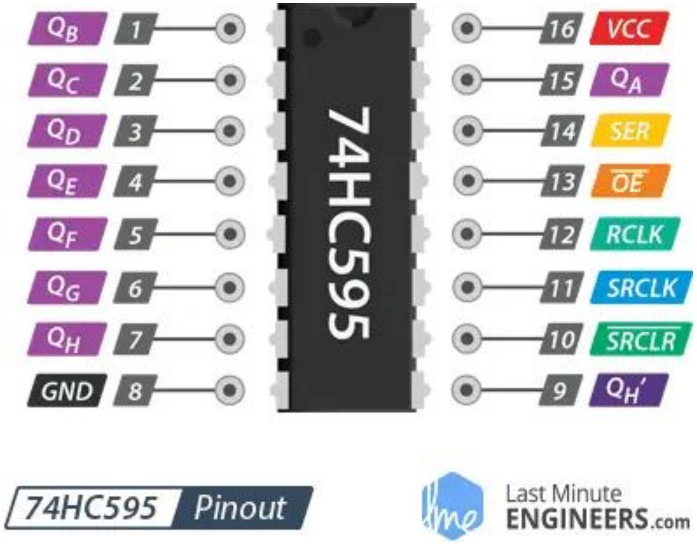
What is a Bitstream?

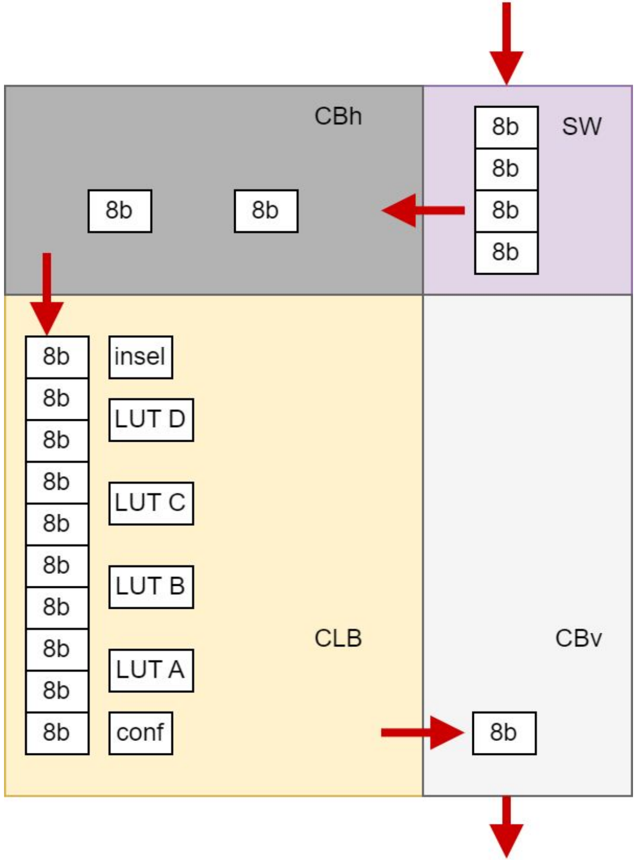
compiler design



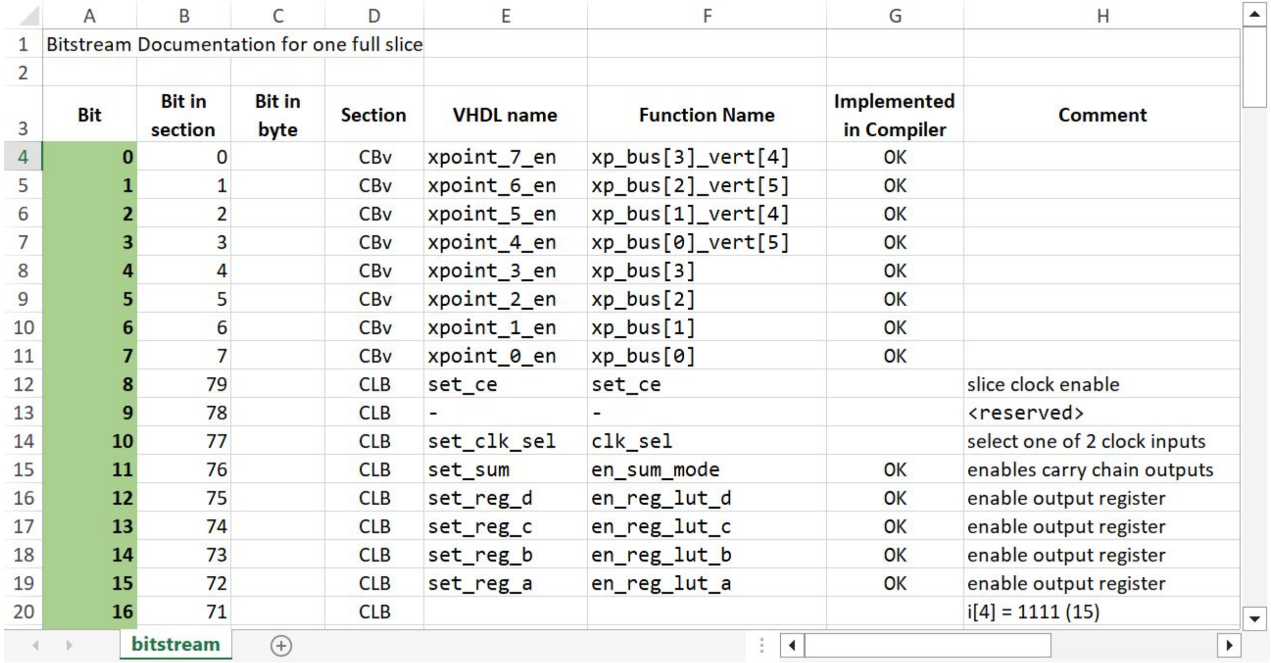
compiler design

Shift Register



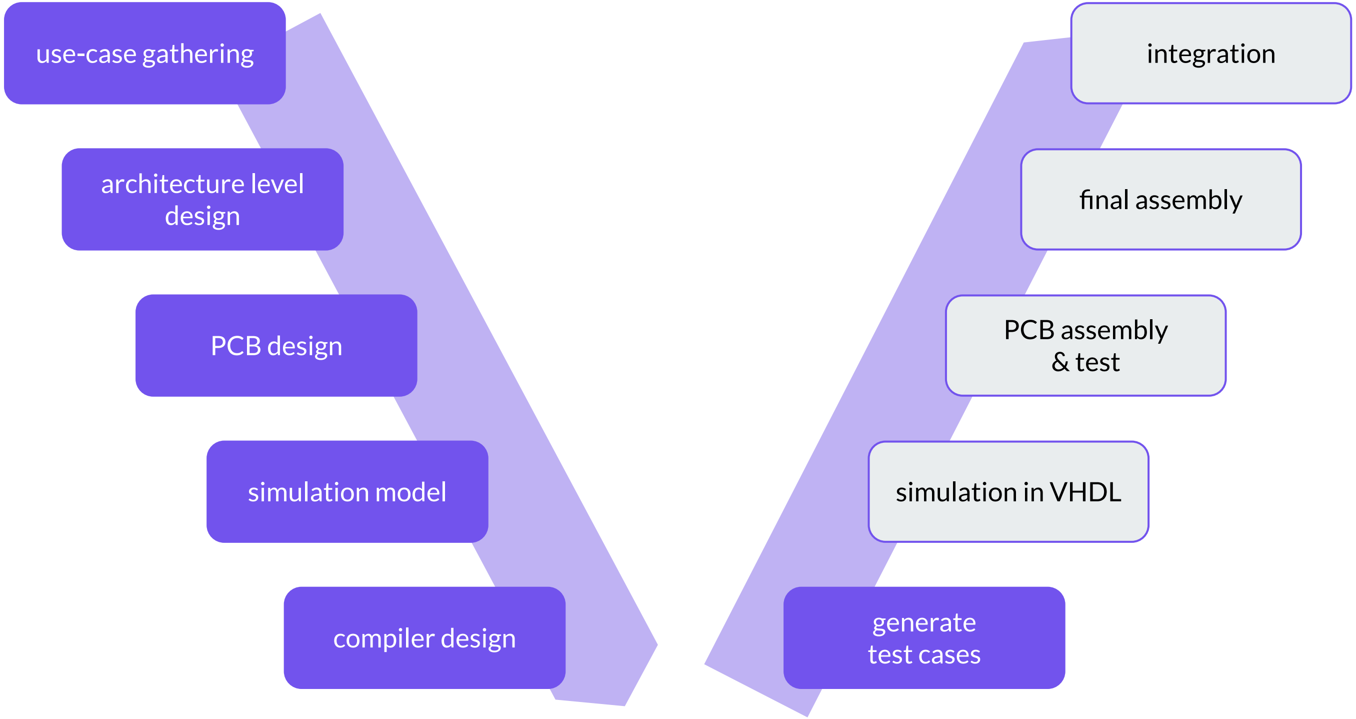


Documentation



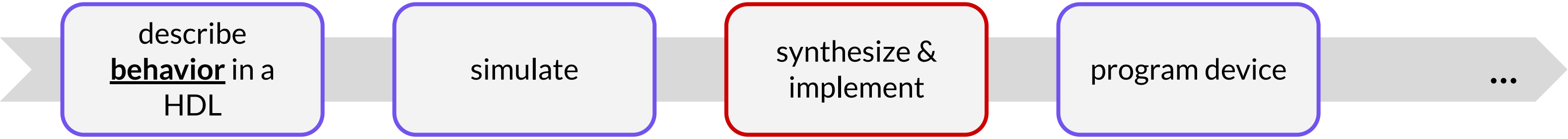
	A	B	C	D	E	F	G	H
1	Bitstream Documentation for one full slice							
2								
3	**Bit**	**Bit in section**	**Bit in byte**	**Section**	**VHDL name**	**Function Name**	**Implemented in Compiler**	**Comment**
4	**0**	0		CBv	xpoint_7_en	xp_bus[3]_vert[4]	OK	
5	**1**	1		CBv	xpoint_6_en	xp_bus[2]_vert[5]	OK	
6	**2**	2		CBv	xpoint_5_en	xp_bus[1]_vert[4]	OK	
7	**3**	3		CBv	xpoint_4_en	xp_bus[0]_vert[5]	OK	
8	**4**	4		CBv	xpoint_3_en	xp_bus[3]	OK	
9	**5**	5		CBv	xpoint_2_en	xp_bus[2]	OK	
10	**6**	6		CBv	xpoint_1_en	xp_bus[1]	OK	
11	**7**	7		CBv	xpoint_0_en	xp_bus[0]	OK	
12	**8**	79		CLB	set_ce	set_ce		slice clock enable
13	**9**	78		CLB	-	-		<reserved>
14	**10**	77		CLB	set_clk_sel	clk_sel		select one of 2 clock inputs
15	**11**	76		CLB	set_sum	en_sum_mode	OK	enables carry chain outputs
16	**12**	75		CLB	set_reg_d	en_reg_lut_d	OK	enable output register
17	**13**	74		CLB	set_reg_c	en_reg_lut_c	OK	enable output register
18	**14**	73		CLB	set_reg_b	en_reg_lut_b	OK	enable output register
19	**15**	72		CLB	set_reg_a	en_reg_lut_a	OK	enable output register
20	**16**	71		CLB				i[4] = 1111 (15)

V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
generate test cases
simulation in VHDL
PCB assembly & test
final assembly
integration

generate
test cases

Regular FPGA workflow



describe **behavior** in a HDL

simulate

synthesize & implement

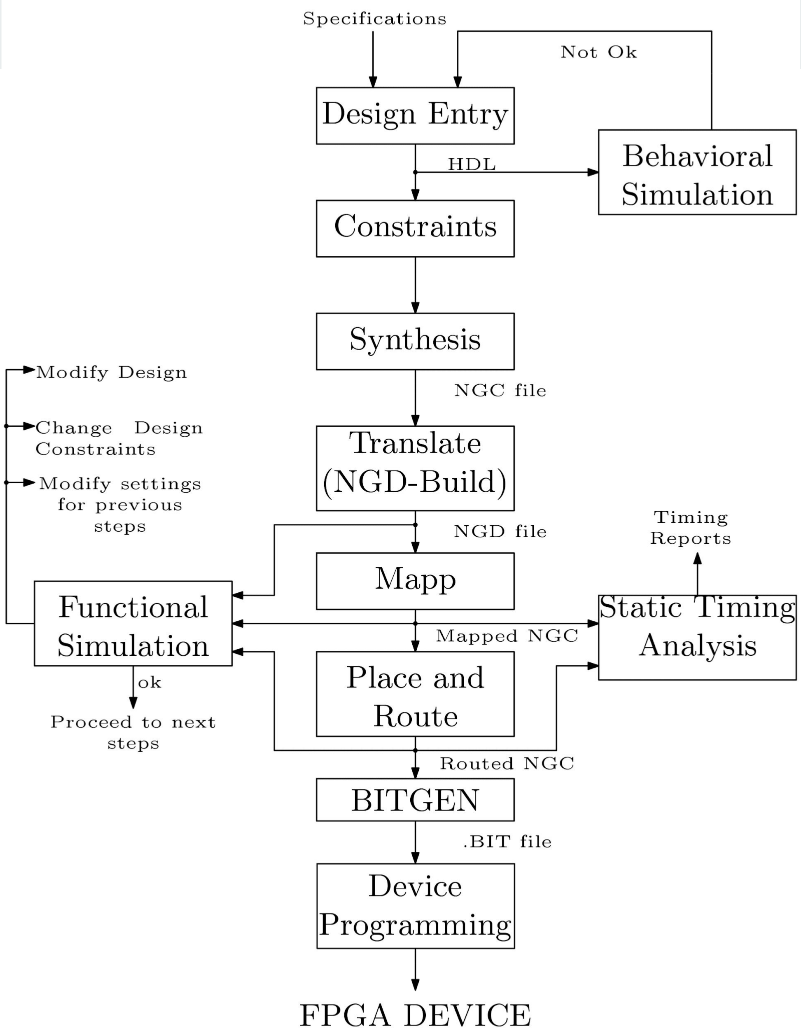
program device

...

https://digitalsystemdesign.in/fpga-implementation-step-by-step

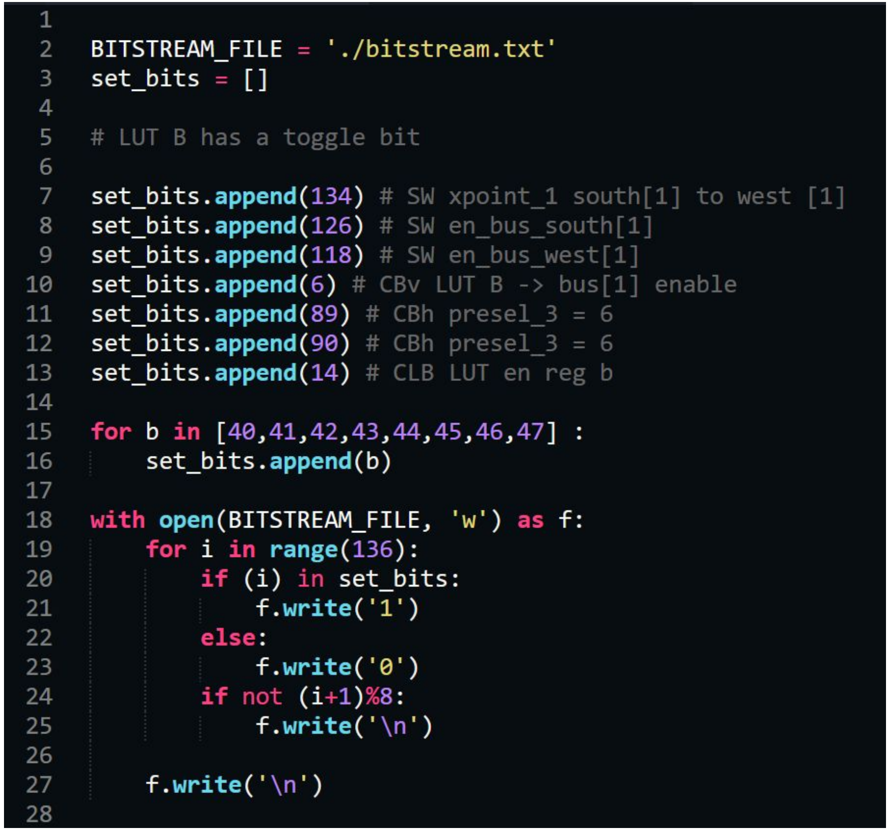
generate
test cases

Synthesis & Implementation



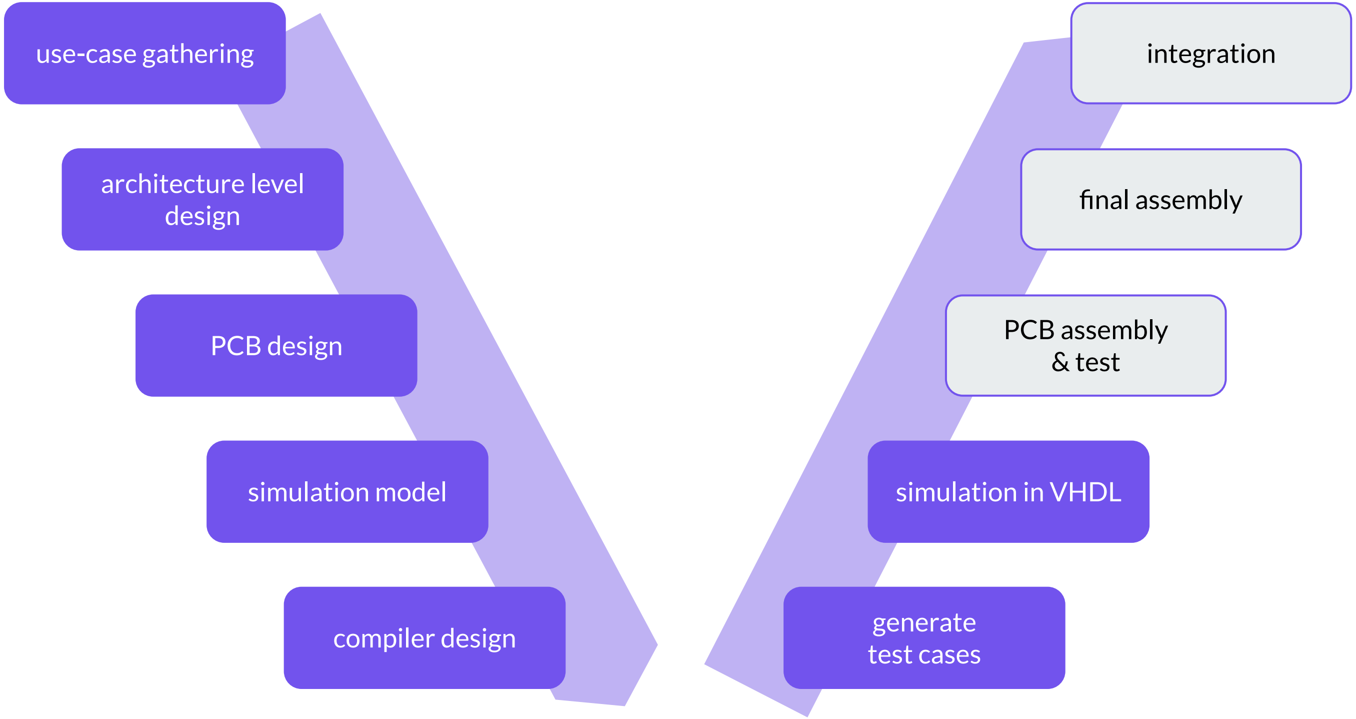
https://digitalsystemdesign.in/fpga-implementation-step-by-step

generate
test cases

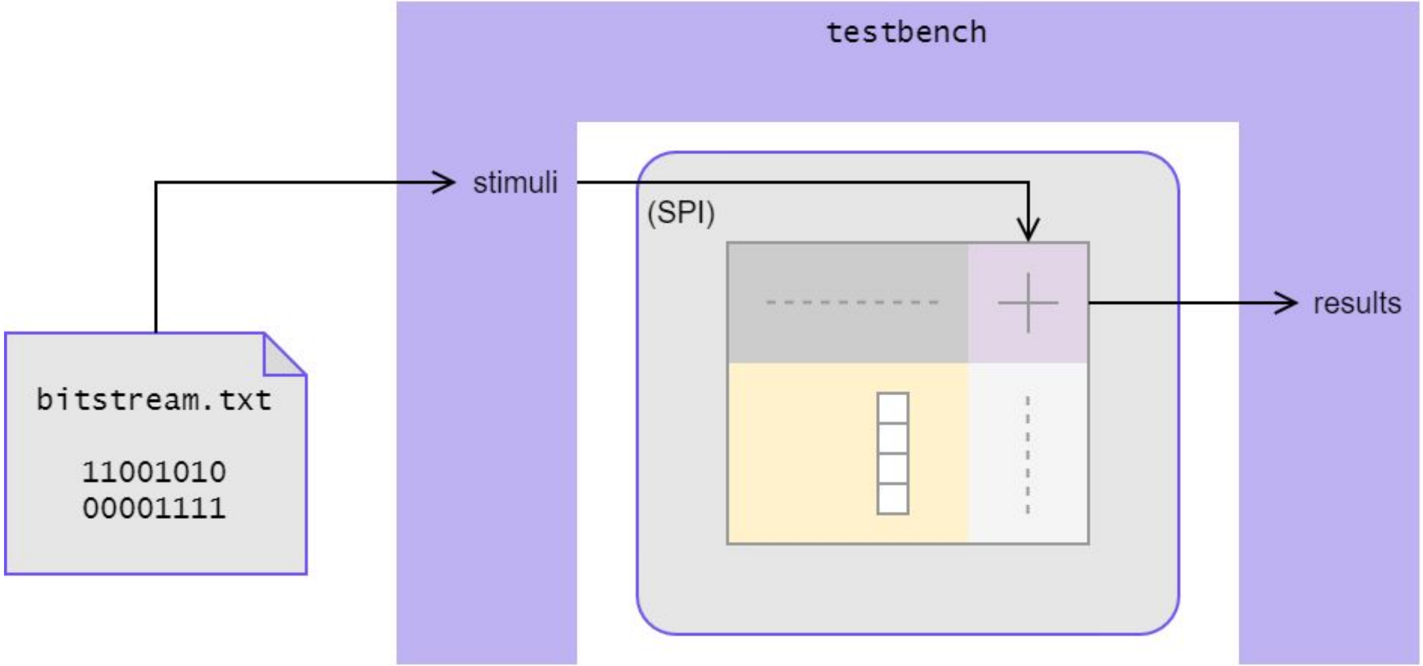


```

BITSTREAM_FILE = './bitstream.txt'
set_bits = []

# LUT B has a toggle bit

set_bits.append(134) # SW xpoint_1 south[1] to west [1]
set_bits.append(126) # SW en_bus_south[1]
set_bits.append(118) # SW en_bus_west[1]
set_bits.append(6) # CBv LUT B -> bus[1] enable
set_bits.append(89) # CBh presel_3 = 6
set_bits.append(90) # CBh presel_3 = 6
set_bits.append(14) # CLB LUT en reg b

for b in [40,41,42,43,44,45,46,47] :
    set_bits.append(b)

with open(BITSTREAM_FILE, 'w') as f:
    for i in range(136):
        if (i) in set_bits:
            f.write('1')
        else:
            f.write('0')
        if not (i+1)%8:
            f.write('\n')

    f.write('\n')

```

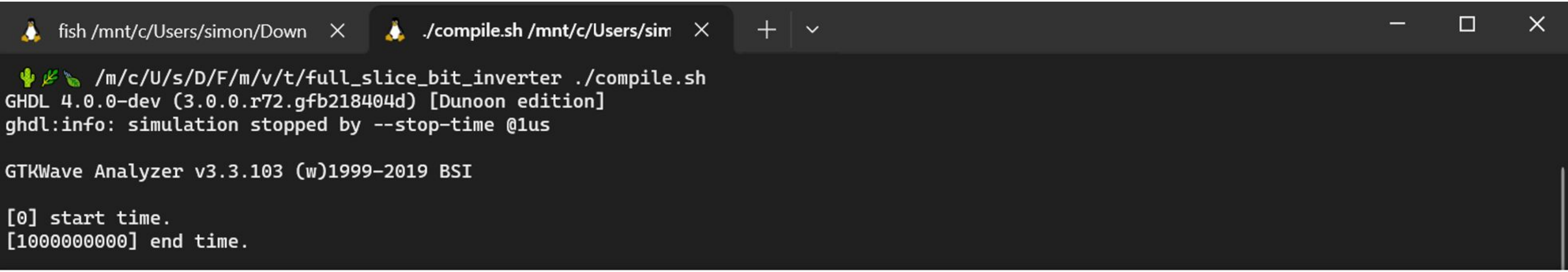
V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
generate test cases
simulation in VHDL
PCB assembly & test
final assembly
integration

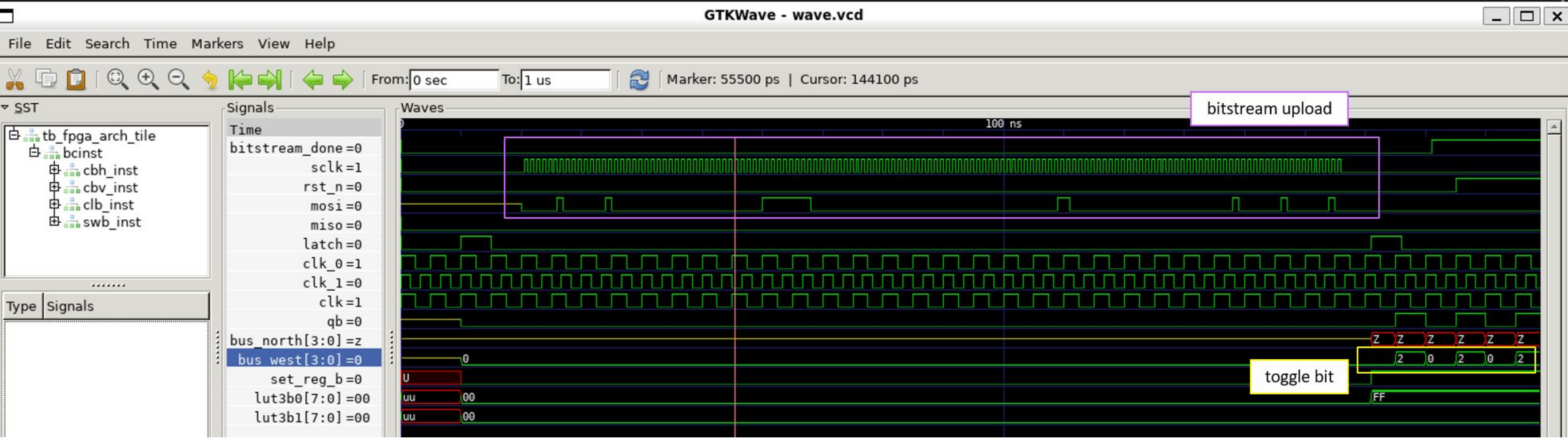
simulation in VHDL



simulation in VHDL



```
🌵🌿🍃 /m/c/U/s/D/F/m/v/t/full_slice_bit_inverter ./compile.sh
GHDL 4.0.0-dev (3.0.0.r72.gfb218404d) [Dunoon edition]
ghdl:info: simulation stopped by --stop-time @1us

GTKWave Analyzer v3.3.103 (w)1999-2019 BSI

[0] start time.
[1000000000] end time.
```



From: 0 sec To: 1 us | Marker: 55500 ps | Cursor: 144100 ps

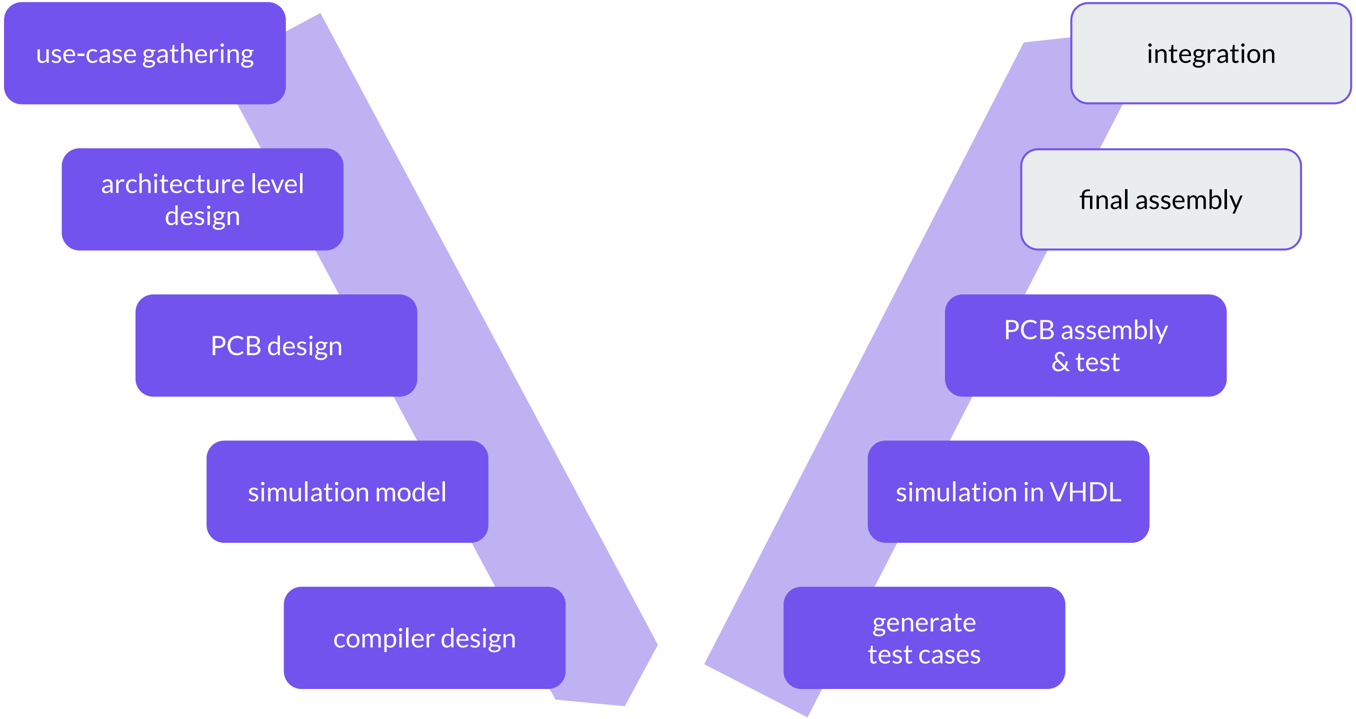
```
tb_fpga_arch_tile
  bcinst
    cbh_inst
    cbv_inst
    clb_inst
    swb_inst
```

```
Time
bitstream_done =0
sclk =1
rst_n =0
mosi =0
miso =0
latch =0
clk_0 =1
clk_1 =0
clk =1
qb =0
bus_north[3:0] =z
bus_west[3:0] =0
set_reg_b =0
lut3b0[7:0] =00
lut3b1[7:0] =00
```

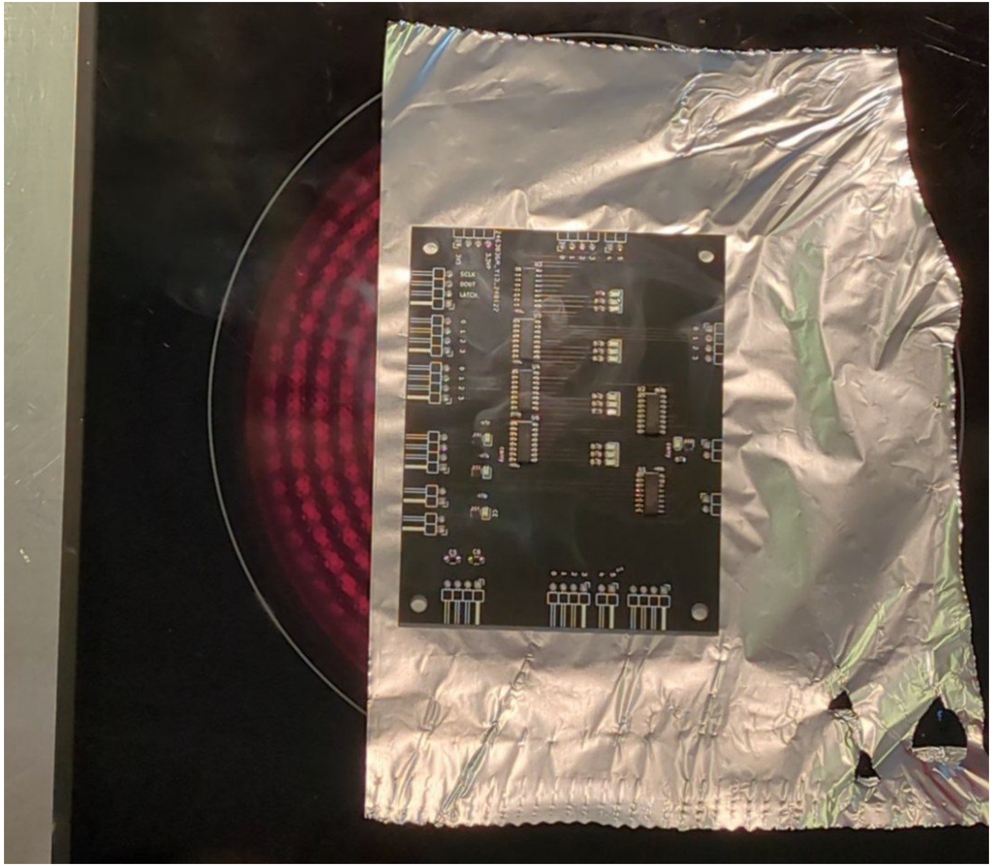
bitstream upload

toggle bit

V-Model

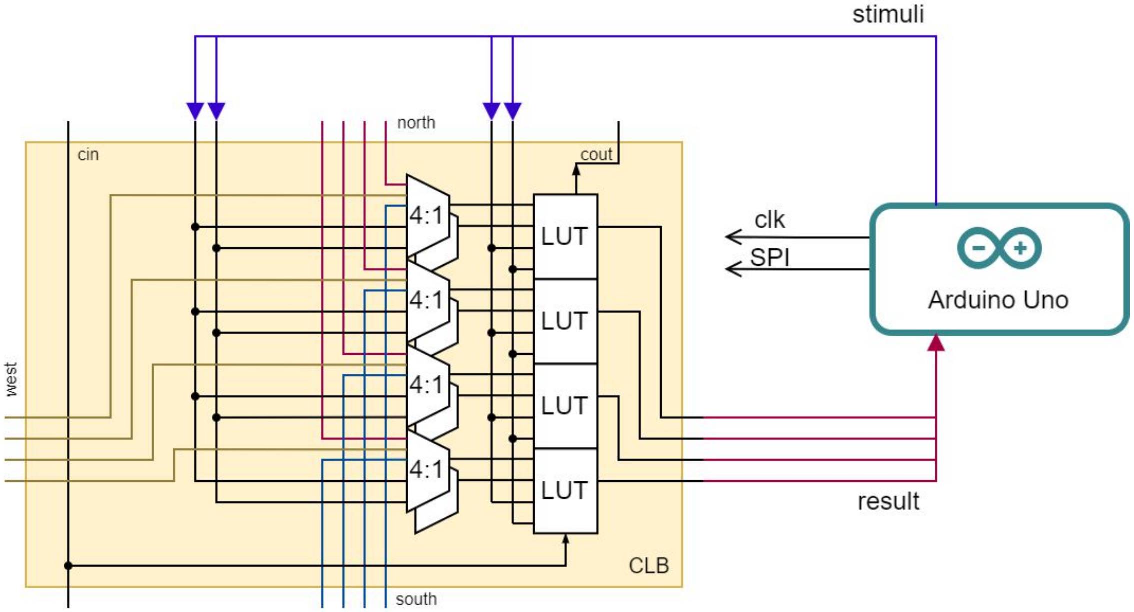

use-case gathering
architecture level design
PCB design
simulation model
compiler design
integration
final assembly
PCB assembly & test
simulation in VHDL
generate test cases

“don’t try this at home”

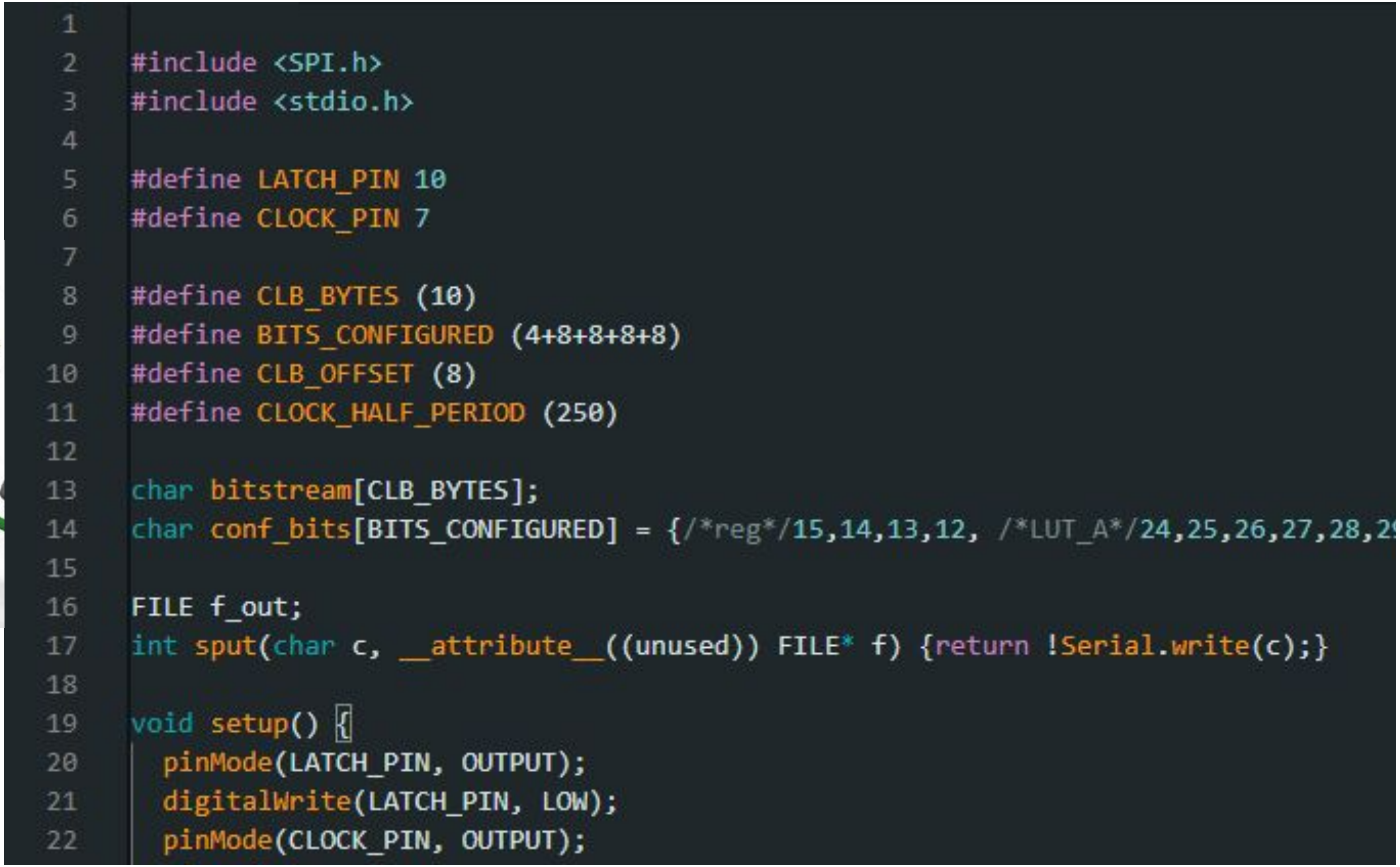


PCB assembly
& test

Testbench

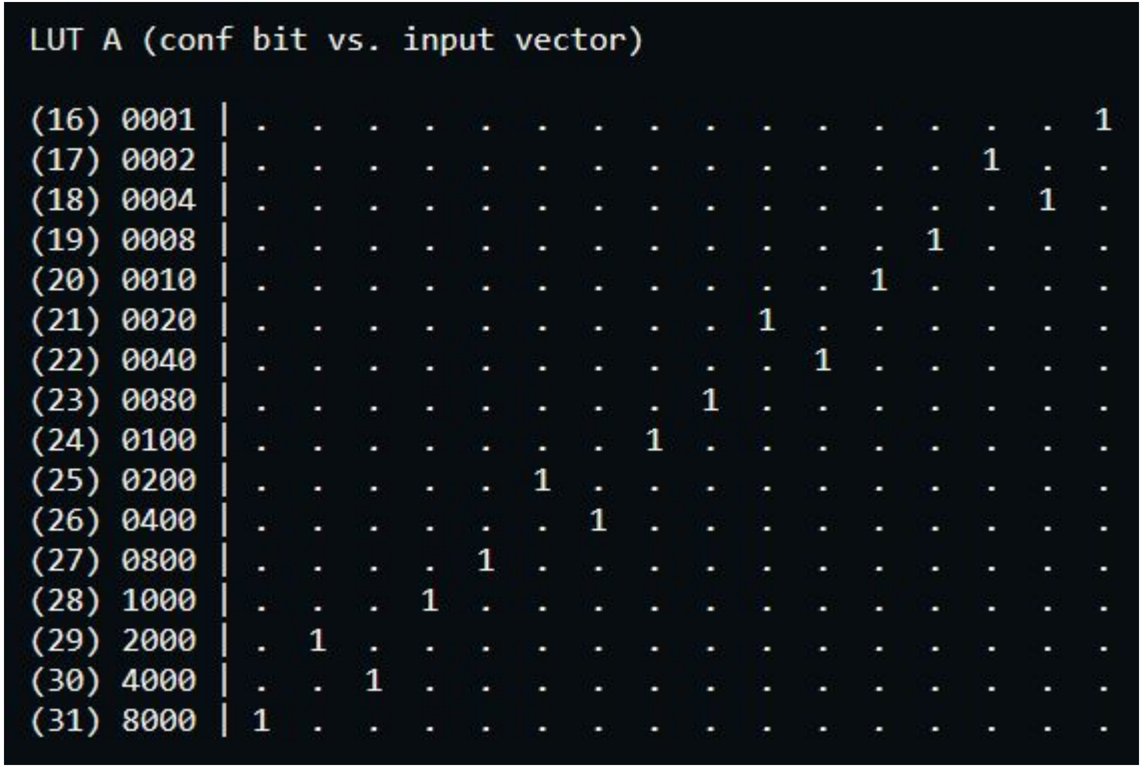


PCB assembly & test



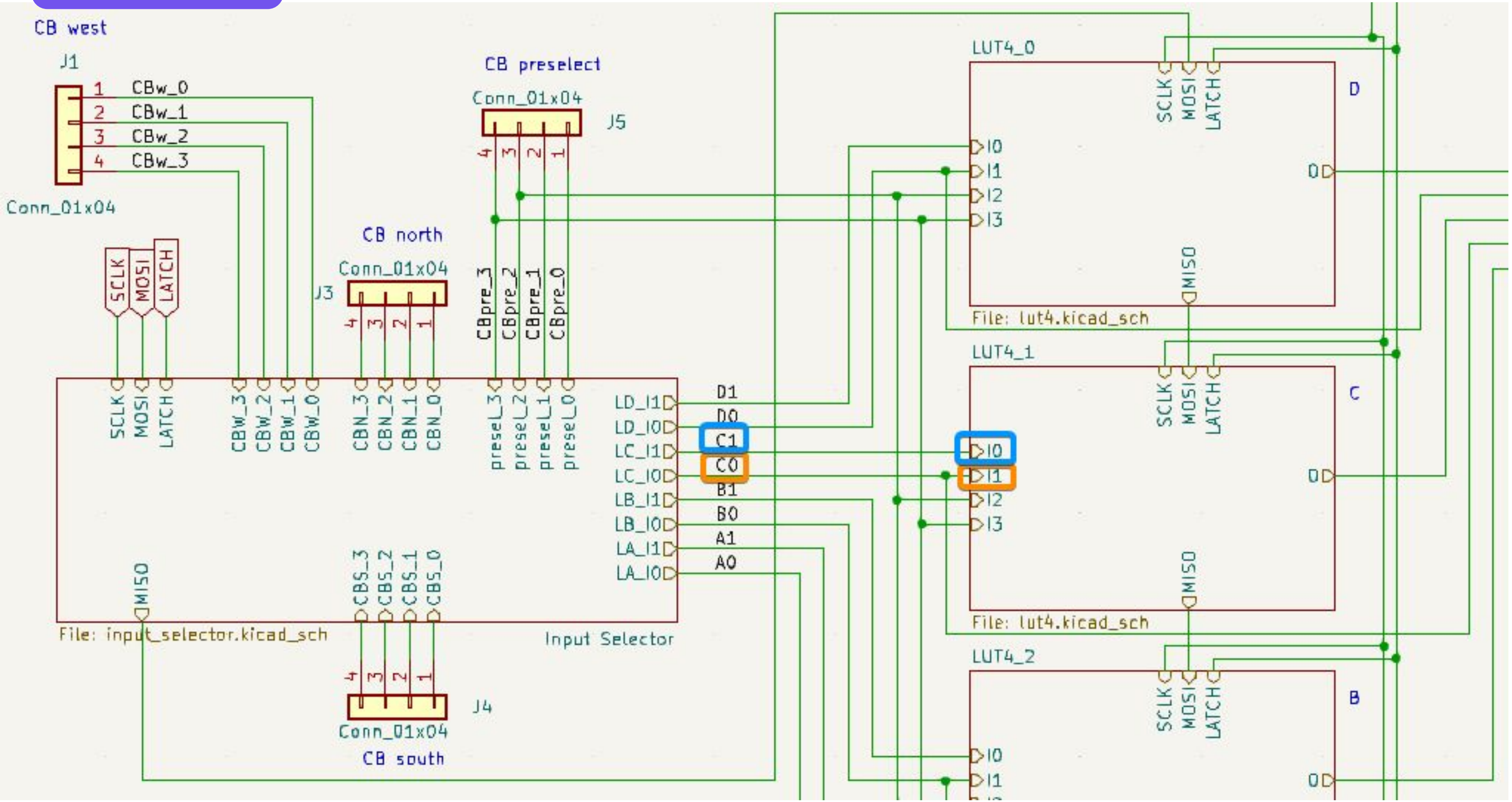
```

#include <SPI.h>
#include <stdio.h>

#define LATCH_PIN 10
#define CLOCK_PIN 7

#define CLB_BYTES (10)
#define BITS_CONFIGURED (4+8+8+8+8)
#define CLB_OFFSET (8)
#define CLOCK_HALF_PERIOD (250)

char bitstream[CLB_BYTES];
char conf_bits[BITS_CONFIGURED] = {/*reg*/15,14,13,12, /*LUT_A*/24,25,26,27,28,2

FILE f_out;
int sput(char c, __attribute__((unused)) FILE* f) {return !Serial.write(c);}

void setup() {
  pinMode(LATCH_PIN, OUTPUT);
  digitalWrite(LATCH_PIN, LOW);
  pinMode(CLOCK_PIN, OUTPUT);
```

oops...



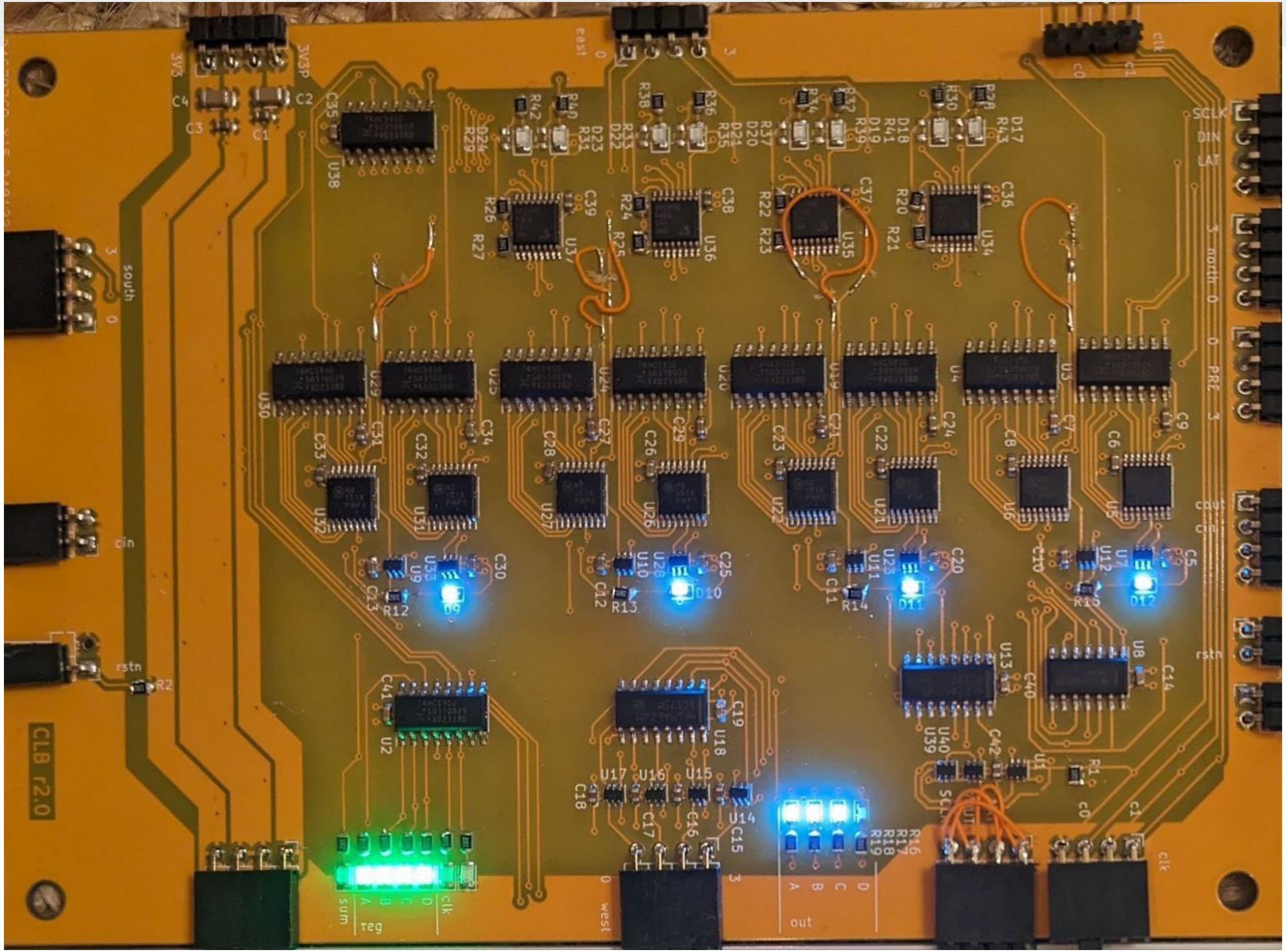
```
LUT A (conf bit vs. input vector)

(16) 0001 | .  .  .  .  .  .  .  .  .  .  .  .  .  .  .  1
(17) 0002 | .  .  .  .  .  .  .  .  .  .  .  .  .  1  .  .
(18) 0004 | .  .  .  .  .  .  .  .  .  .  .  .  .  .  1  .
(19) 0008 | .  .  .  .  .  .  .  .  .  .  .  .  1  .  .  .
(20) 0010 | .  .  .  .  .  .  .  .  .  .  .  1  .  .  .  .
(21) 0020 | .  .  .  .  .  .  .  .  .  1  .  .  .  .  .  .
(22) 0040 | .  .  .  .  .  .  .  .  .  .  1  .  .  .  .  .
(23) 0080 | .  .  .  .  .  .  .  .  1  .  .  .  .  .  .  .
(24) 0100 | .  .  .  .  .  .  .  1  .  .  .  .  .  .  .  .
(25) 0200 | .  .  .  .  .  1  .  .  .  .  .  .  .  .  .  .
(26) 0400 | .  .  .  .  .  .  1  .  .  .  .  .  .  .  .  .
(27) 0800 | .  .  .  .  1  .  .  .  .  .  .  .  .  .  .  .
(28) 1000 | .  .  .  1  .  .  .  .  .  .  .  .  .  .  .  .
(29) 2000 | .  1  .  .  .  .  .  .  .  .  .  .  .  .  .  .
(30) 4000 | .  .  1  .  .  .  .  .  .  .  .  .  .  .  .  .
(31) 8000 | 1  .  .  .  .  .  .  .  .  .  .  .  .  .  .  .
```

PCB assembly
& test

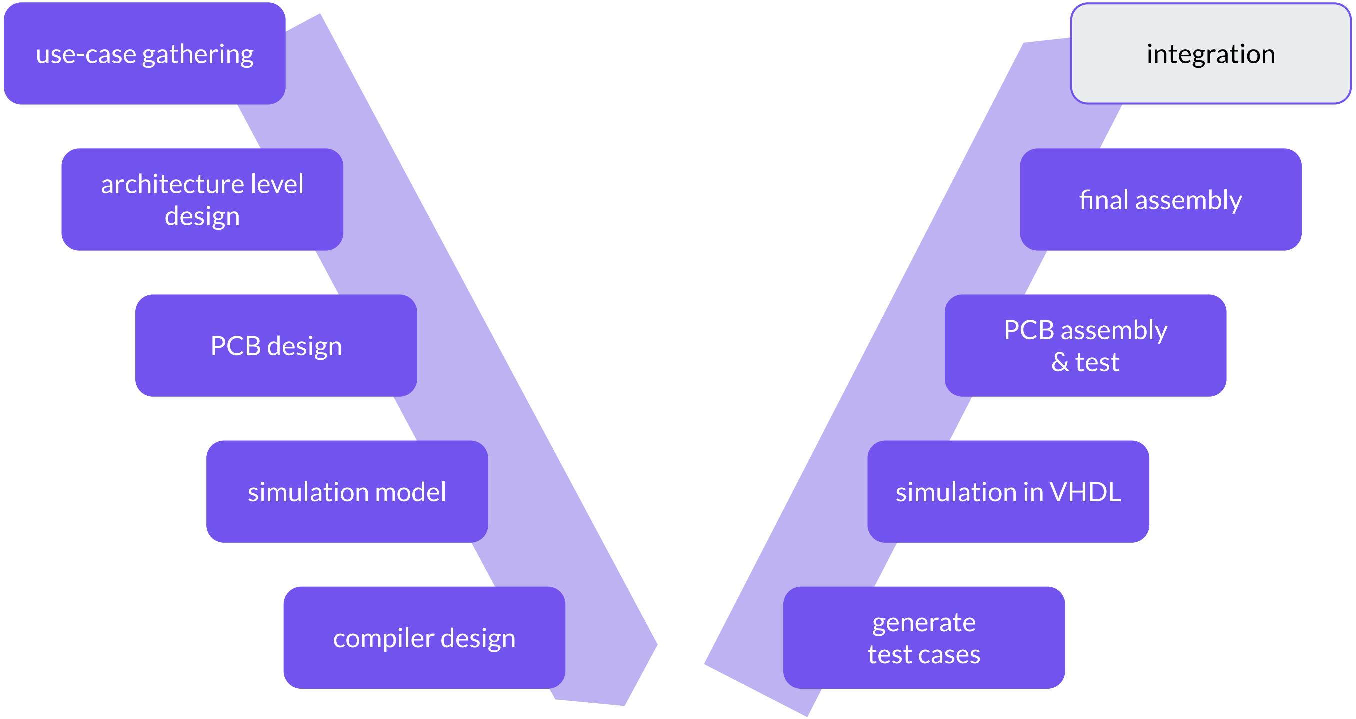


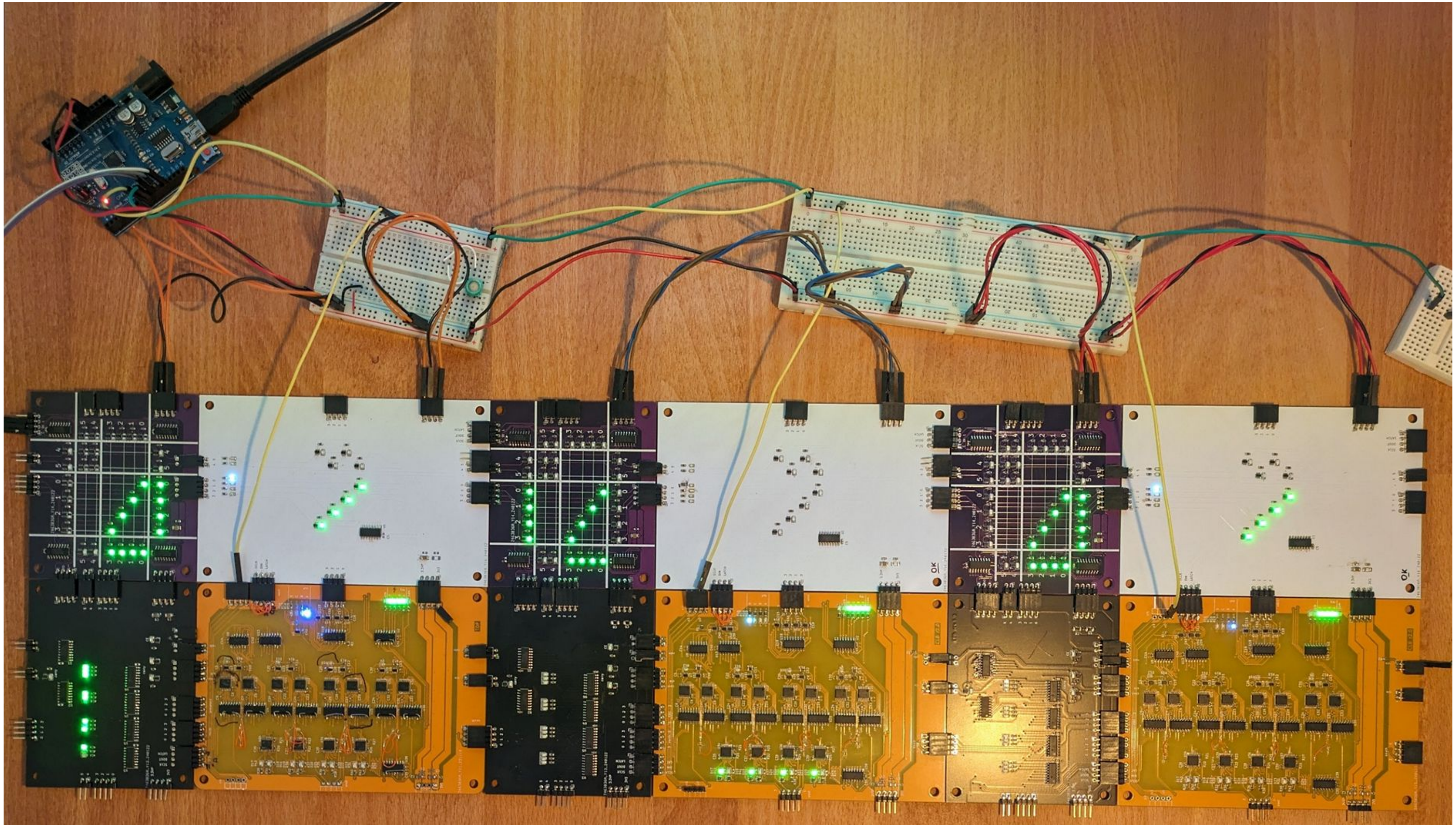
PCB assembly
& test

mod wires

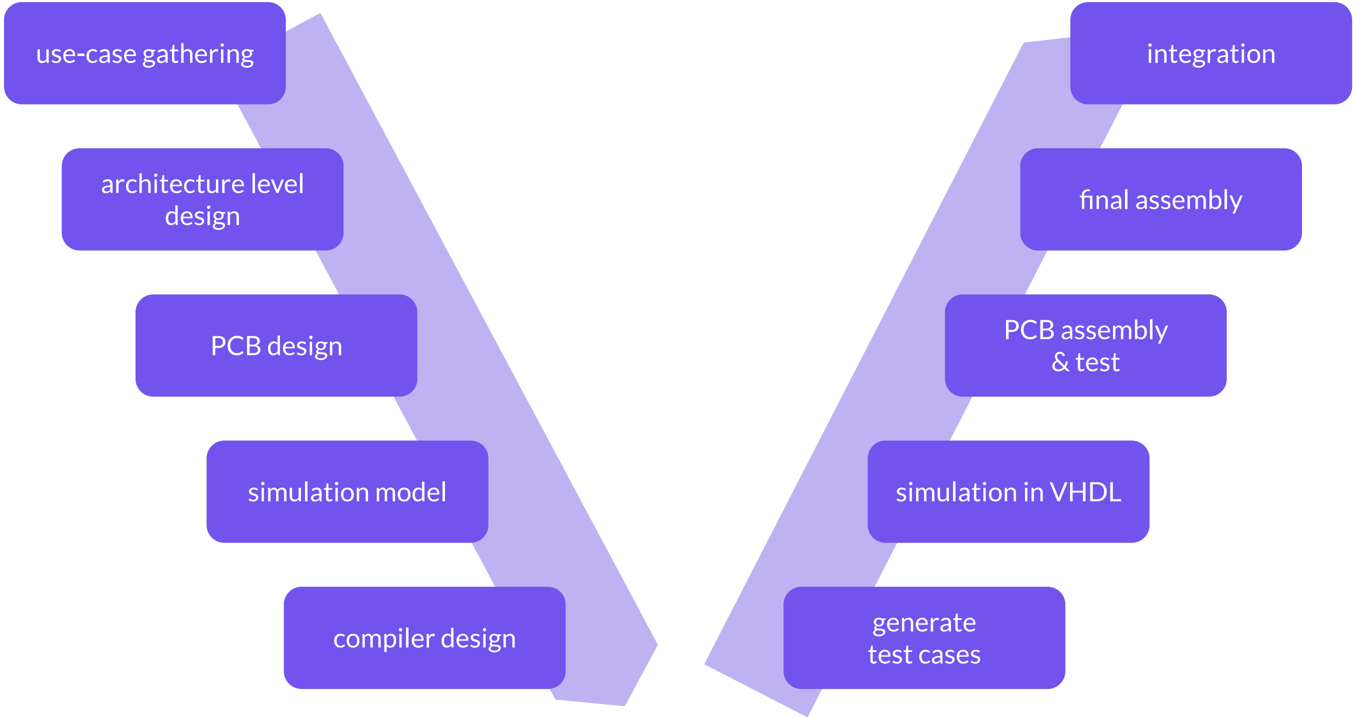


V-Model


use-case gathering
architecture level design
PCB design
simulation model
compiler design
generate test cases
simulation in VHDL
PCB assembly & test
final assembly
integration



V-Model



use-case gathering

architecture level design

PCB design

simulation model

compiler design

integration

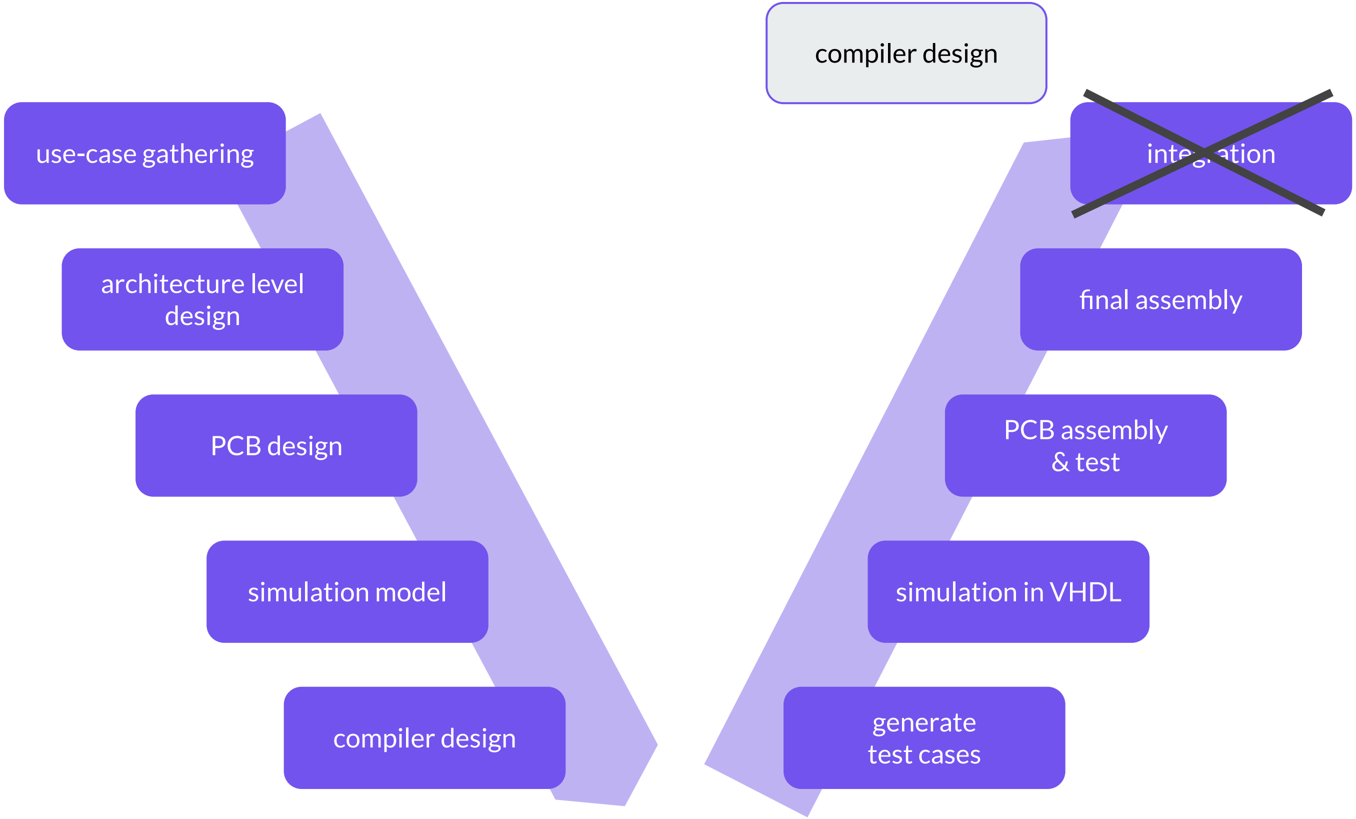
final assembly

PCB assembly & test

simulation in VHDL

generate test cases

V-Model



use-case gathering

architecture level design

PCB design

simulation model

compiler design

compiler design

~~integration~~

final assembly

PCB assembly & test

simulation in VHDL

generate test cases

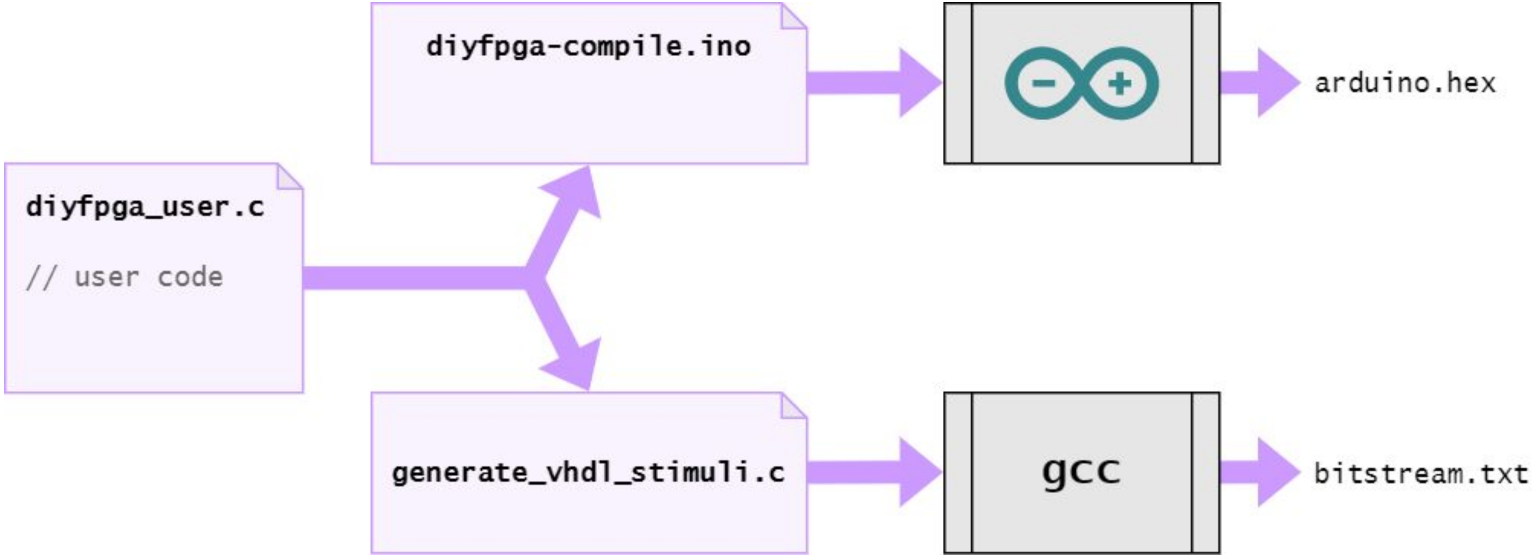
compiler design

Hardware Instantiation in C

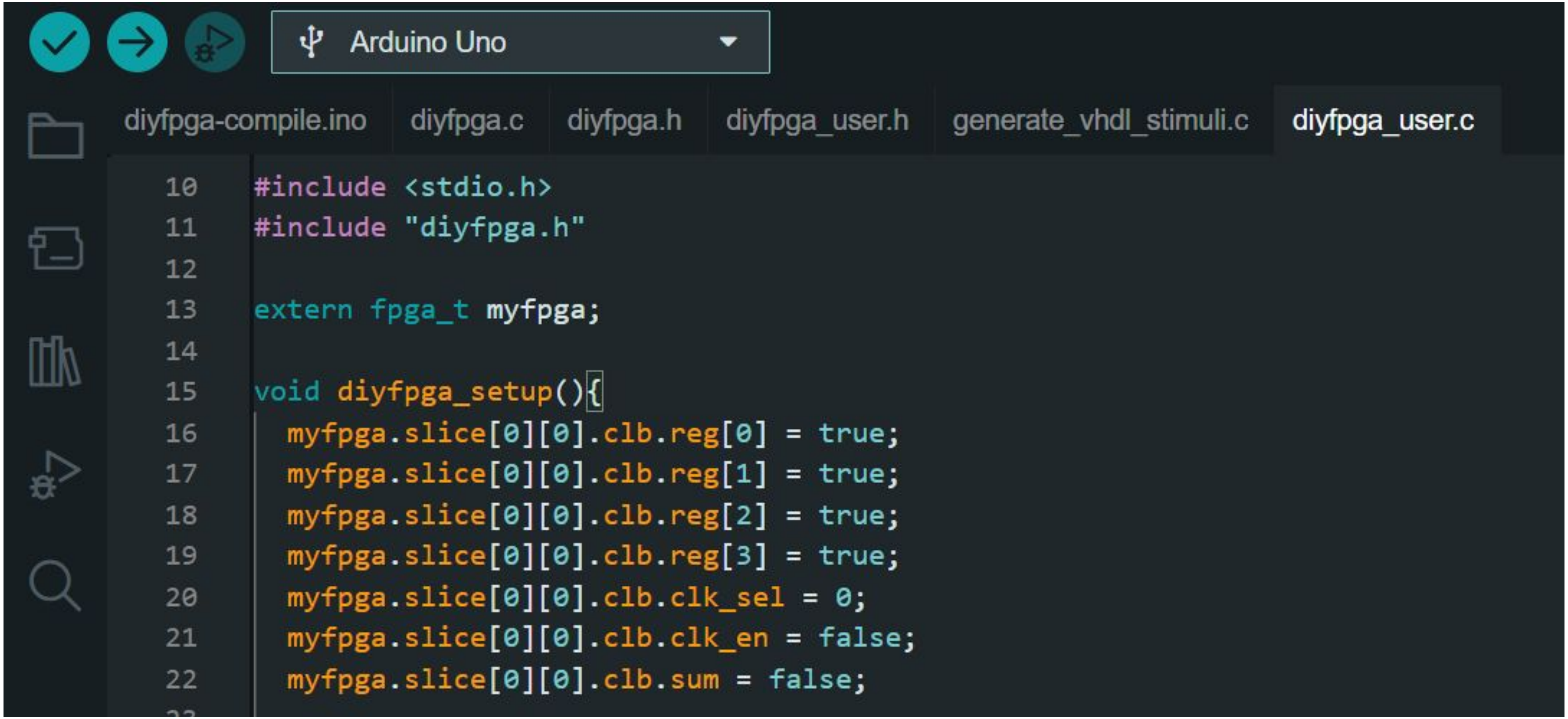
- *“know what you ~~infer~~ instantiate”*
- kinda like assembly for FPGA

https://docs.amd.com/r/en-US/ug949-vivado-design-methodology/Know-What-You-Infer

Same Code - Different Targets



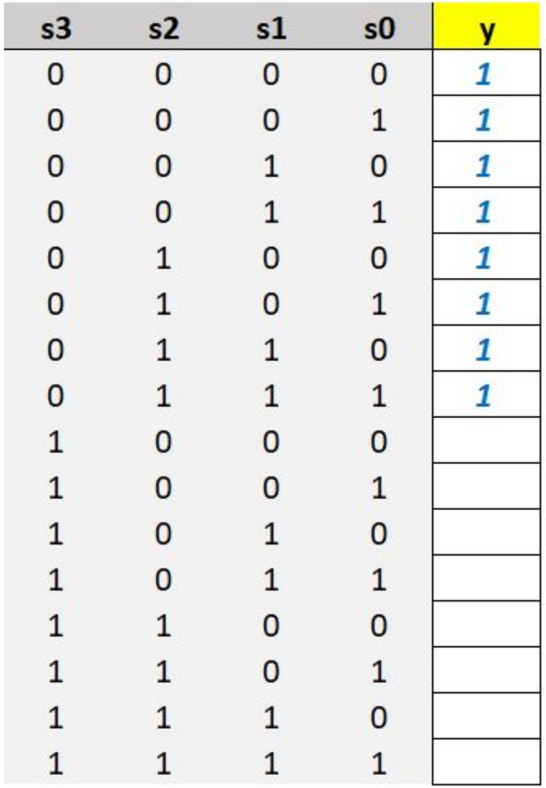
compiler design



```
#include <stdio.h>
#include "diyfpga.h"

extern fpga_t myfpga;

void diyfpga_setup(){
  myfpga.slice[0][0].clb.reg[0] = true;
  myfpga.slice[0][0].clb.reg[1] = true;
  myfpga.slice[0][0].clb.reg[2] = true;
  myfpga.slice[0][0].clb.reg[3] = true;
  myfpga.slice[0][0].clb.clk_sel = 0;
  myfpga.slice[0][0].clb.clk_en = false;
  myfpga.slice[0][0].clb.sum = false;
```

compiler design

Truth Table to HEX

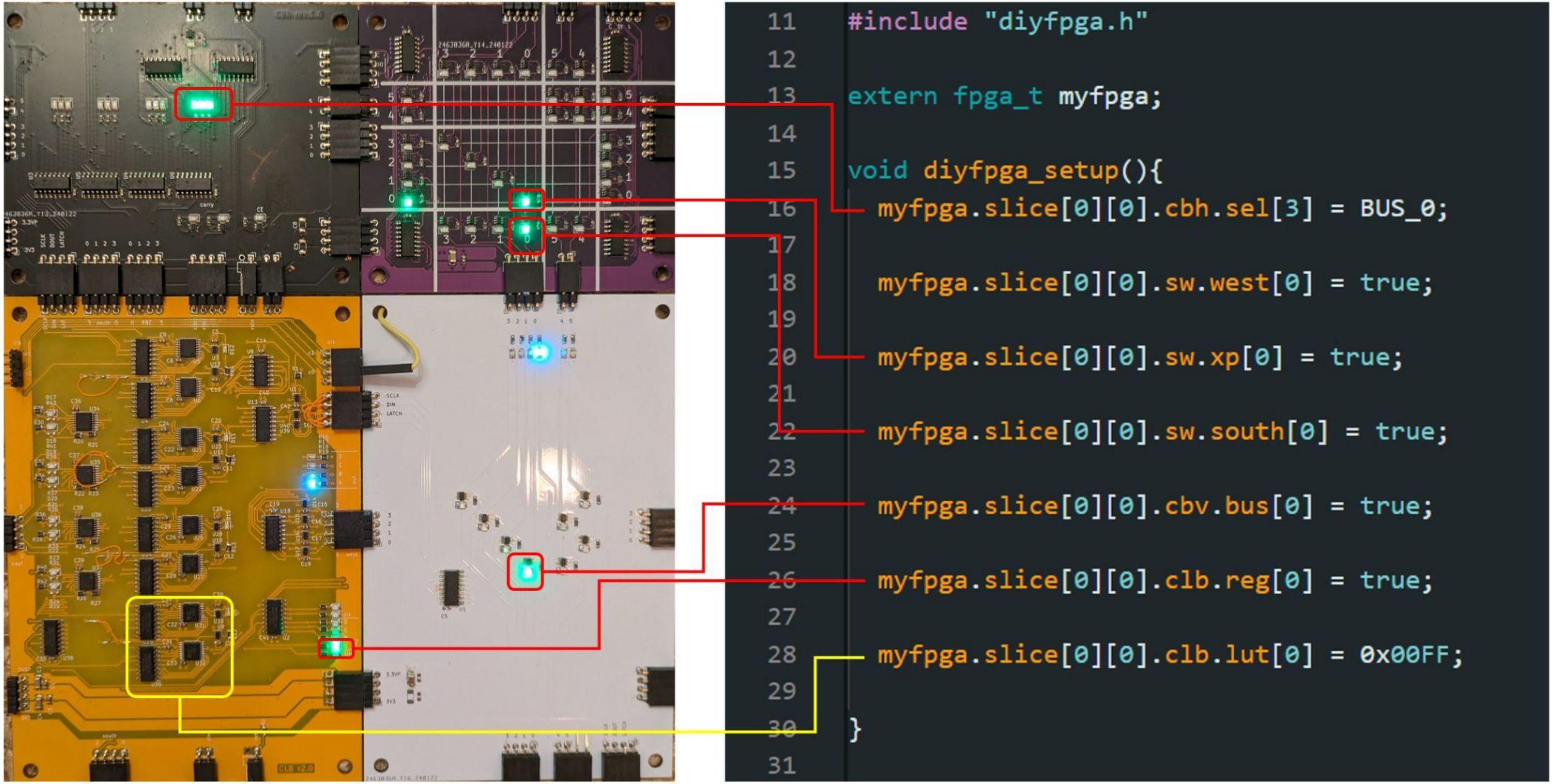


s3	s2	s1	s0	y
0	0	0	0	*1*
0	0	0	1	*1*
0	0	1	0	*1*
0	0	1	1	*1*
0	1	0	0	*1*
0	1	0	1	*1*
0	1	1	0	*1*
0	1	1	1	*1*
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

0x FF

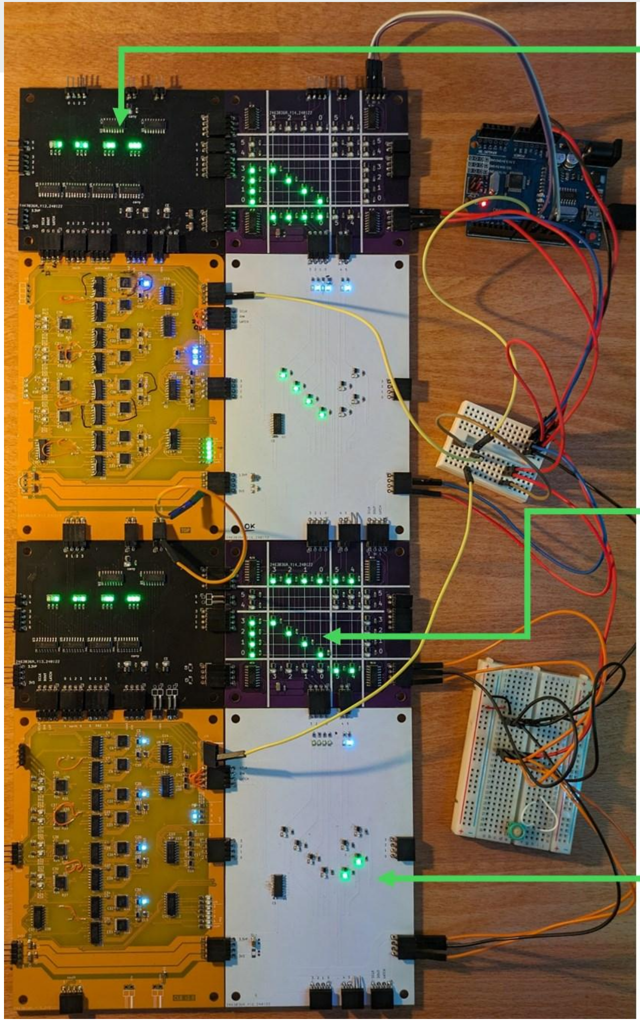
0x FF00

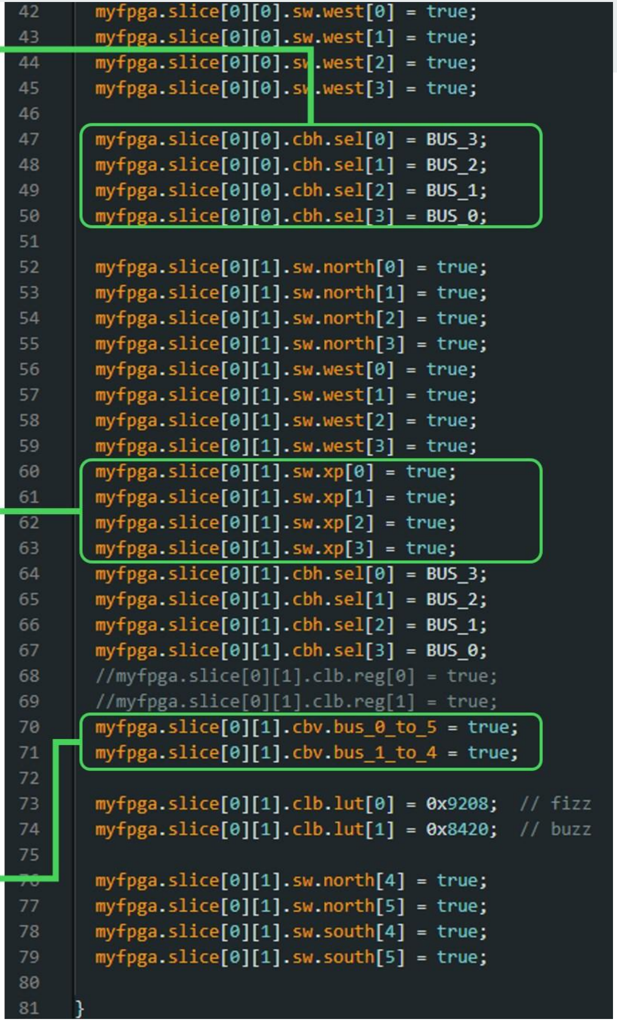
compiler design



```
#include "diyfpga.h"

extern fpga_t myfpga;

void diyfpga_setup(){
  myfpga.slice[0][0].cbh.sel[3] = BUS_0;

  myfpga.slice[0][0].sw.west[0] = true;

  myfpga.slice[0][0].sw.xp[0] = true;

  myfpga.slice[0][0].sw.south[0] = true;

  myfpga.slice[0][0].cbv.bus[0] = true;

  myfpga.slice[0][0].clb.reg[0] = true;

  myfpga.slice[0][0].clb.lut[0] = 0x00FF;

}
```

compiler design





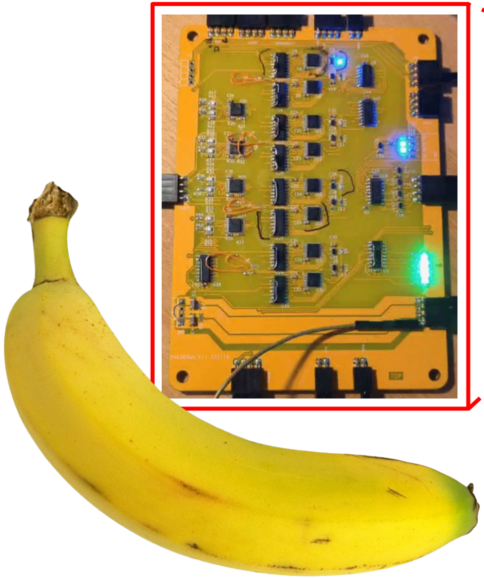
```
    myfpga.slice[0][0].sw.west[0] = true;
    myfpga.slice[0][0].sw.west[1] = true;
    myfpga.slice[0][0].sw.west[2] = true;
    myfpga.slice[0][0].sw.west[3] = true;

    myfpga.slice[0][0].cbh.sel[0] = BUS_3;
    myfpga.slice[0][0].cbh.sel[1] = BUS_2;
    myfpga.slice[0][0].cbh.sel[2] = BUS_1;
    myfpga.slice[0][0].cbh.sel[3] = BUS_0;

    myfpga.slice[0][1].sw.north[0] = true;
    myfpga.slice[0][1].sw.north[1] = true;
    myfpga.slice[0][1].sw.north[2] = true;
    myfpga.slice[0][1].sw.north[3] = true;
    myfpga.slice[0][1].sw.west[0] = true;
    myfpga.slice[0][1].sw.west[1] = true;
    myfpga.slice[0][1].sw.west[2] = true;
    myfpga.slice[0][1].sw.west[3] = true;
    myfpga.slice[0][1].sw.xp[0] = true;
    myfpga.slice[0][1].sw.xp[1] = true;
    myfpga.slice[0][1].sw.xp[2] = true;
    myfpga.slice[0][1].sw.xp[3] = true;
    myfpga.slice[0][1].cbh.sel[0] = BUS_3;
    myfpga.slice[0][1].cbh.sel[1] = BUS_2;
    myfpga.slice[0][1].cbh.sel[2] = BUS_1;
    myfpga.slice[0][1].cbh.sel[3] = BUS_0;
    //myfpga.slice[0][1].clb.reg[0] = true;
    //myfpga.slice[0][1].clb.reg[1] = true;
    myfpga.slice[0][1].cbv.bus_0_to_5 = true;
    myfpga.slice[0][1].cbv.bus_1_to_4 = true;

    myfpga.slice[0][1].clb.lut[0] = 0x9208;  // fizz
    myfpga.slice[0][1].clb.lut[1] = 0x8420;  // buzz

    myfpga.slice[0][1].sw.north[4] = true;
    myfpga.slice[0][1].sw.north[5] = true;
    myfpga.slice[0][1].sw.south[4] = true;
    myfpga.slice[0][1].sw.south[5] = true;

}
```

We have a 4-bit counter

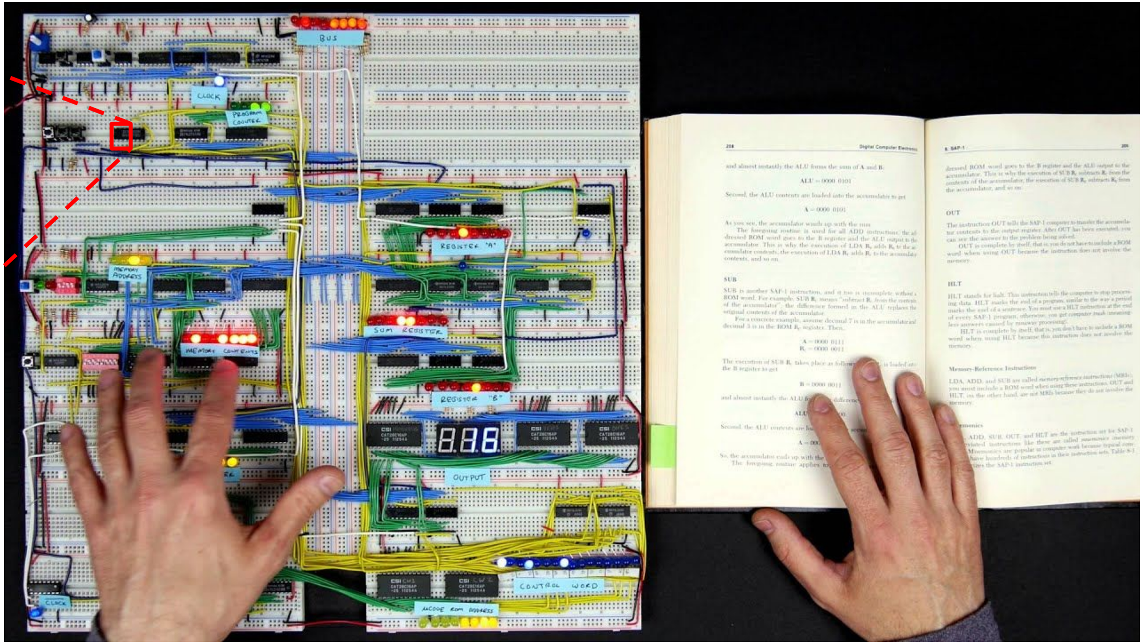
YAY!

à propos “8 bit CPU”…

What about the 8-bit CPU?



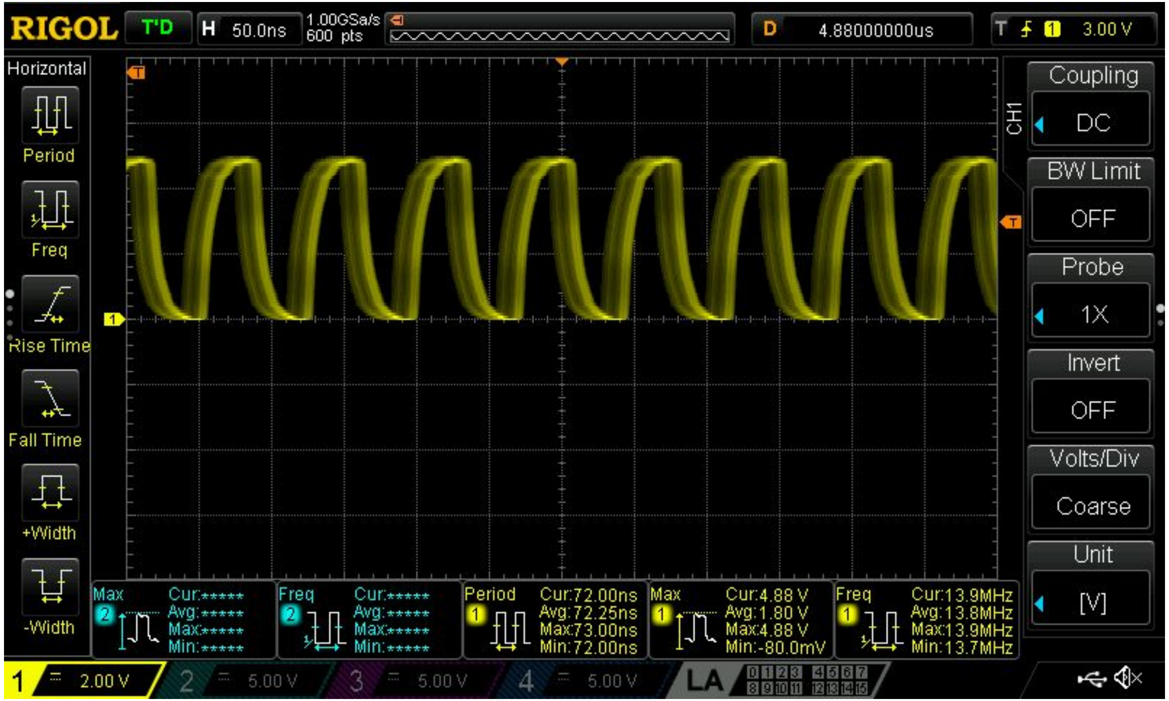
Banana for scale



f-Max

13.9 MHz Ring oscillator

(inverter loop without register)



```
H 50.0ns   1.00GSa/s 600 pts   D 4.88000000us   T 1 3.00 V
```

Coupling: DC

BW Limit: OFF

Probe: 1X

Invert: OFF

Volts/Div: Coarse

Unit: [V]

```
Max 2     Cur:*****     Freq 2    Cur:*****     Period 1  Cur:72.00ns   Max 1  Cur:4.88 V     Freq 1  Cur:13.9MHz
          Avg:*****               Avg:*****               Avg:72.25ns          Avg:1.80 V             Avg:13.8MHz
          Max:*****               Max:*****               Max:73.00ns          Max:4.88 V             Max:13.9MHz
          Min:*****               Min:*****               Min:72.00ns          Min:-80.0mV            Min:13.7MHz
```

```
1  2.00 V    2  5.00 V    3  5.00 V    4  5.00 V
```

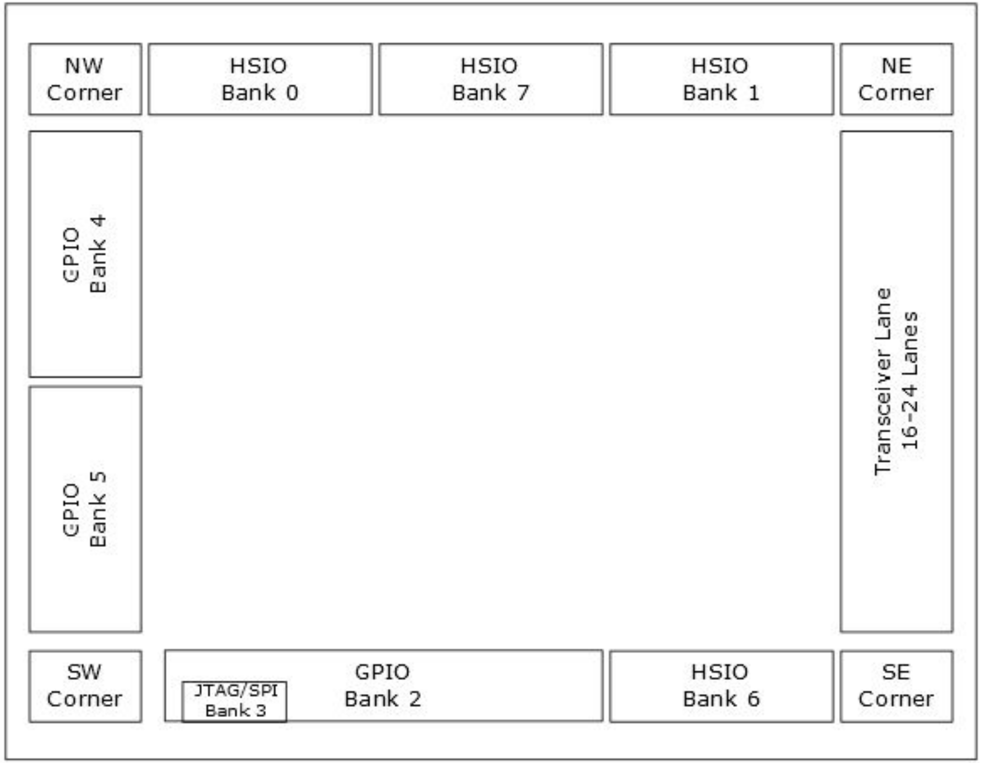
Next Steps

Oh cool you ~~go to the gym~~? :)
program FPGAs

What other hobbies do you have??

A few ideas...

IO Banks
+
PLL (clocking)

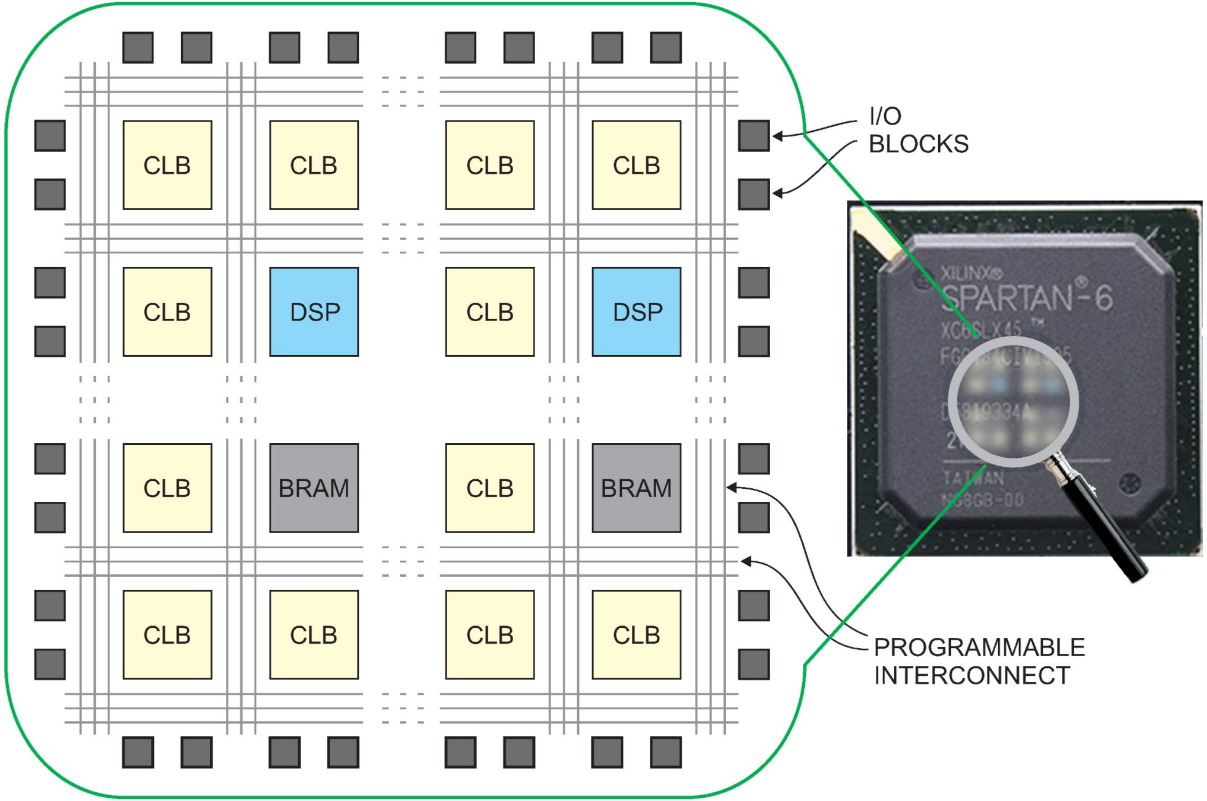


https://onlinedocs.microchip.com/pr/GUID-53092BEF-DCEB-4741-9EFC-5843AA55C657-en-US-1/index.html

DSP

+

Block RAM

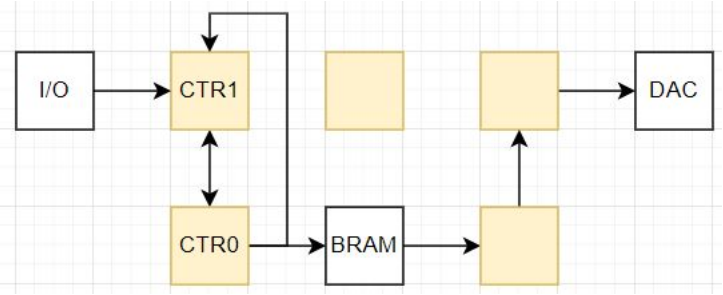


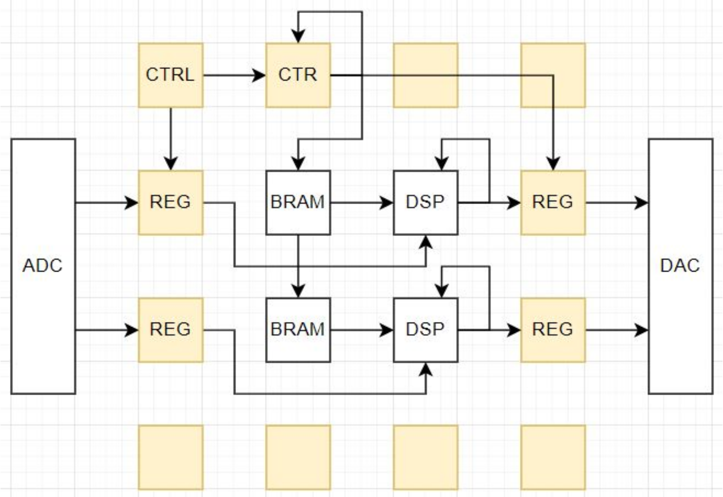
https://www.mdpi.com/2079-9292/12/17/3585

Audio DSP

- Can it act as a waveform generator
  - user input → frequency tuning
  - DDS with BRAM sine wave lookup

- Can it perform audio DSP?
  - 8 bit quantization
  - ~ 40 kHz sample rate
  - *time domain multiplex* on single DSP slice
  - 10 Tap FIR filter → 400 kHz FPGA clock

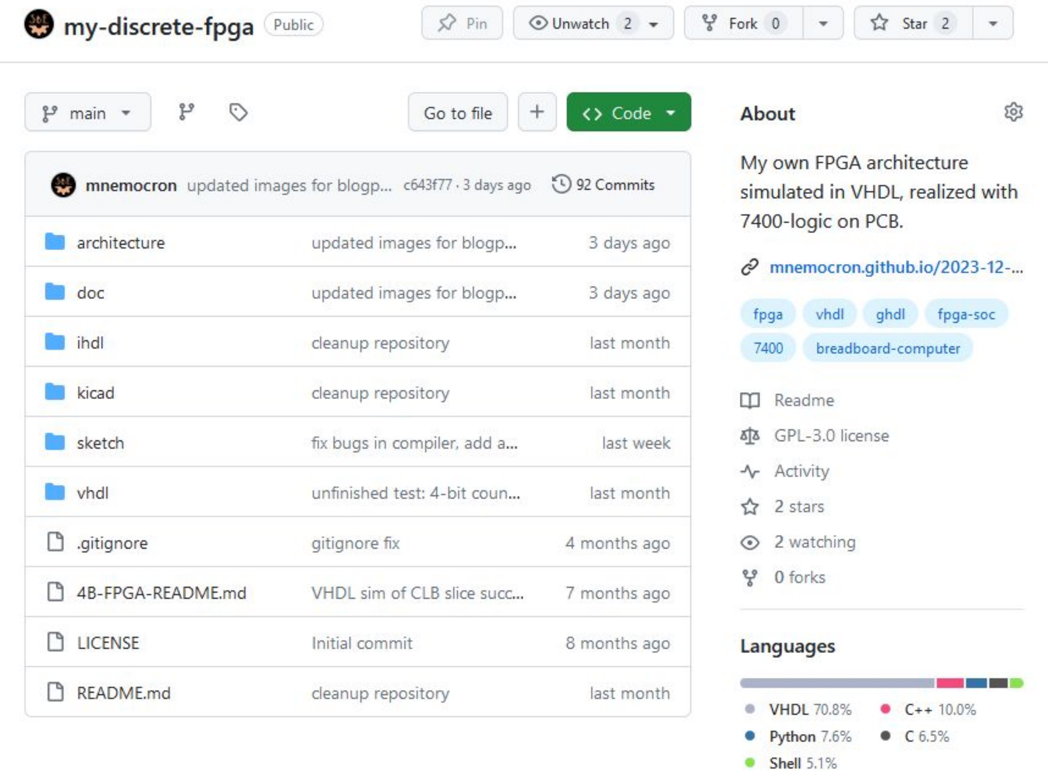




Further reading

github.com/mnemocron/my-discrete-fpga

mnemocron.github.io/tags/#diy-fpga



mnemocron updated images for blogp... c643f77 · 3 days ago 92 Commits

architecture	updated images for blogp...	3 days ago
doc	updated images for blogp...	3 days ago
ihdl	cleanup repository	last month
kicad	cleanup repository	last month
sketch	fix bugs in compiler, add a...	last week
vhdl	unfinished test: 4-bit coun...	last month
.gitignore	gitignore fix	4 months ago
4B-FPGA-README.md	VHDL sim of CLB slice succ...	7 months ago
LICENSE	Initial commit	8 months ago
README.md	cleanup repository	last month

About

My own FPGA architecture simulated in VHDL, realized with 7400-logic on PCB.

mnemocron.github.io/2023-12-...

Languages