The Raven chip:

First-time silicon success with qflow and efabless

Tim Edwards
Mohamed Kassem
efabless

The Raven chip: First-time silicon success with qflow and efabless

The Challenge:

Design and verify a working microprocessor SoC in < 3 months.

Validate first-time silicon success.

Requirements:

Chip to be simulated and layout drawn with all open-source tools.

All software, firmware, and hardware to be open source.[*]

Chip must demonstrate function of a set of digital and analog IP.

Test board PCB design, BOM, and USB driver to be open source

Final design to be placed in efabless catalog and in github

Design becomes a reference design to be customized by other platform users.

[*](caveats apply)

The starting point:

The PicoRV32 RISC-V core by Clifford Wolf

- Fully open source under generous license
- Available for download from github
- Packaged with a reference SoC implementation with UART and SPI flash driver
- Packaged with instructions for obtaining and installing the RISC-V gcc cross-compiler (for RV32IMC)
- Packaged with example C code and testbenches

“Because Instruction Sets Want To Be Free”

RISC-V®

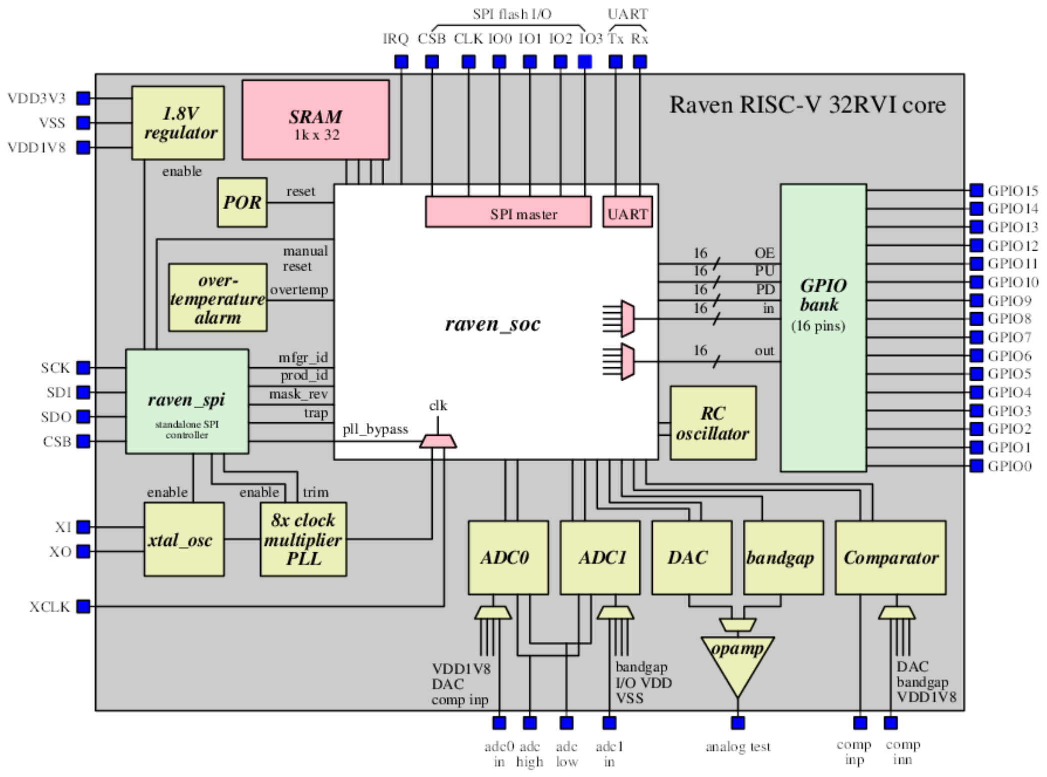
Modifications to verilog source needed:

SRAM pulled out of core module

Parameters adjusted for 1k words (4kB) SRAM

Memory mapping for analog IP

GPIO controls extended for general-purpose digital I/O cell



The target process:

X-Fab XH018

Base MOS LP (low power) option

6 metal stack (5 standard route layers, 1 thicker top metal)

XFAB
MIXED-SIGNAL FOUNDRY EXPERTS

The proprietary parts:

X-Fab digital standard cells

X-Fab padframe I/O (3.3V with both 3.3V and 1.8V core)

X-Fab analog IP

X-Fab SRAM (from memory compiler)

Can a design have proprietary IP and still be called "open hardware"?

To be continued. . .

The platform:

efabless.com https://efabless.com

The efabless platform provides a way to present the open source tools configured for real foundry processes.

efabless has installed custom PDKs for X-Fab XH035 and XH018.

efabless requires registration, approved manually, to satisfy ECCN rules and help protect foundry proprietary data.

The platform consists of a web interface including a marketplace catalog, and a design platform which is a virtual CentOS machine.

The platform is free to use as a sandbox for experimenting with designs.

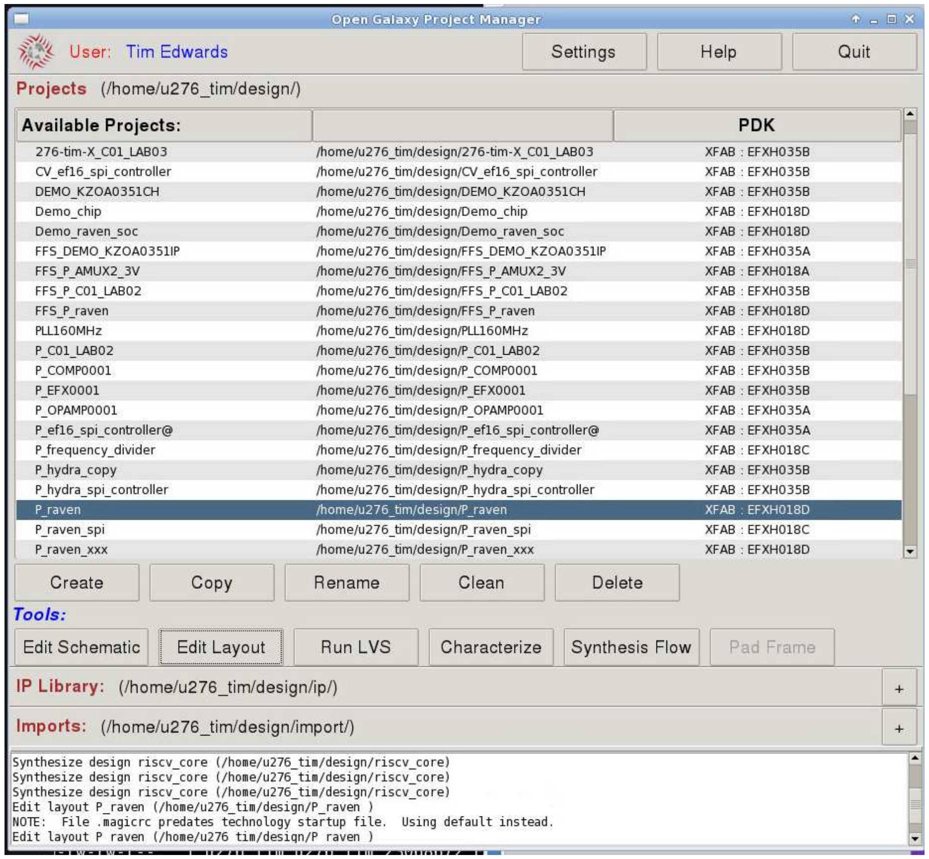
The Open Galaxy open source software environment:

Electric
https://www.staticfreesoft.com

ngspice
http://ngspice.sourceforge.net

Icarus verilog
http://iverilog.icarus.com

GTKwave
http://gtkwave.sourceforge.net



Projects (/home/u276_tim/design/)

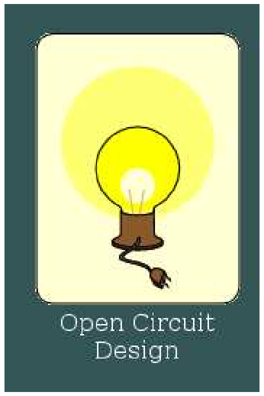
Available Projects:		PDK
276-tim-X_C01_LAB03	/home/u276_tim/design/276-tim-X_C01_LAB03	XFAB : EFXH035B
CV_ef16_spi_controller	/home/u276_tim/design/CV_ef16_spi_controller	XFAB : EFXH035B
DEMO_KZOA0351CH	/home/u276_tim/design/DEMO_KZOA0351CH	XFAB : EFXH035B
Demo_chip	/home/u276_tim/design/Demo_chip	XFAB : EFXH018D
Demo_raven_soc	/home/u276_tim/design/Demo_raven_soc	XFAB : EFXH018D
FFS_DEMO_KZOA0351IP	/home/u276_tim/design/FFS_DEMO_KZOA0351IP	XFAB : EFXH035A
FFS_P_AMUX2_3V	/home/u276_tim/design/FFS_P_AMUX2_3V	XFAB : EFXH018A
FFS_P_C01_LAB02	/home/u276_tim/design/FFS_P_C01_LAB02	XFAB : EFXH035B
FFS_P_raven	/home/u276_tim/design/FFS_P_raven	XFAB : EFXH018D
PLL160MHz	/home/u276_tim/design/PLL160MHz	XFAB : EFXH018D
P_C01_LAB02	/home/u276_tim/design/P_C01_LAB02	XFAB : EFXH035B
P_COMP0001	/home/u276_tim/design/P_COMP0001	XFAB : EFXH035B
P_EFX0001	/home/u276_tim/design/P_EFX0001	XFAB : EFXH035B
P_OPAMP0001	/home/u276_tim/design/P_OPAMP0001	XFAB : EFXH035A
P_ef16_spi_controller@	/home/u276_tim/design/P_ef16_spi_controller@	XFAB : EFXH035A
P_frequency_divider	/home/u276_tim/design/P_frequency_divider	XFAB : EFXH018C
P_hydra_copy	/home/u276_tim/design/P_hydra_copy	XFAB : EFXH035B
P_hydra_spi_controller	/home/u276_tim/design/P_hydra_spi_controller	XFAB : EFXH035B
P_raven	/home/u276_tim/design/P_raven	XFAB : EFXH018D
P_raven_spi	/home/u276_tim/design/P_raven_spi	XFAB : EFXH018C
P_raven_xxx	/home/u276_tim/design/P_raven_xxx	XFAB : EFXH018D

IP Library: (/home/u276_tim/design/ip/)

Imports: (/home/u276_tim/design/import/)

```
Synthesize design riscv_core (/home/u276_tim/design/riscv_core)
Synthesize design riscv_core (/home/u276_tim/design/riscv_core)
Synthesize design riscv_core (/home/u276_tim/design/riscv_core)
Edit layout P_raven (/home/u276_tim/design/P_raven )
NOTE:  File .magicrc predates technology startup file.  Using default instead.
Edit layout P raven (/home/u276 tim/design/P raven )
```

Custom scripts and tools on the platform



Open Circuit Design tools
http://opencircuitdesign.com

Open Circuit Design tools:

Magic VLSI Layout Tool

Current development version 8.2

http://opencircuitdesign.com/magic

Read/write LEF, DEF, GDS

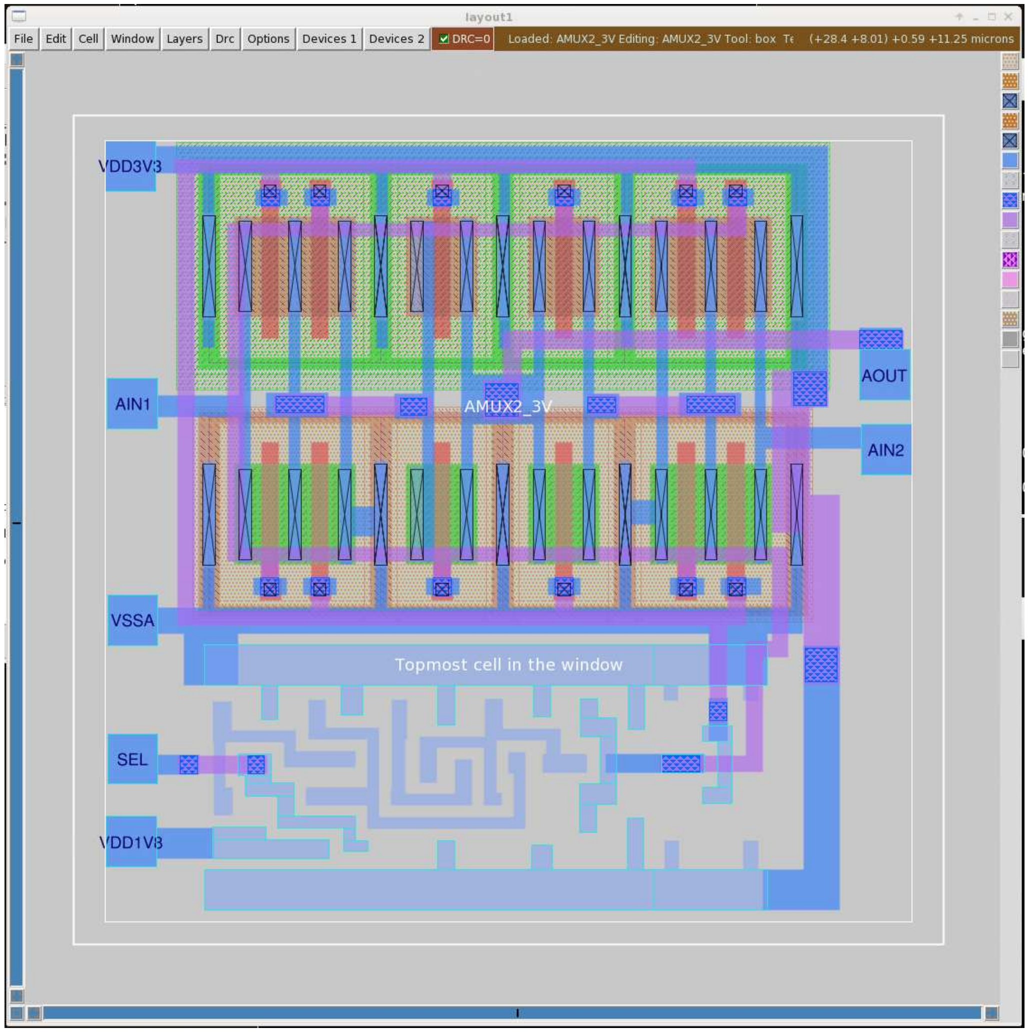
Interactive DRC

Interactive and batch Tcl scripting

Device and parasitic extraction, SPICE netlisting

Interactive wiring

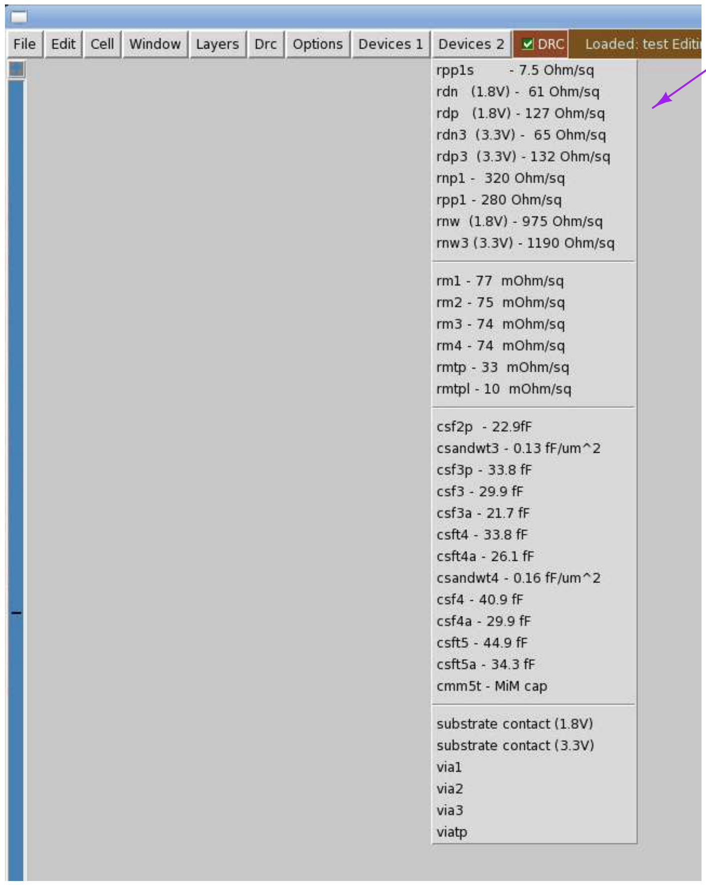
Tcl-based PDK



Loaded: AMUX2_3V Editing: AMUX2_3V Tool: box Te (+28.4 +8.01) +0.59 +11.25 microns

The Magic PDK:

Device list for XH018



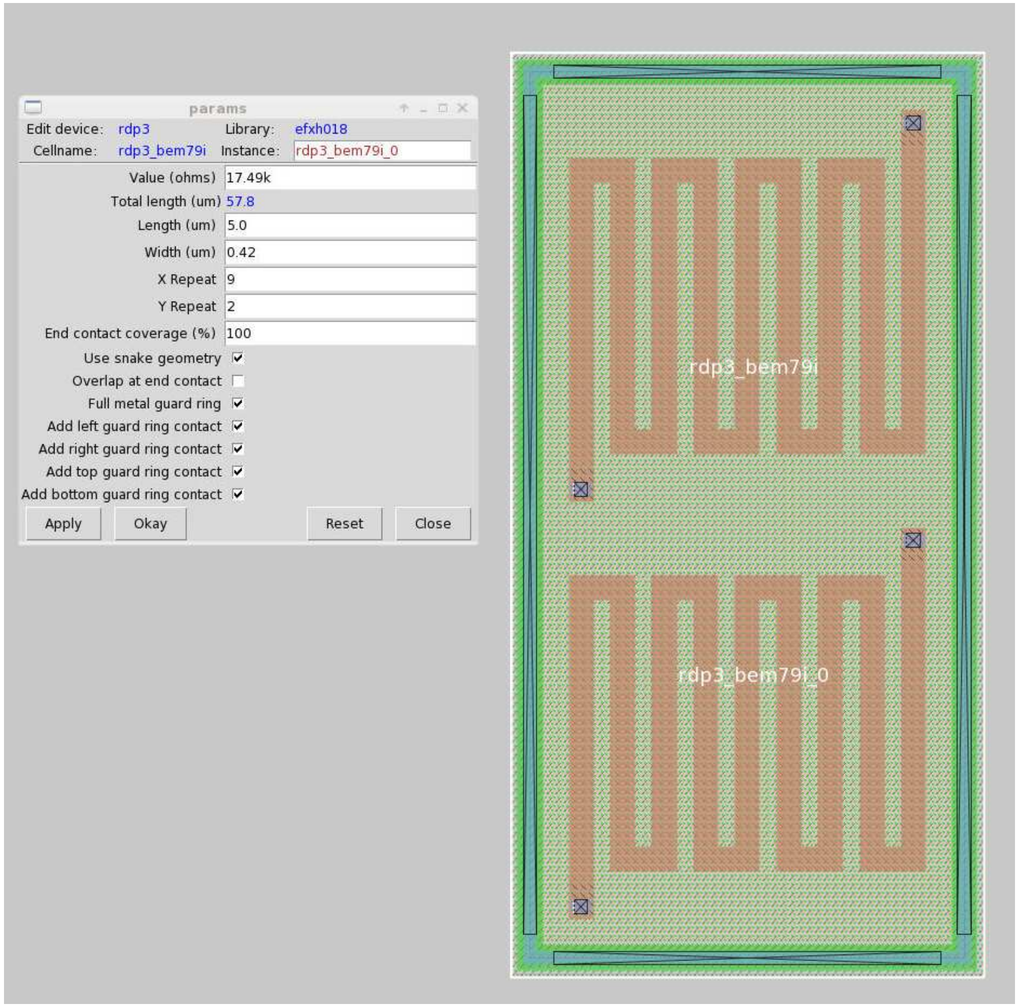
rppls - 7.5 Ohm/sq
rdn (1.8V) - 61 Ohm/sq
rdp (1.8V) - 127 Ohm/sq
rdn3 (3.3V) - 65 Ohm/sq
rdp3 (3.3V) - 132 Ohm/sq
rnp1 - 320 Ohm/sq
rpp1 - 280 Ohm/sq
rnw (1.8V) - 975 Ohm/sq
rnw3 (3.3V) - 1190 Ohm/sq

rm1 - 77 mOhm/sq
rm2 - 75 mOhm/sq
rm3 - 74 mOhm/sq
rm4 - 74 mOhm/sq
rmtp - 33 mOhm/sq
rmtpl - 10 mOhm/sq

csf2p - 22.9fF
csandwt3 - 0.13 fF/um^2
csf3p - 33.8 fF
csf3 - 29.9 fF
csf3a - 21.7 fF
csft4 - 33.8 fF
csft4a - 26.1 fF
csandwt4 - 0.16 fF/um^2
csf4 - 40.9 fF
csf4a - 29.9 fF
csft5 - 44.9 fF
csft5a - 34.3 fF
cmm5t - MiM cap

substrate contact (1.8V)
substrate contact (3.3V)
via1
via2
via3
viatp

Tcl-scripted device generators



Edit device:	rdp3	Library:	efxh018
Cellname:	rdp3_bem79i	Instance:	rdp3_bem79i_0

Value (ohms)	17.49k
Total length (um)	57.8
Length (um)	5.0
Width (um)	0.42
X Repeat	9
Y Repeat	2
End contact coverage (%)	100
Use snake geometry	✓
Overlap at end contact	
Full metal guard ring	✓
Add left guard ring contact	✓
Add right guard ring contact	✓
Add top guard ring contact	✓
Add bottom guard ring contact	✓

Open Circuit Design tools:

Netgen (version 1.5)

http://opencircuitdesign.com/netgen

LVS tool

Compares SPICE and verilog netlists

Device (e.g., resistor) network matching

Hierarchical matching

Property matching

Property expression evaluation

Pin name matching

Open Circuit Design tools:

http://opencircuitdesign.com/qflow

Qflow (stable version 1.3) digital synthesis:

Flow comprised of “mix and match” tools

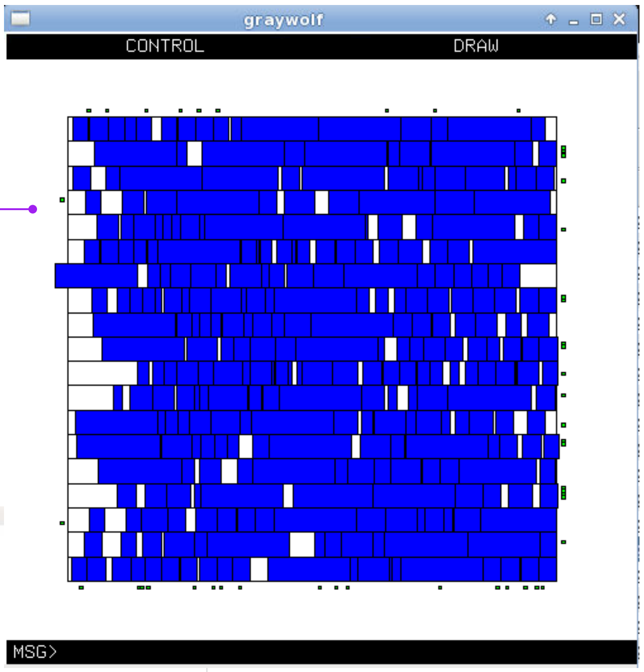
Current tools handle designs of up to ~30k gates

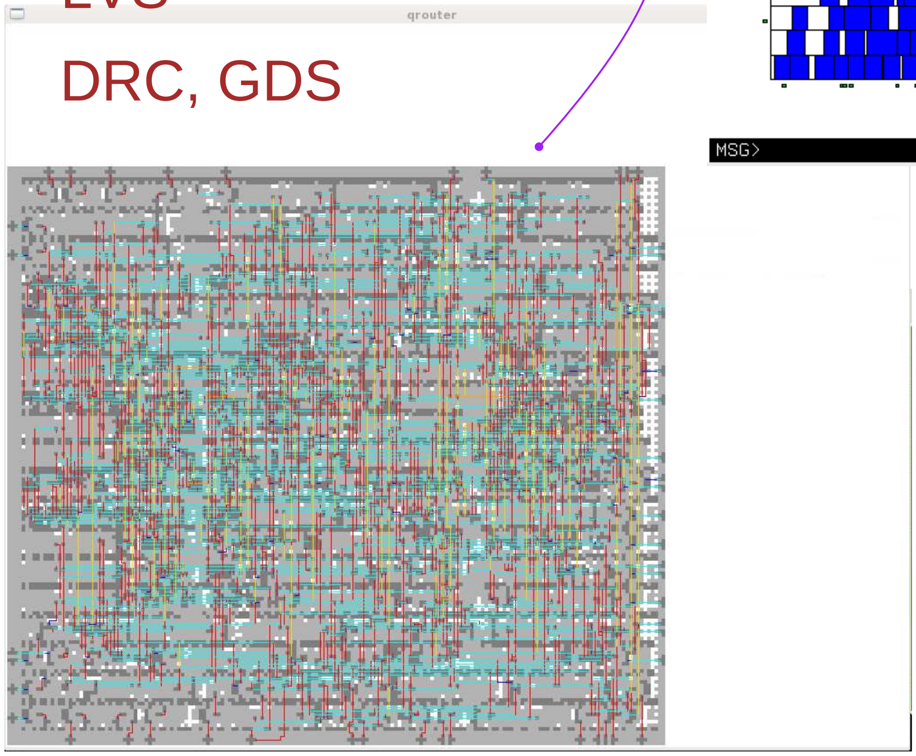
Current tools understand layout practices at ~65nm and above

Easily synthesizes, analyzes, and routes the Raven SoC core

Qflow components:

Yosys	synthesis
Graywolf	placement
Qrouter	routing
Vesta	static timing analysis
Netgen	LVS
Magic	DRC, GDS





Qflow scripts:

Netlist manipulation (BLIF, verilog, SPICE, SPEF)

Resize gates

Add clock tree (coupled with placement, unlike other tools)

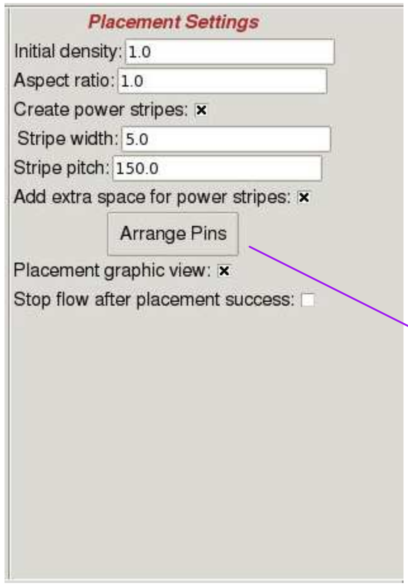
Reduce fanout

Reduce density

Add power buses

Arrange pins

GUI



Placement Settings

Initial density: 1.0

Aspect ratio: 1.0

Create power stripes: [✓]

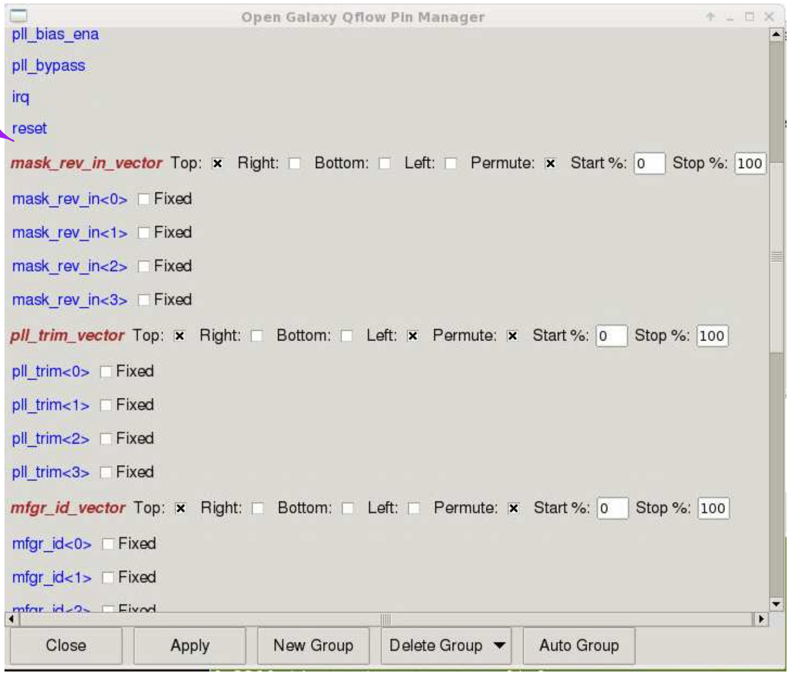
Stripe width: 5.0

Stripe pitch: 150.0

Add extra space for power stripes: [✓]

Placement graphic view: [✓]

Stop flow after placement success: [ ]



pll_bias_ena

pll_bypass

irq

reset

mask_rev_in_vector Top: [✓] Right: [ ] Bottom: [ ] Left: [ ] Permute: [✓] Start %: 0 Stop %: 100

mask_rev_in<0> [ ] Fixed

mask_rev_in<1> [ ] Fixed

mask_rev_in<2> [ ] Fixed

mask_rev_in<3> [ ] Fixed

pll_trim_vector Top: [✓] Right: [ ] Bottom: [ ] Left: [✓] Permute: [✓] Start %: 0 Stop %: 100

pll_trim<0> [ ] Fixed

pll_trim<1> [ ] Fixed

pll_trim<2> [ ] Fixed

pll_trim<3> [ ] Fixed

mfgr_id_vector Top: [✓] Right: [ ] Bottom: [ ] Left: [ ] Permute: [✓] Start %: 0 Stop %: 100

mfgr_id<0> [ ] Fixed

mfgr_id<1> [ ] Fixed

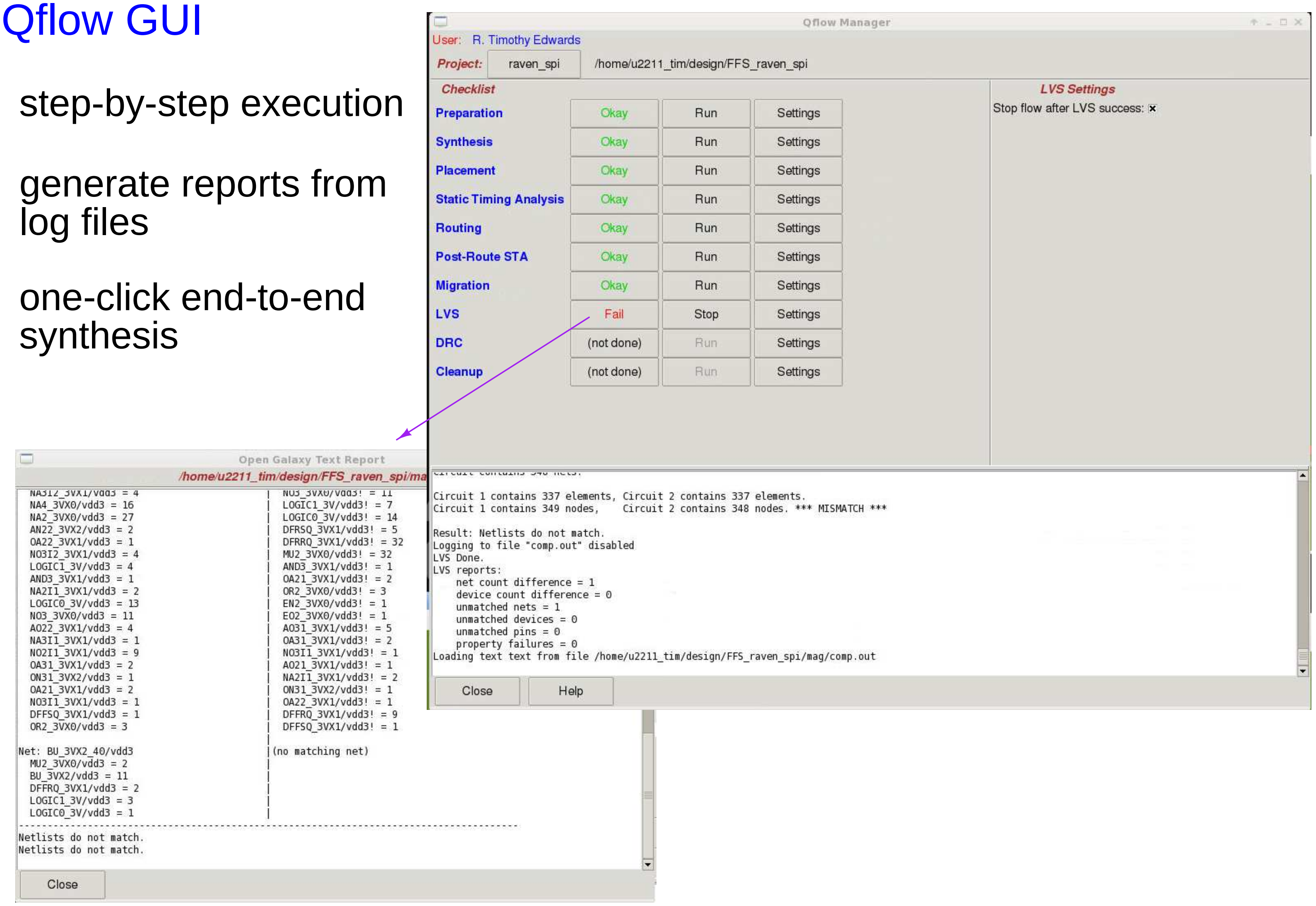
mfgr_id<2> [ ] Fixed

Qflow GUI

step-by-step execution

generate reports from log files

one-click end-to-end synthesis



User: R. Timothy Edwards

Project: raven_spi /home/u2211_tim/design/FFS_raven_spi

Checklist			
Preparation	Okay	Run	Settings
Synthesis	Okay	Run	Settings
Placement	Okay	Run	Settings
Static Timing Analysis	Okay	Run	Settings
Routing	Okay	Run	Settings
Post-Route STA	Okay	Run	Settings
Migration	Okay	Run	Settings
LVS	Fail	Stop	Settings
DRC	(not done)	Run	Settings
Cleanup	(not done)	Run	Settings

LVS Settings

Stop flow after LVS success: ☒

```
Circuit contains 348 nets.

Circuit 1 contains 337 elements, Circuit 2 contains 337 elements.
Circuit 1 contains 349 nodes,    Circuit 2 contains 348 nodes. *** MISMATCH ***

Result: Netlists do not match.
Logging to file "comp.out" disabled
LVS Done.
LVS reports:
    net count difference = 1
    device count difference = 0
    unmatched nets = 1
    unmatched devices = 0
    unmatched pins = 0
    property failures = 0
Loading text text from file /home/u2211_tim/design/FFS_raven_spi/mag/comp.out
```

/home/u2211_tim/design/FFS_raven_spi/ma

```
  NA3I2_3VX1/vdd3 = 4                       |  NO3_3VX0/vdd3! = 11
  NA4_3VX0/vdd3 = 16                        |  LOGIC1_3V/vdd3! = 7
  NA2_3VX0/vdd3 = 27                        |  LOGIC0_3V/vdd3! = 14
  AN22_3VX2/vdd3 = 2                        |  DFRSQ_3VX1/vdd3! = 5
  OA22_3VX1/vdd3 = 1                        |  DFRRQ_3VX1/vdd3! = 32
  NO3I2_3VX1/vdd3 = 4                       |  MU2_3VX0/vdd3! = 32
  LOGIC1_3V/vdd3 = 4                        |  AND3_3VX1/vdd3! = 1
  AND3_3VX1/vdd3 = 1                        |  OA21_3VX1/vdd3! = 2
  NA2I1_3VX1/vdd3 = 2                       |  OR2_3VX0/vdd3! = 3
  LOGIC0_3V/vdd3 = 13                       |  EN2_3VX0/vdd3! = 1
  NO3_3VX0/vdd3 = 11                        |  EO2_3VX0/vdd3! = 1
  AO22_3VX1/vdd3 = 4                        |  AO31_3VX1/vdd3! = 5
  NA3I1_3VX1/vdd3 = 1                       |  OA31_3VX1/vdd3! = 2
  NO2I1_3VX1/vdd3 = 9                       |  NO3I1_3VX1/vdd3! = 1
  OA31_3VX1/vdd3 = 2                        |  AO21_3VX1/vdd3! = 1
  ON31_3VX2/vdd3 = 1                        |  NA2I1_3VX1/vdd3! = 2
  OA21_3VX1/vdd3 = 2                        |  ON31_3VX2/vdd3! = 1
  NO3I1_3VX1/vdd3 = 1                       |  OA22_3VX1/vdd3! = 1
  DFFSQ_3VX1/vdd3 = 1                       |  DFFRQ_3VX1/vdd3! = 9
  OR2_3VX0/vdd3 = 3                         |  DFFSQ_3VX1/vdd3! = 1
                                            |
Net: BU_3VX2_40/vdd3                        |(no matching net)
  MU2_3VX0/vdd3 = 2                         |
  BU_3VX2/vdd3 = 11                         |
  DFFRQ_3VX1/vdd3 = 2                       |
  LOGIC1_3V/vdd3 = 3                        |
  LOGIC0_3V/vdd3 = 1                        |
-------------------------------------------------------------------------------------
Netlists do not match.
Netlists do not match.
```

What is Open Layout?

Mask-geometry layout is foundry proprietary.

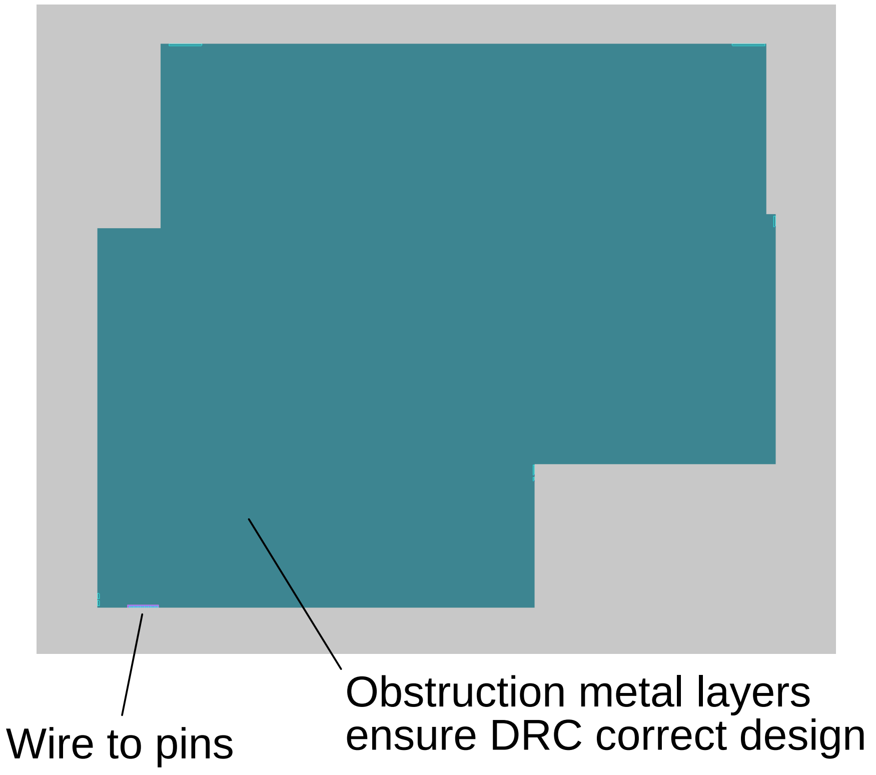
How can you design an entire chip and submit to the foundry for fabrication without signing an NDA, purchasing commercial tools, and installing PDKs?

All cells at the transistor level are abstracted views using information from the corresponding LEF files.

3.3V ADC, Original vendor layout:

(Layout blurred to protect the identity of the victim)

3.3V ADC, Abstracted layout (from LEF view):

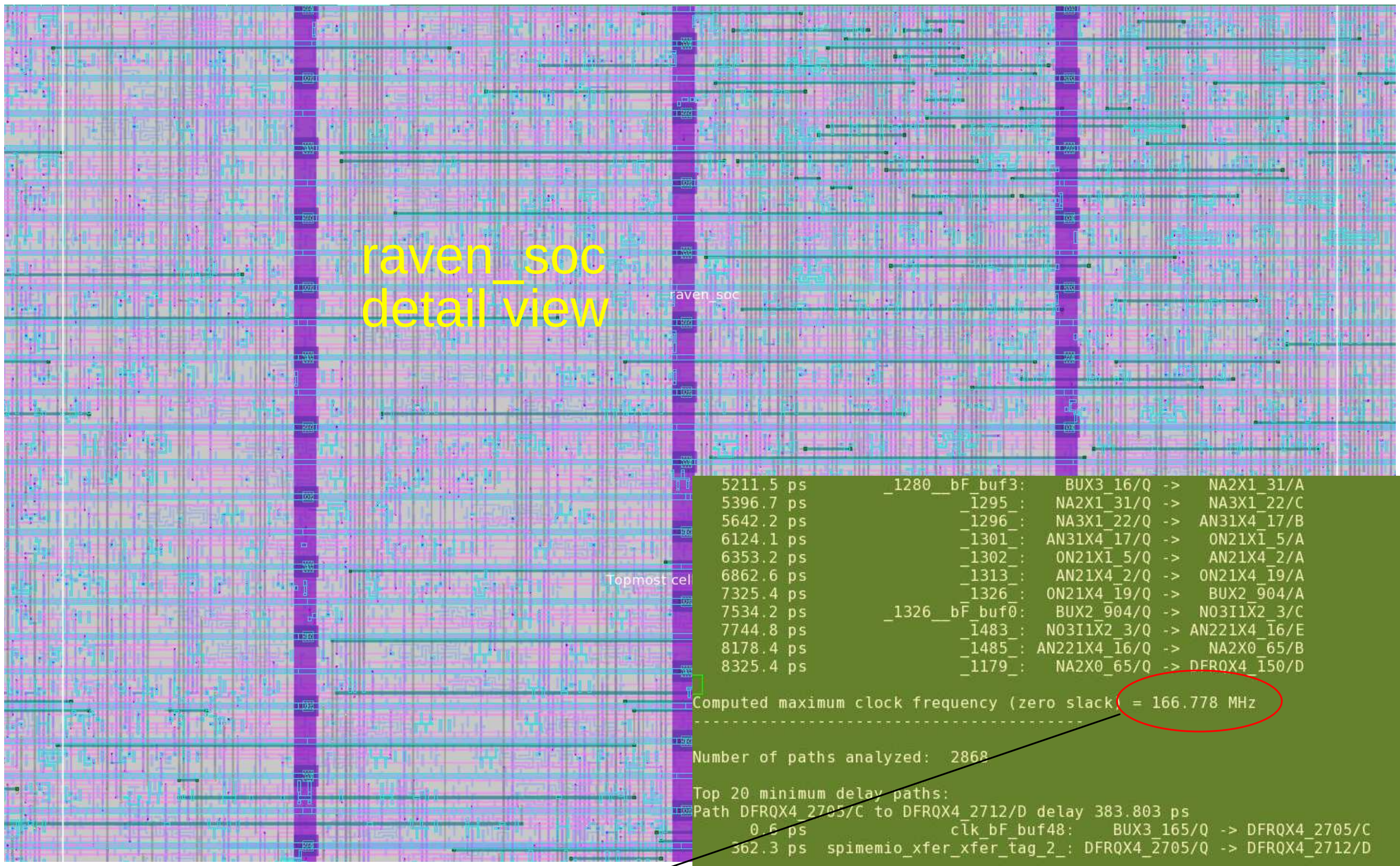


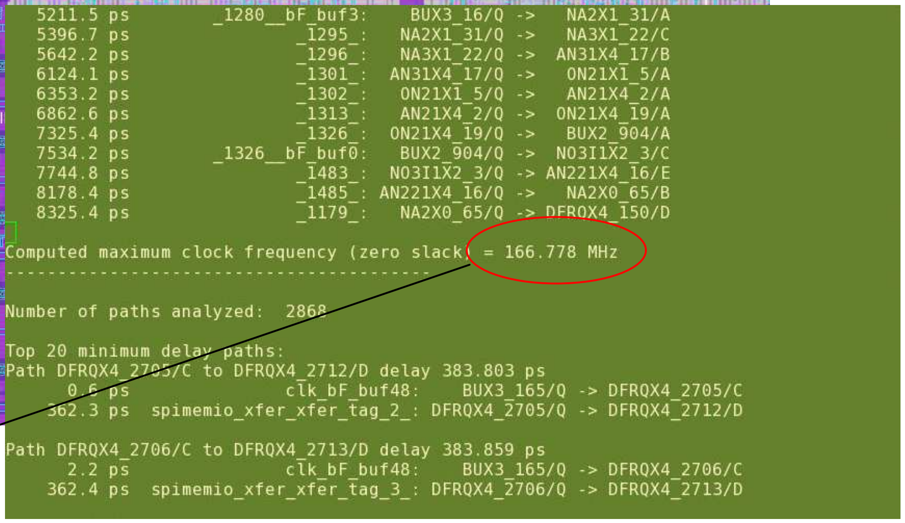
Building the Raven chip

Part 1: SoC Synthesis

Large design, needs 5 route layers
+ 6th thick metal for power

Reduce density to ensure route success.





```
   5211.5 ps        _1280__bF_buf3:    BUX3_16/Q ->   NA2X1_31/A
   5396.7 ps                _1295_:   NA2X1_31/Q ->   NA3X1_22/C
   5642.2 ps                _1296_:   NA3X1_22/Q ->  AN31X4_17/B
   6124.1 ps                _1301_:  AN31X4_17/Q ->   ON21X1_5/A
   6353.2 ps                _1302_:   ON21X1_5/Q ->   AN21X4_2/A
   6862.6 ps                _1313_:   AN21X4_2/Q ->  ON21X4_19/A
   7325.4 ps                _1326_:  ON21X4_19/Q ->   BUX2_904/A
   7534.2 ps        _1326__bF_buf0:   BUX2_904/Q ->  NO3I1X2_3/C
   7744.8 ps                _1483_:  NO3I1X2_3/Q -> AN221X4_16/E
   8178.4 ps                _1485_: AN221X4_16/Q ->   NA2X0_65/B
   8325.4 ps                _1179_:   NA2X0_65/Q -> DFRQX4_150/D

Computed maximum clock frequency (zero slack) = 166.778 MHz
-----------------------------------------

Number of paths analyzed:  2868

Top 20 minimum delay paths:
Path DFRQX4_2705/C to DFRQX4_2712/D delay 383.803 ps
      0.6 ps                clk_bF_buf48:    BUX3_165/Q -> DFRQX4_2705/C
    362.3 ps  spimemio_xfer_xfer_tag_2_: DFRQX4_2705/Q -> DFRQX4_2712/D

Path DFRQX4_2706/C to DFRQX4_2713/D delay 383.859 ps
      2.2 ps                clk_bF_buf48:    BUX3_165/Q -> DFRQX4_2706/C
    362.4 ps  spimemio_xfer_xfer_tag_3_: DFRQX4_2706/Q -> DFRQX4_2713/D
```

Post-layout static timing analysis
shows clock rate > 150 MHz

Building the Raven chip

Part 2: Housekeeping SPI synthesis

3.3V digital standard cells

Accesses critical functions:

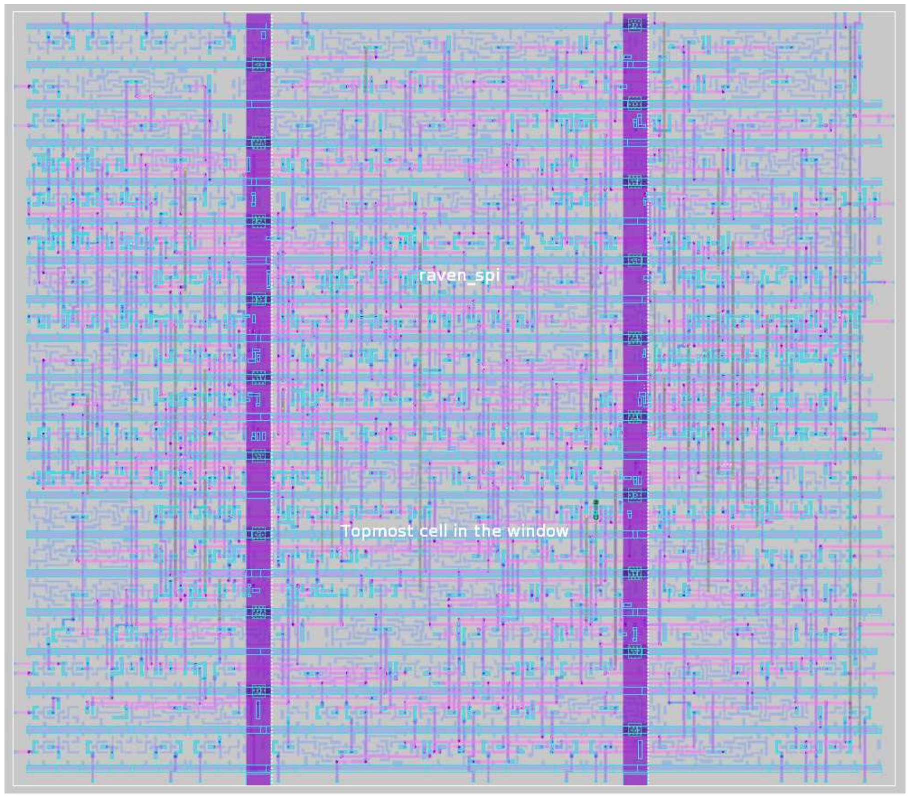
- Core 1.8V power supply
- Crystal oscillator enable
- Clock multiplier enable
- CPU trap signal

Generates signals:

- CPU reset
- IRQ event

Vendor/Product IDs

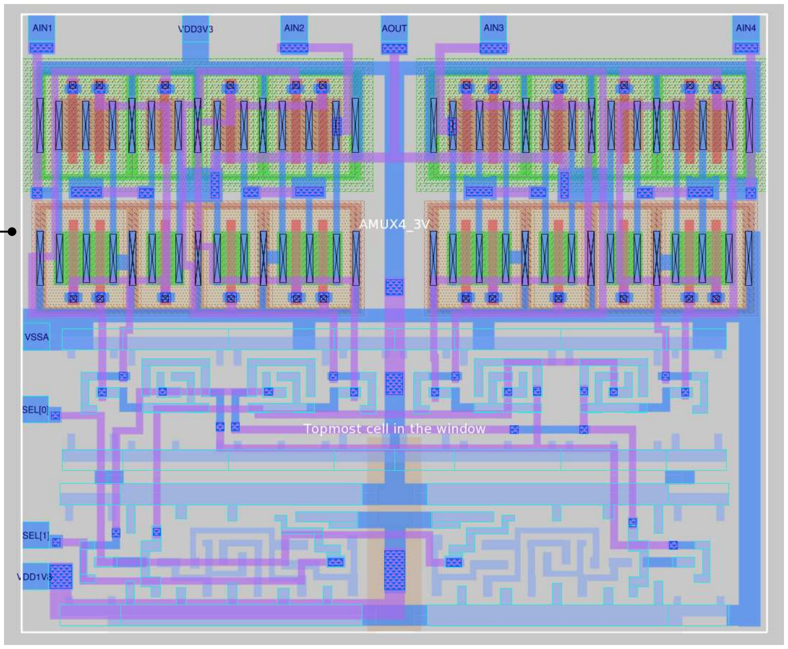
- Metal mask programmed



Building the Raven chip

Part 3: Custom Analog Cells

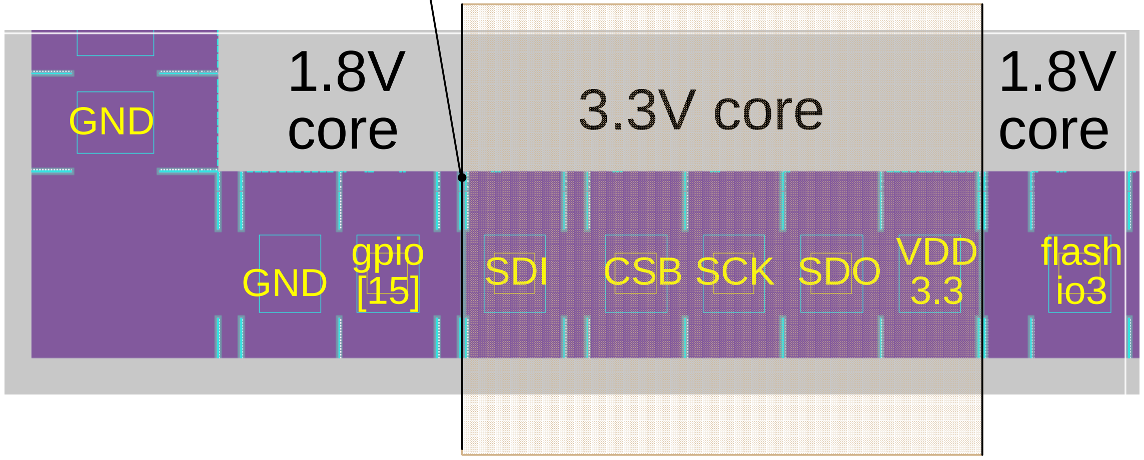
Analog Multiplexers



Level Shifters

Go to/from 1.8V and
3.3V circuits

Padframe voltage domain splitter



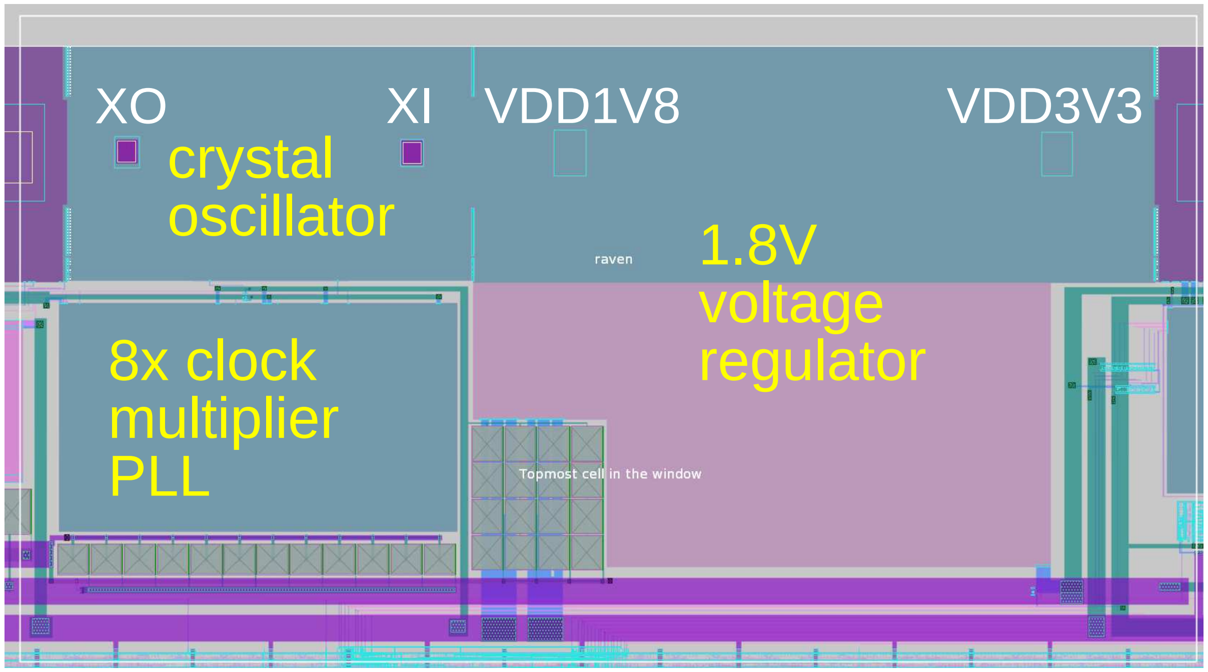
Building the Raven chip

Part 4: Process-compatible X-Fab cells

Six metal layer stack variants:

1.8V Regulator

Crystal Oscillator



Building the Raven chip

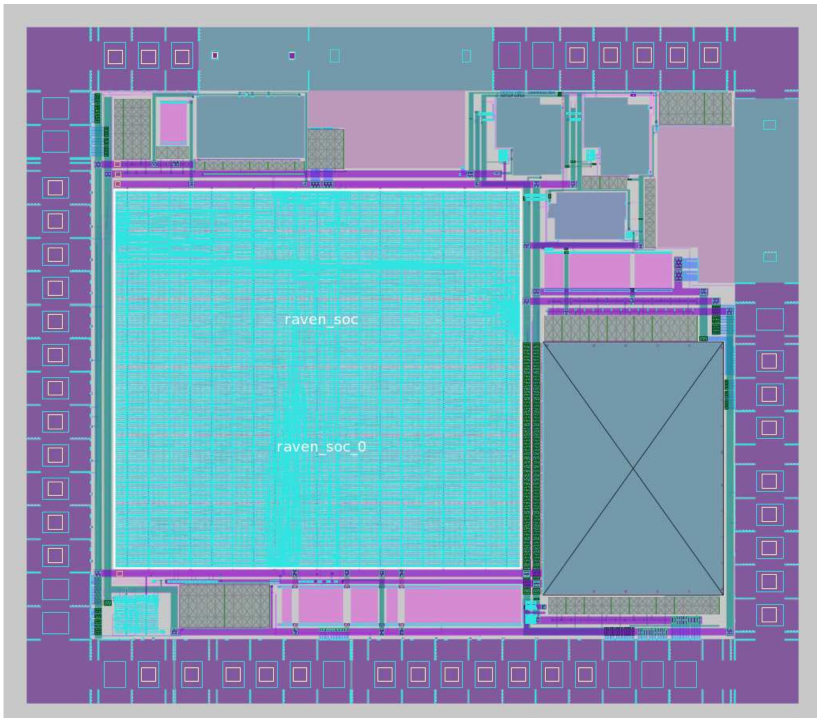
Part 5: Chip top-level assembly with Magic

All top-level wiring done with Magic interactive wiring tool

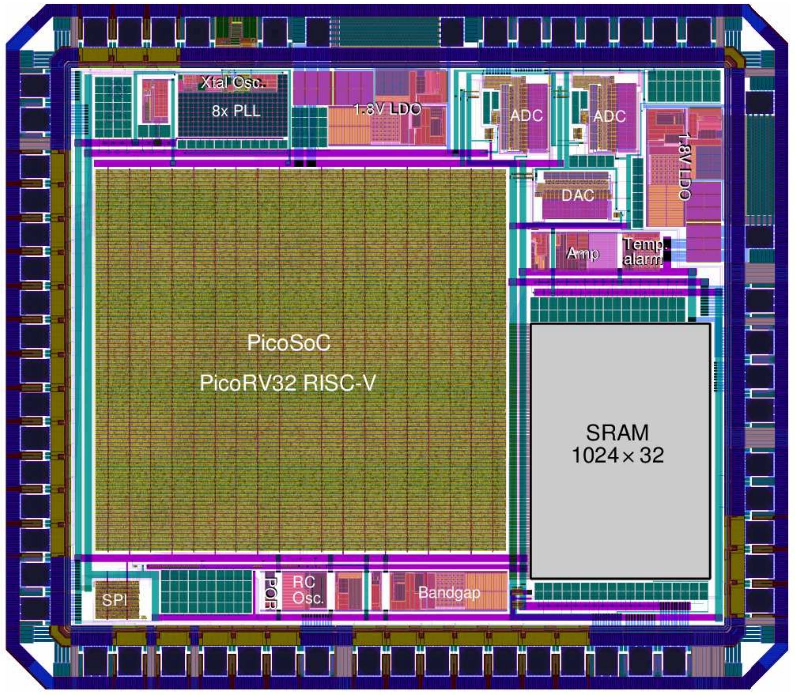
Manual top-level floorplanning

SRAM has no other view than the abstracted one.

Abstracted view:



GDS view (except for SRAM):



Building the Raven chip

Part 6: Top-level verification

Simulation:

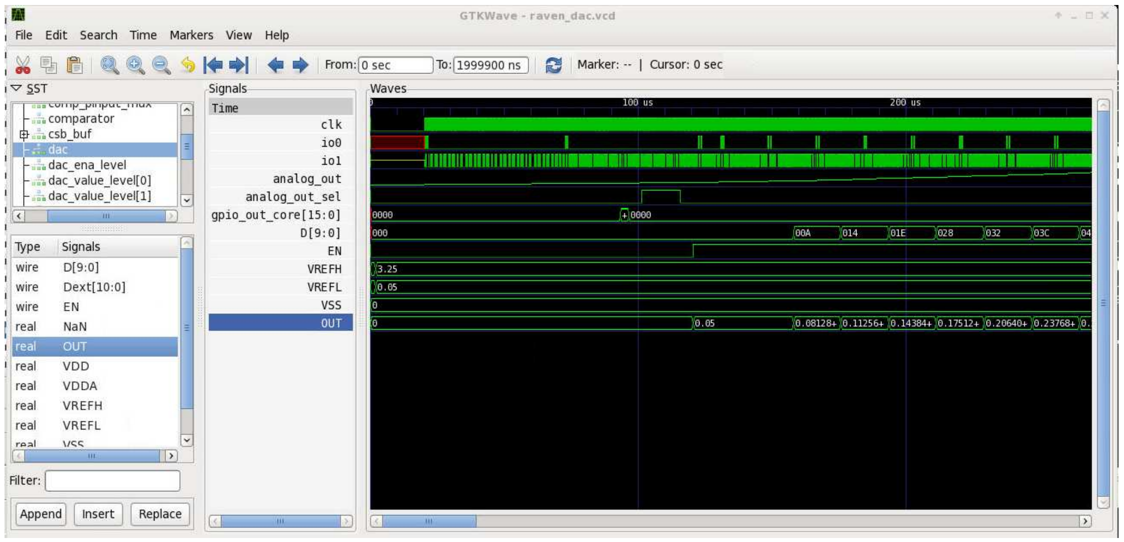
Simulation testbenches written in C and verilog. Makefile generates hex file from C code.

Simulate with iverilog

View waveforms with GTKwave

One testbench program for each system function to validate.

Analog blocks have behavioral verilog with real-valued I/O to enable full-chip simulation



DRC

Subcells assumed validated. DRC only looks at the top level wiring and interactions with abstracted subcells.

LVS

Abstracted subcells are “black boxes”

Building the Raven chip

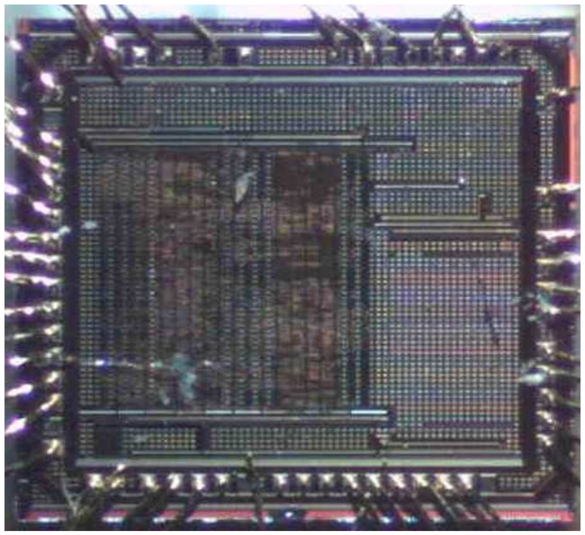
Part 7: Tape-out

GDS generated with Magic running in batch mode.

GDS is proprietary; GDS generation is not a user-accessible tool on the efabless platform.

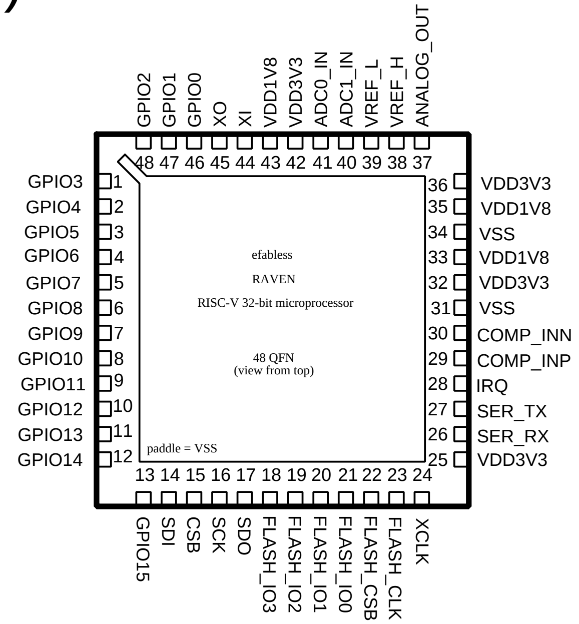
Foundry (X-Fab) runs its own DRC for sign-off verification

Shuttle run is slow (5 months) and limited (two process variants, 4- and 6-metal back-end stacks)


Raven chip die photo

Standard packaging (QFN-48), outsourced.

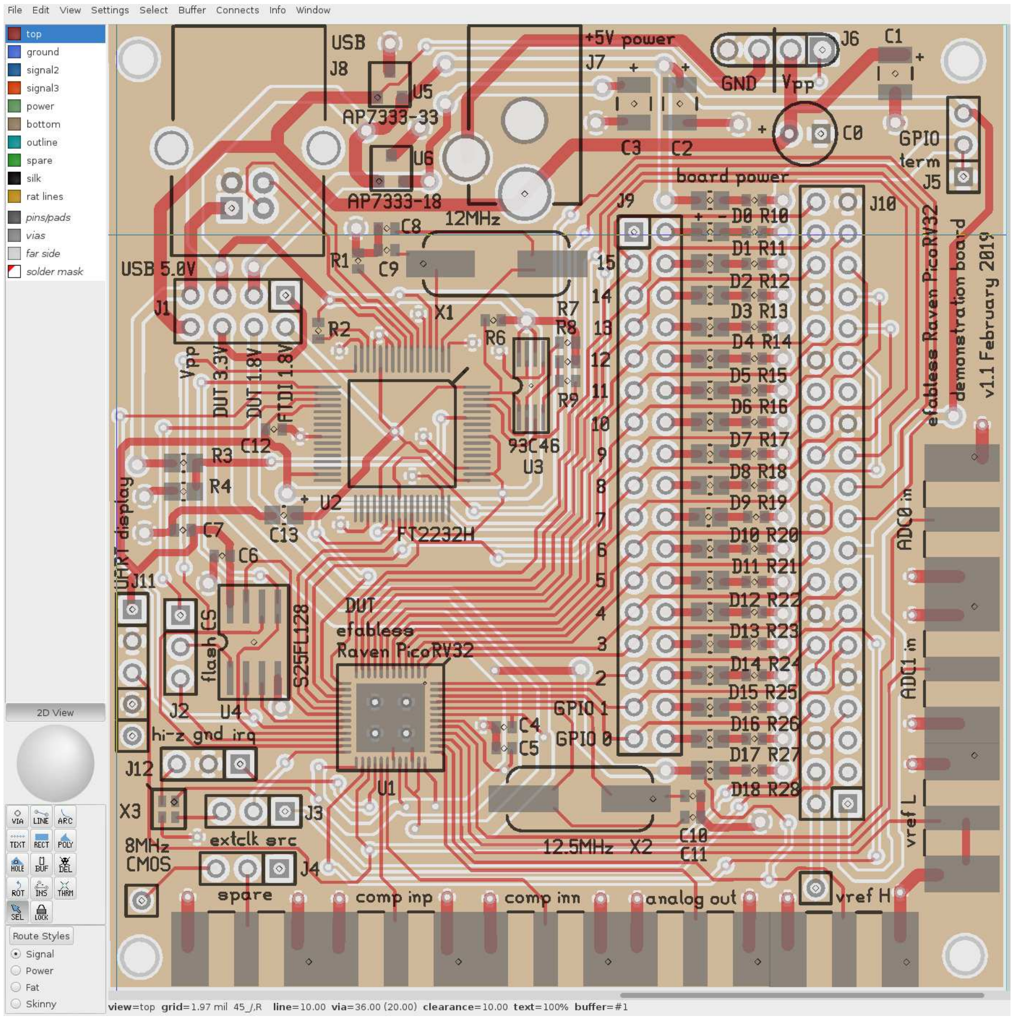
Wire bond diagram is open source on github.



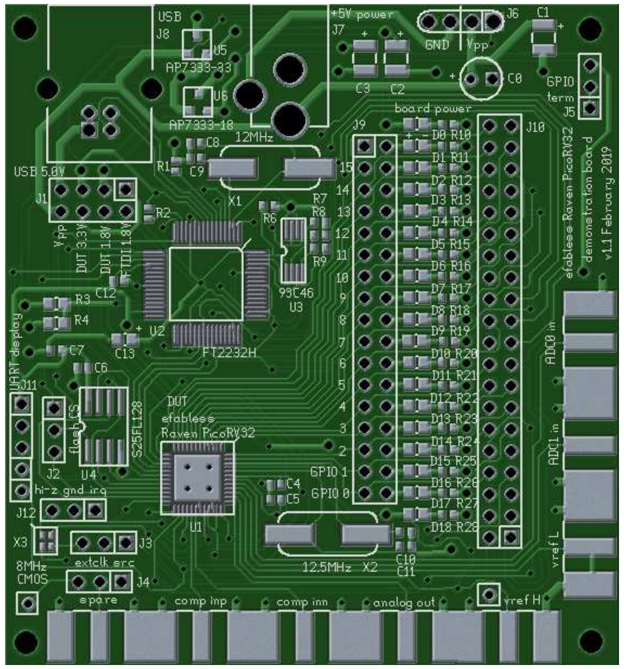
Building the Raven chip

Part 8: Designing the test board with gEDA's Pcb

Open source, of course: http://pcb.geda-project.org



File Edit View Settings Select Buffer Connects Info Window
top
ground
signal2
signal3
power
bottom
outline
spare
silk
rat lines
pins/pads
vias
far side
solder mask
2D View
Route Styles
Signal
Power
Fat
Skinny
view=top grid=1.97 mil 45_/_R line=10.00 via=36.00 (20.00) clearance=10.00 text=100% buffer=#1



Board is designed for characterization, not for ease of use.

Building the Raven chip

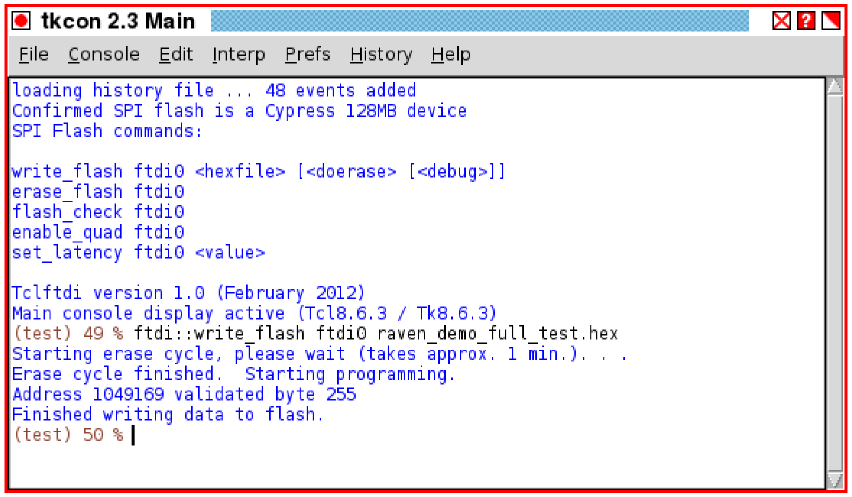
Part 9: Test environment with Tclftdi

http://opencircuitdesign.com/tclftdi

Tool uses the same interactive/batch methods as Magic, netgen, etc.

Tool uses libftdi (or FTDI's D2XX) libraries for communicating with the on-board FTDI chip, which communicates with the SPI flash and the Raven housekeeping SPI.

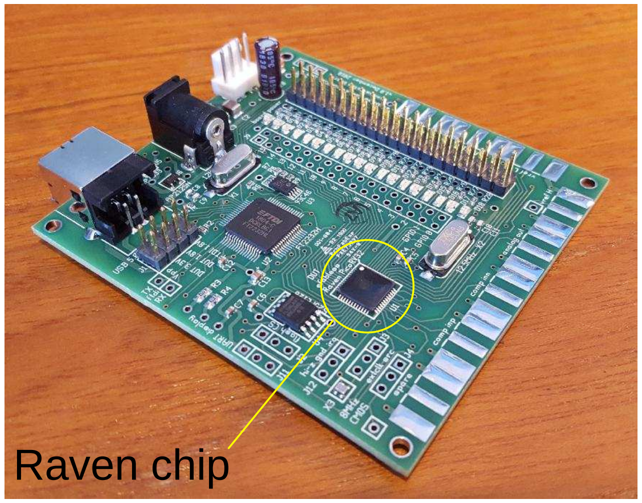
Assembled test board



```
loading history file ... 48 events added
Confirmed SPI flash is a Cypress 128MB device
SPI Flash commands:

write_flash ftdi0 <hexfile> [<doerase> [<debug>]]
erase_flash ftdi0
flash_check ftdi0
enable_quad ftdi0
set_latency ftdi0 <value>

Tclftdi version 1.0 (February 2012)
Main console display active (Tcl8.6.3 / Tk8.6.3)
(test) 49 % ftdi::write_flash ftdi0 raven_demo_full_test.hex
Starting erase cycle, please wait (takes approx. 1 min.). . .
Erase cycle finished.  Starting programming.
Address 1049169 validated byte 255
Finished writing data to flash.
(test) 50 %
```



Tclftdi running with script for flash access and programming

Building the Raven chip

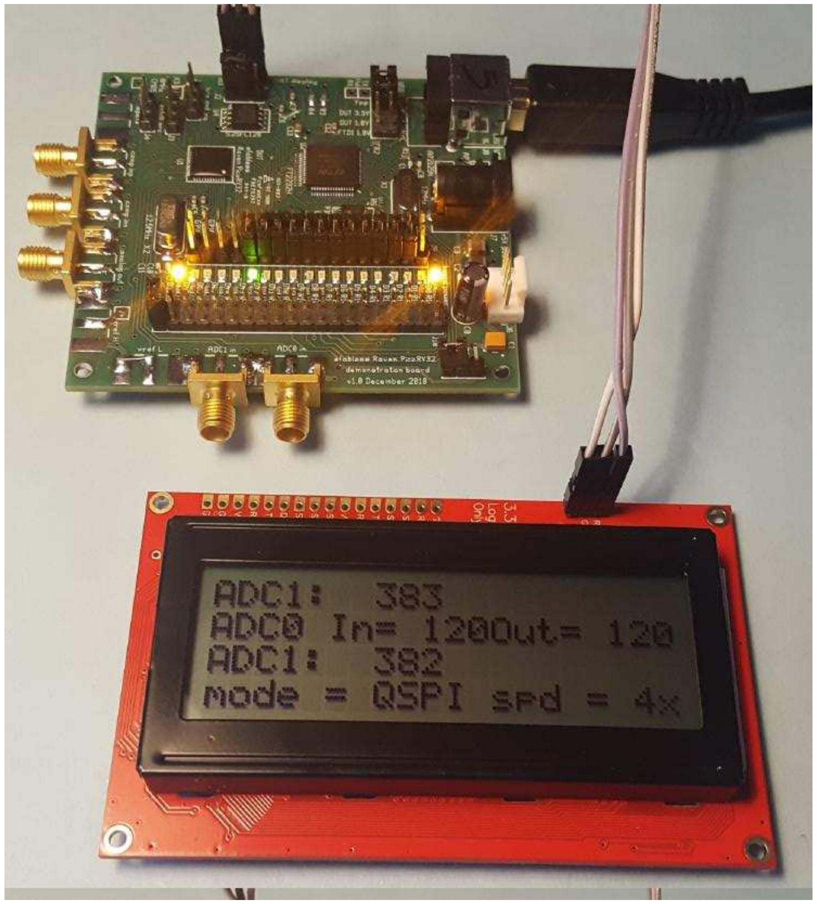
Part 10: Testing

Demonstration program written in C and compiled with gcc from the raven “standalone” version in the git repository:

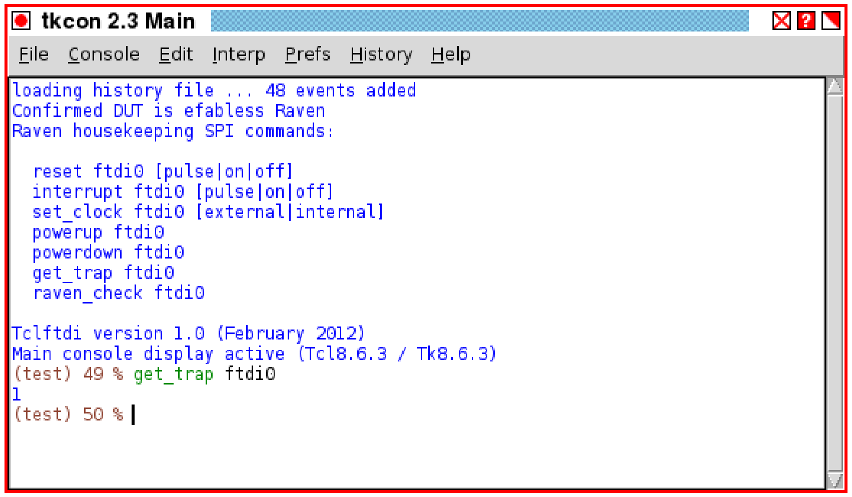
https://github.com/efabless/raven-picorv32

Demo program shows flash SPI modes, DAC and ADC operation, RC oscillator, and GPIO functions (via LEDs).

The Raven board running 100 MHz core clock



26.8 mA (88 mW)



```
loading history file ... 48 events added
Confirmed DUT is efabless Raven
Raven housekeeping SPI commands:

  reset ftdi0 [pulse|on|off]
  interrupt ftdi0 [pulse|on|off]
  set_clock ftdi0 [external|internal]
  powerup ftdi0
  powerdown ftdi0
  get_trap ftdi0
  raven_check ftdi0

Tclftdi version 1.0 (February 2012)
Main console display active (Tcl8.6.3 / Tk8.6.3)
(test) 49 % get_trap ftdi0
1
(test) 50 %
```

Tclftdi running with script for Raven housekeeping SPI access

Current tool and platform limitations:

No IR drop analysis

No dynamic power estimation (or static, but that's easier)

No electromigration analysis

DRC rules at 90nm feature size and lower get increasingly difficult

Run-length rules not handled in the router

Handling of non-standard cell macros is a work in progress

Top-level routing is feasible but still a work in progress

No timing-aware synthesis

Practical size limitation of about 50k gates

Certain standard cell libraries generate lots of DRC errors

No DFT (scan chain insertion, ATPG)

Future tool and platform development:

Looking to ABK Open Road for large designs and cutting-edge technologies

https://github.com/abk-openroad

qflow version 1.4 under development

- OpenSTA and OpenTimer integration done
- Move from BLIF netlists to verilog
- Hard macro handling and top-level routing (in progress)

Starting work on DFT in qflow

- Based on Atalanta ATPG

Additional foundry process PDKs under development at efabless

Work on XH018, XH035 support for OpenRAM
(one less proprietary block)

https://github.com/VLSIDA/OpenRAM

Conclusions:

End-to-end open source hardware is possible, although transistor level descriptions (i.e., GDS) remain elusive without open foundries

On mature process nodes (e.g., 0.18µm), using best practices and reasonable margins, open source EDA tools are capable of making production−grade chips

First-time silicon success is possible with open source EDA tools

Community involvement makes open source happen

Go forth and code!

Go forth and design! https://efabless.com/toolbox/design
https://github.com/efabless/raven-picorv32

This presentation was written with xcircuit
http://opencircuitdesign.com/xcircuit