From Production Rule Set (PRS) to tape-out with industry standard tools

Async Summer School 24 – 15-07-2024

Ole Richter
Bio-Inspired Circuit and Systems
Rijksuniversiteit Groningen

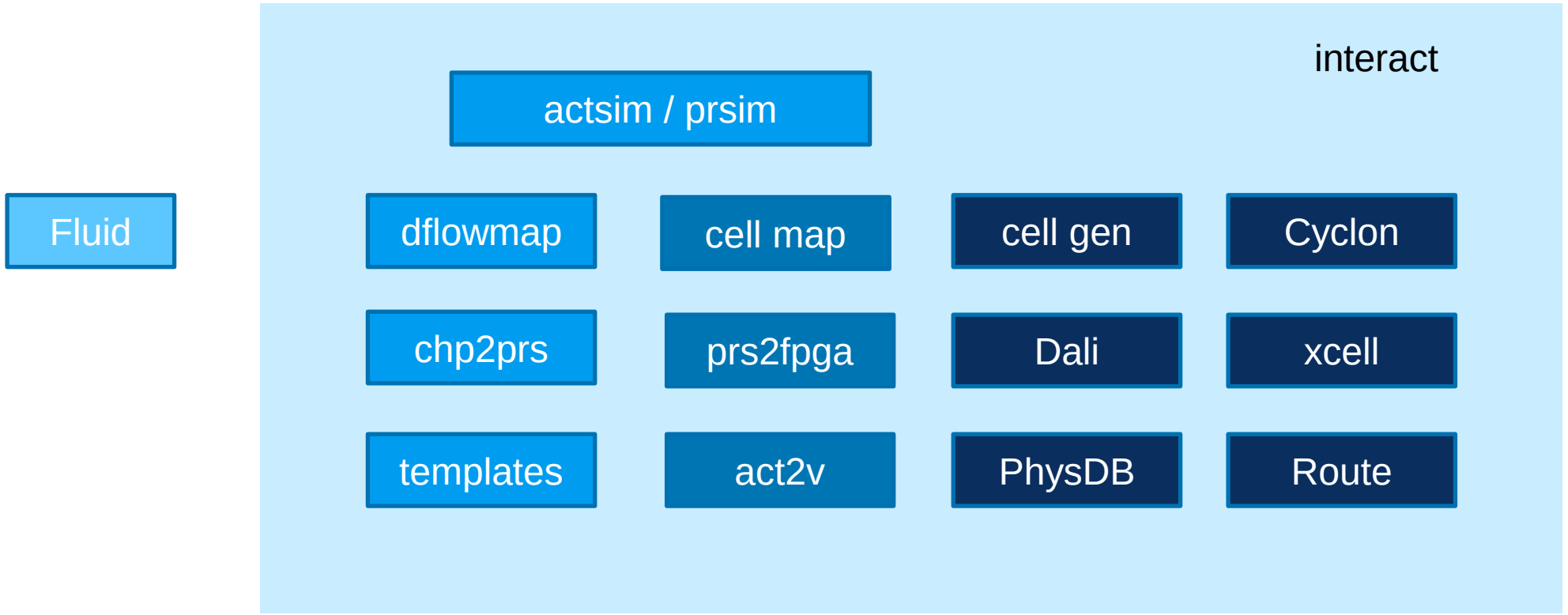
Innovation vs. risk management

- Complex and unconventional projects due to mixed-signal nature
- Design reuse and risk minimisation
- Tight project timeline: 1 month prep, 2 month exec
  (reality +2 month with <50% reduced team)

- Design flow migration problematic: new training and unforeseen roadblocks

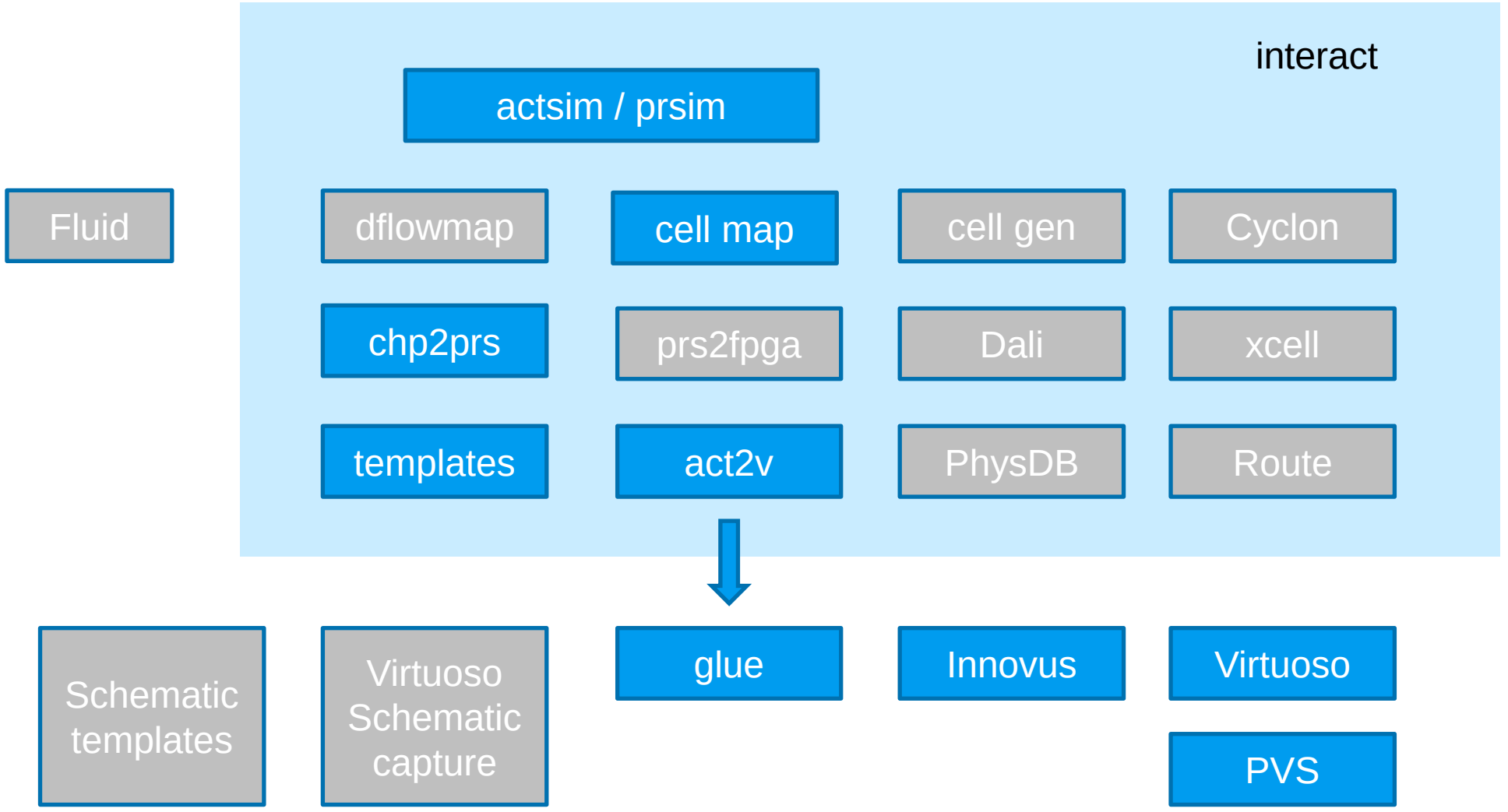
- Migration in steps: first digital FE flow only

ACT Flow



non exhaustive selection

Our Flow


interact
actsim / prsim
Fluid
dflowmap
cell map
cell gen
Cyclon
chp2prs
prs2fpga
Dali
xcell
templates
act2v
PhysDB
Route
Schematic templates
Virtuoso Schematic capture
glue
Innovus
Virtuoso
PVS

Detour: What did we build?

Texel

PI: E. Chicca, G. Indiveri


Arianna


Junren


Philipp

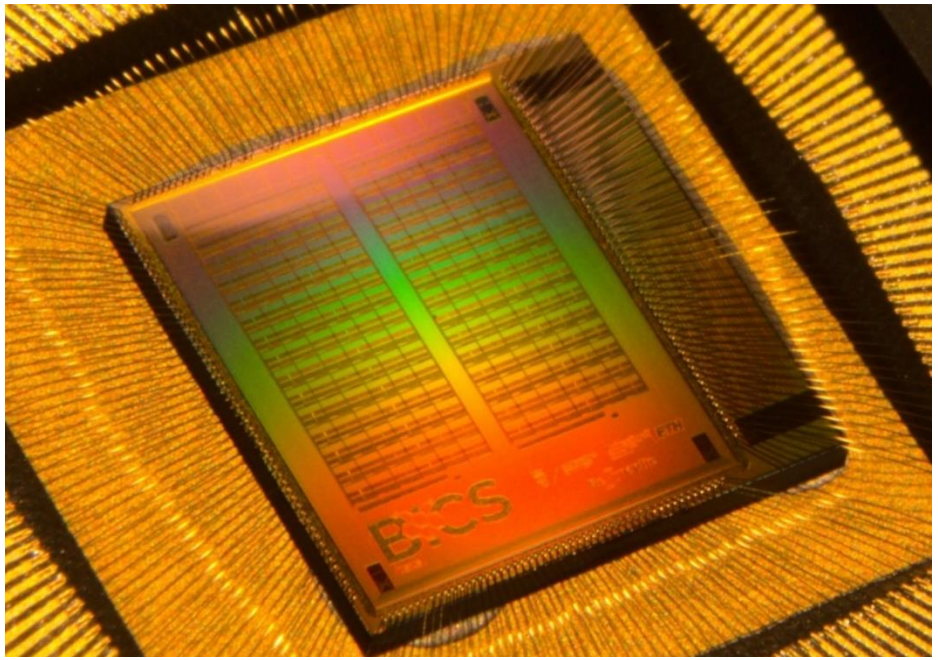

Michele

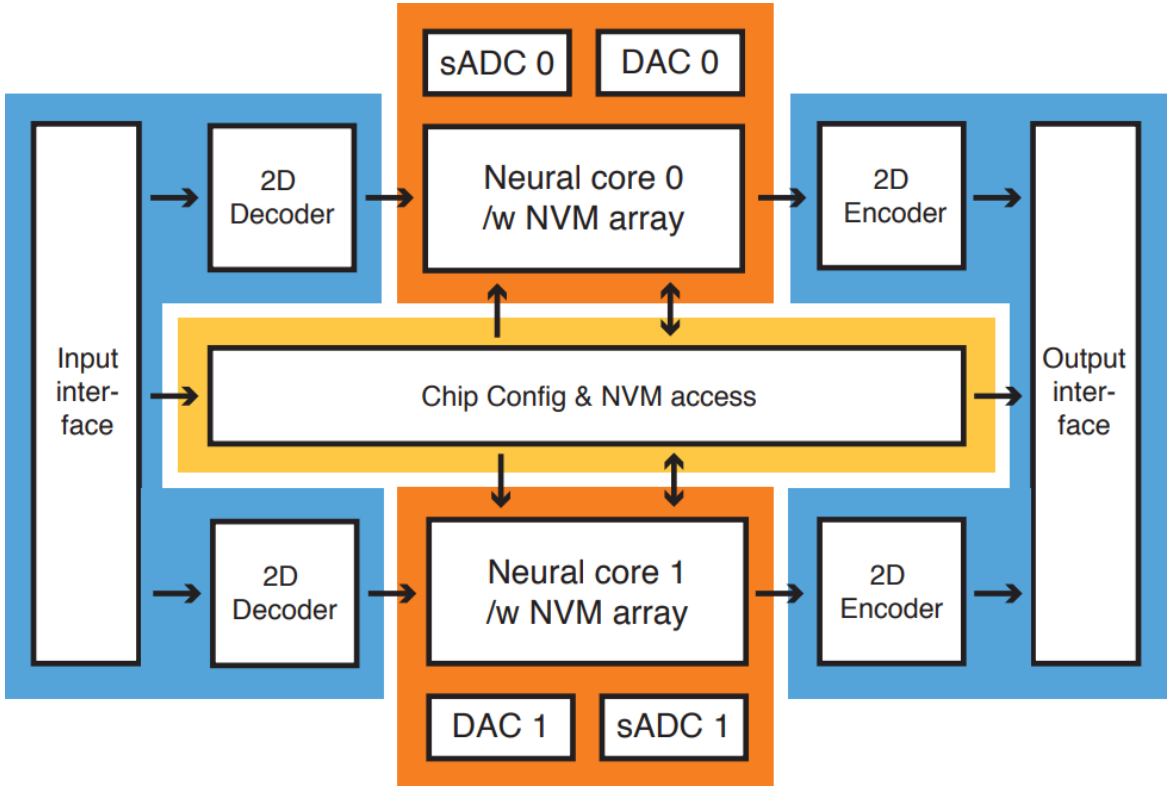

Willian


Madison


Hugh


Maxime





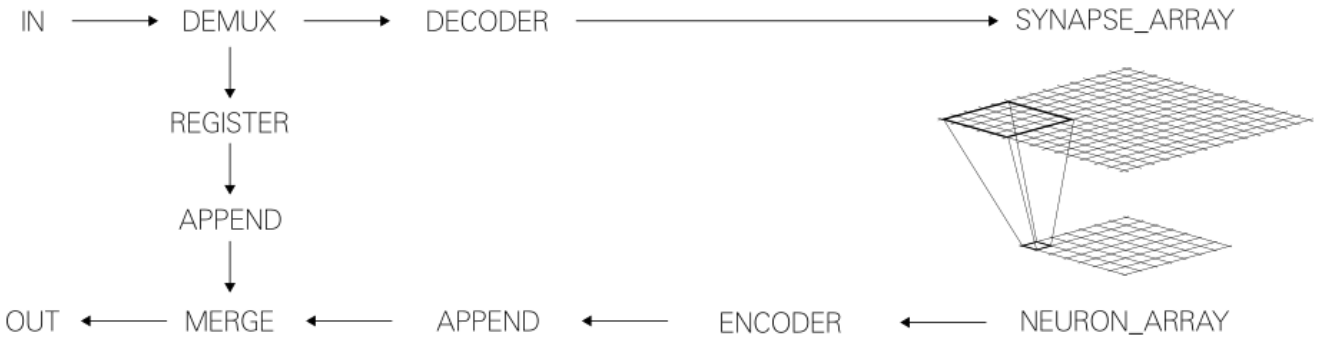
Templates


Madison

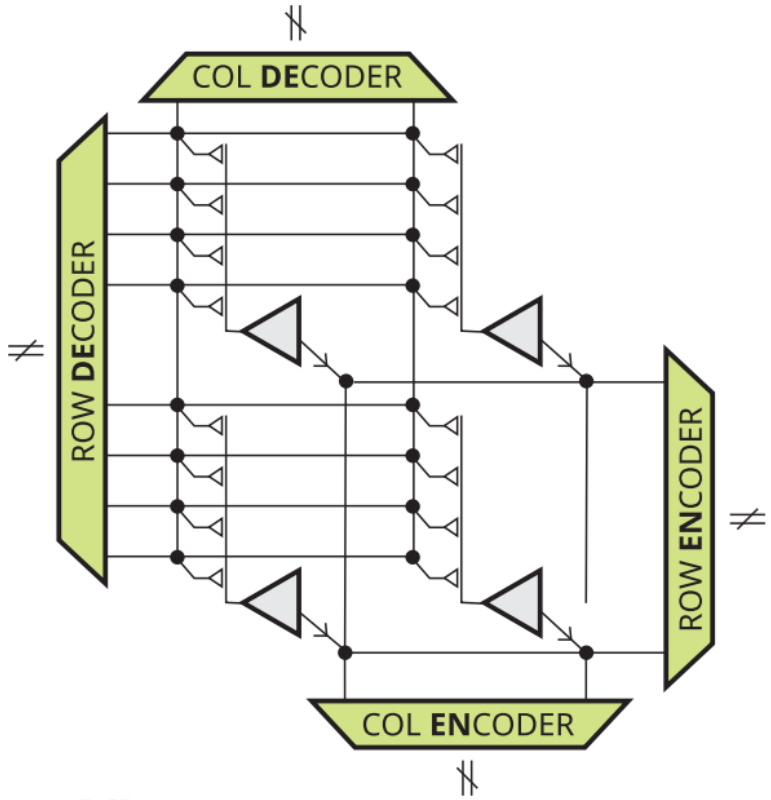

Hugh


Michele

Single core infrastructure:



Array infrastructure:

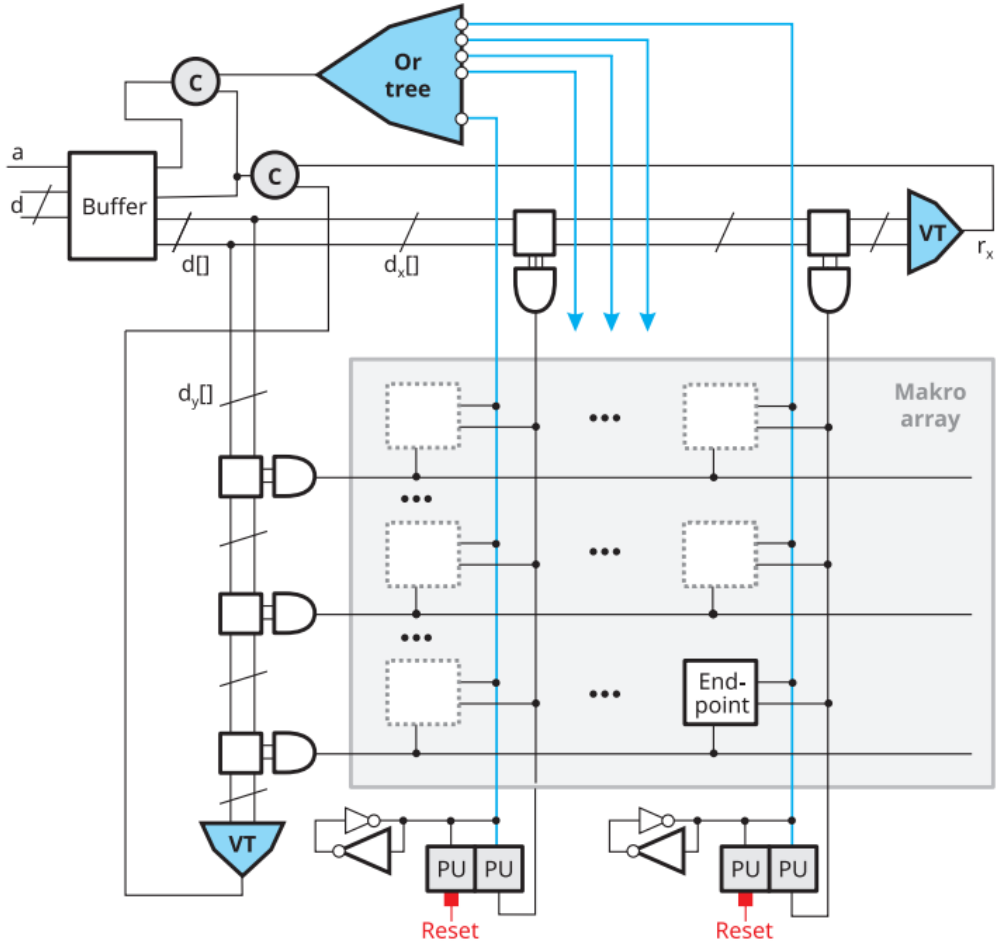


PRS Templates

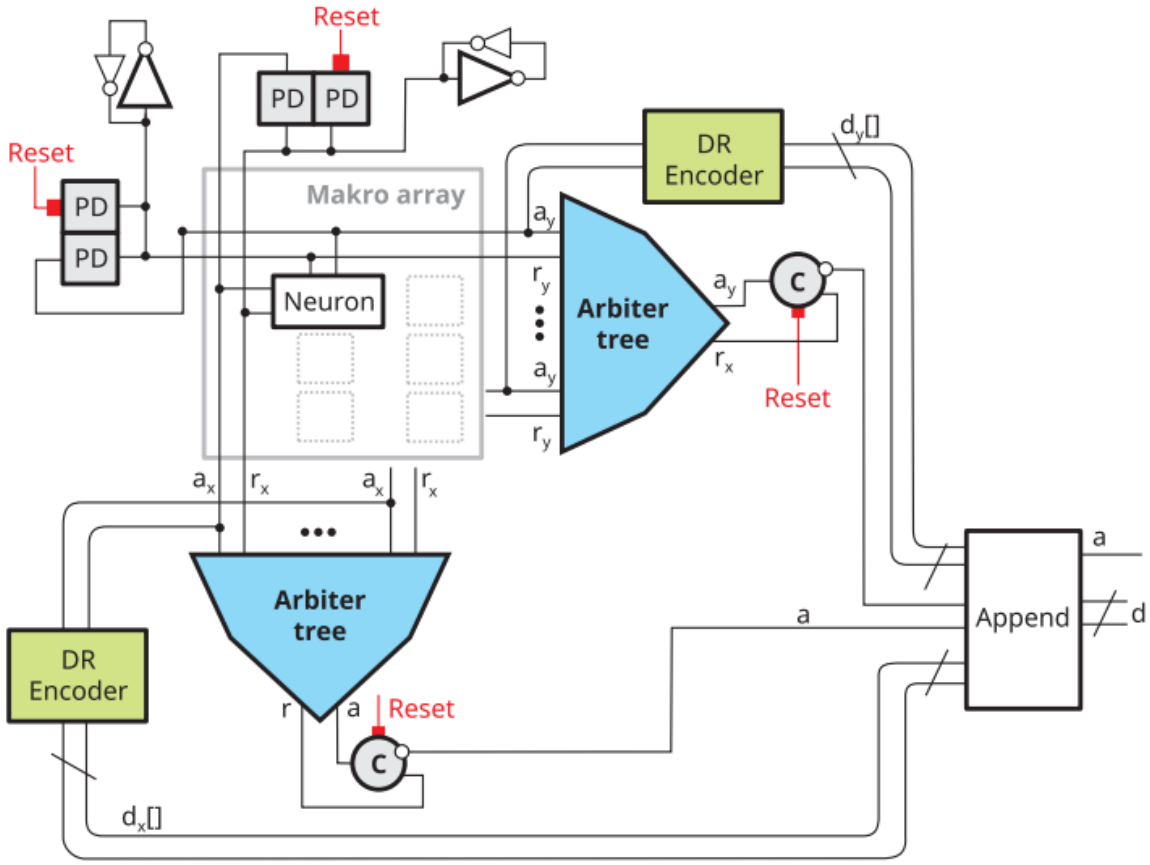

Madison


Hugh


Michele

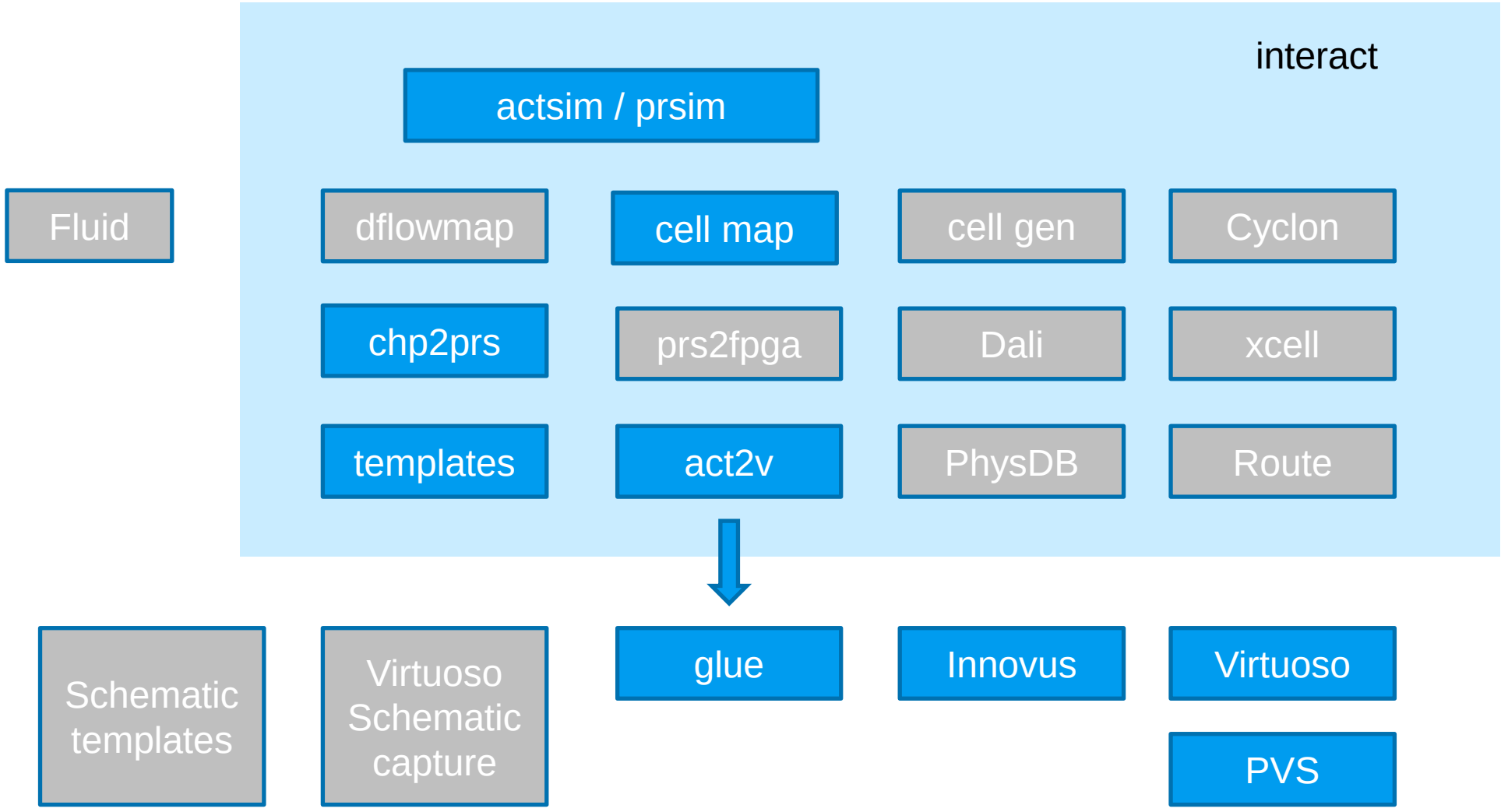


2D receiver - decoder



2D transmitter - encoder

Our Flow


interact
actsim / prsim
Fluid
dflowmap
cell map
cell gen
Cyclon
chp2prs
prs2fpga
Dali
xcell
templates
act2v
PhysDB
Route
Schematic templates
Virtuoso Schematic capture
glue
Innovus
Virtuoso
PVS

PRsim


Madison


Hugh


Michele

PRsim init:

actlib_dataflow_neuro > test > unit_tests > init_qdi.prsim

```
1   initialize
2   load-scm "helper.scm"
3   random
4   set GND 0
5   set Vdd 1
6   set Reset 1
```

Unit test:

```
17   mode run
18   system "echo '[] Set reset 0'"
19   status X
20   set Reset 0
21   cycle
22
23   system "echo '[] Sending in a 7 packet'"
24   set-qdi-channel-valid "e.in" 3 7
25   cycle
26   assert e.out[0].r 0
27   assert e.out[1].r 0
```

Test instance:

```
38  open std::data;
39
40  open tmpl::dataflow_neuro;
41
42  defproc decoder_2d_hs_2x4 (avMx1of2<3> in; a1of1 out[8]){
43      bool _reset_B;
44      prs {
45          Reset => _reset_B-
46      }
47      power supply;
48      supply.vdd = Vdd;
49      supply.vss = GND;
50
51      decoder_2d_hs<1,2,2,4> decoder(.in = in, .out = out,
52          .reset_B = _reset_B, .supply = supply);
```

› Now: actsim with addon to read csv-test-vectors (test_bench_lib - public)

Cell mapping


Madison


Hugh


Michele

Template example:

```
540  export
541  defproc arbiter_handshake(a1of1 in1; a1of1 in2; a1of1 out; power supply)
542  {
543    bool _y1_arb,_y2_arb;
544
545    A_2C_B_X1 ack_cell1(.c1 = out.a,.c2 = _y1_arb,.y = in1.a,.vdd = supply.vdd, .vss = supply.vss);
546    A_2C_B_X1 ack_cell2(.c1 = out.a,.c2 = _y2_arb,.y = in2.a,.vdd = supply.vdd, .vss = supply.vss);
547    OR2_X1 or_cell(.a = _y1_arb, .b = _y2_arb, .y = out.r,.vdd = supply.vdd, .vss = supply.vss);
548    ARBITER arbiter(.a = in1.r, .b = in2.r, .c = in2.a, .d = in1.a, .y1 = _y1_arb, .y2 = _y2_arb, .vdd = supply.vdd, .vss = supply.vss);
549
550  }
```

Cell example:

```
380  export defcell A_2C_B_X1 (bool ! y; bool? c1, c2; bool? vdd, vss)
381  {
382  bool _y;
383  prs{
384      ~c1 & ~c2 -> _y+
385      c1 & c2 -> _y-
386      _y => y-
387  }
388  sizing {
389          leak_adjust <- 1;
390          p_n_mode <- 1;
391      y {-1}; _y{-1}}
392  }
```

Change defcell to defproc
for full act cell mapping

act2v


Madison


Hugh


Michele

```
export template<pint N>
defproc sigbuf (bool? in; bool! out[N]; power supply)
{

  { N >= 0 : "sigbuf: parameter error" };
//  { N <= 43 : "sigbuf: parameter error, N too big" };

    /* -- just use in sized driver here -- */
    [ N <= 4 ->
    BUF_X1 buf1 (.a = in, .y = out[0], .vdd = supply.vdd, .vss = supply.vss);
    [] N >= 5 & N <= 7 ->
    BUF_X2 buf2 (.a = in, .y = out[0], .vdd = supply.vdd, .vss = supply.vss);
    [] N >= 8 & N <= 10 ->
    BUF_X3 buf3 (.a = in, .y = out[0], .vdd = supply.vdd, .vss
   [] N >= 11 & N <= 14 ->
    BUF_X4 buf4 (.a = in, .y = out[0], .vdd = supply.vdd, .vss
   [] N >= 15 & N <= 18 ->
    BUF_X6 buf6 (.a = in, .y = out[0], .vdd = supply.vdd, .vss
   [] N >= 19 & N <= 29 ->
    BUF_X8 buf8 (.a = in, .y = out[0], .vdd = supply.vdd, .vss
   [] N >= 30 & N<= 48->
    BUF_X12 buf12 (.a = in, .y = out[0], .vdd = supply.vdd, .vs
```

```
//
// Verilog module for: sigbuf<47>
//
module _0_0tmpl_0_0dataflow__neuro_0_0sigbuf_347_4(in, \out[0] );
   input in;
   output \out[0] ;

// -- signals ---
   reg \out[0] ;
   wire in;

// --- instances
 _0_0tmpl_0_0dataflow__neuro_0_0BUF__X12 \buf12  (.y(\out[0] ), .a(in));
endmodule
```

Sanitisation – the glue


Madison


Hugh


Michele

Act2v:

```
32  //
33  // Verilog module for: sigbuf<47>
34  //
35  module _0_0tmpl_0_0dataflow__neuro_0_0sigbuf_347_4(in, \out[0] );
36     input in;
37     output \out[0] ;
38
39  // -- signals ---
40     reg \out[0] ;
41     wire in;
42
43  // --- instances
44  _0_0tmpl_0_0dataflow__neuro_0_0BUF__X12 \buf12  (.y(\out[0] ), .a(in));
45  endmodule
```

Innovus compatible:

```
11  //
12  // Verilog module for: sigbuf<47>
13  //
14  module tmpl_0_0dataflow__neuro_0_0sigbuf_347_4(in, Iout0 , vdd, vss
15     input vdd;
16     input vss;
17     input in;
18
19
20  // -- signals ---
21     output Iout0 ;
22     wire in;
23
24  // --- instances
25  BUF_X12 Ibuf12  (.y(Iout0 ), .a(in), .vdd(vdd), .vss(vss));
26  endmodule
```

OA – the glue


Madison


Hugh


Michele

Innovus compatible:

```
11  //
12  // Verilog module for: sigbuf<47>
13  //
14  module tmpl_0_0dataflow__neuro_0_0sigbuf_347_4(in, Iout0 , vdd, vss);
15     input vdd;
16     input vss;
17     input in;
18
19
20  // -- signals ---
21     output Iout0 ;
22     wire in;
23
24  // --- instances
25  BUF_X12 Ibuf12  (.y(Iout0 ), .a(in), .vdd(vdd), .vss(vss));
26  endmodule
```

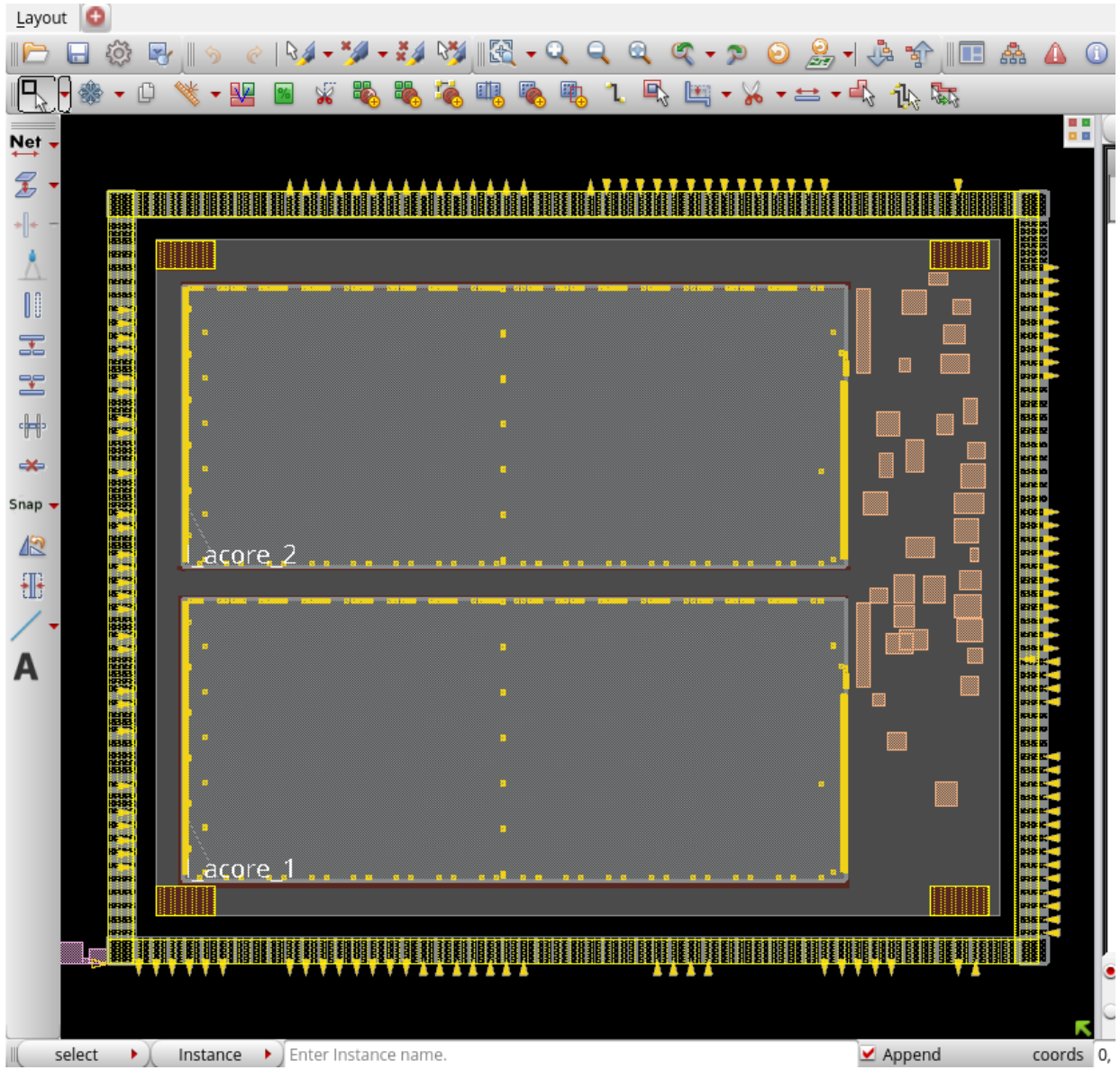
Virtuoso compatible:

```
├── tmpl_0_0dataflow__neuro_0_0sigbuf_347_4
│   └── netlist
│       ├── master.tag
│       ├── pc.db
│       └── verilog.v
├── tmpl_0_0dataflow__neuro_0_0sigbuf_348_4
│   └── netlist
│       ├── master.tag
│       ├── pc.db
│       └── verilog.v
├── tmpl_0_0dataflow__neuro_0_0sigbuf_35_4
│   └── netlist
```

Innovus

Name

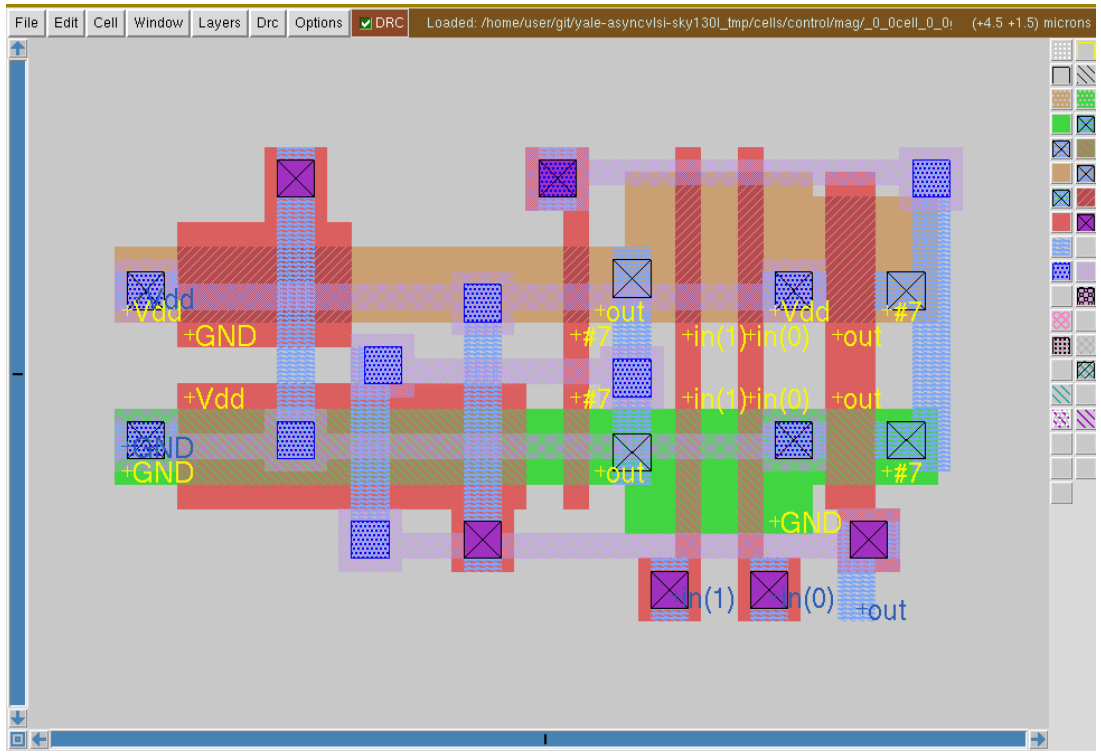
- 000_enable_multicore.tcl
- 010_init_design.tcl
- 020_padframe.tcl
- 030_place_floorplanmode.tcl
- 040_set_power_rings.tcl
- 050_place_blockages.tcl
- 060_place_design_fillers.tcl
- 070_set_powers.tcl
- 080_routing.tcl
- 090_export.tcl
- 090_export_padframe.tcl

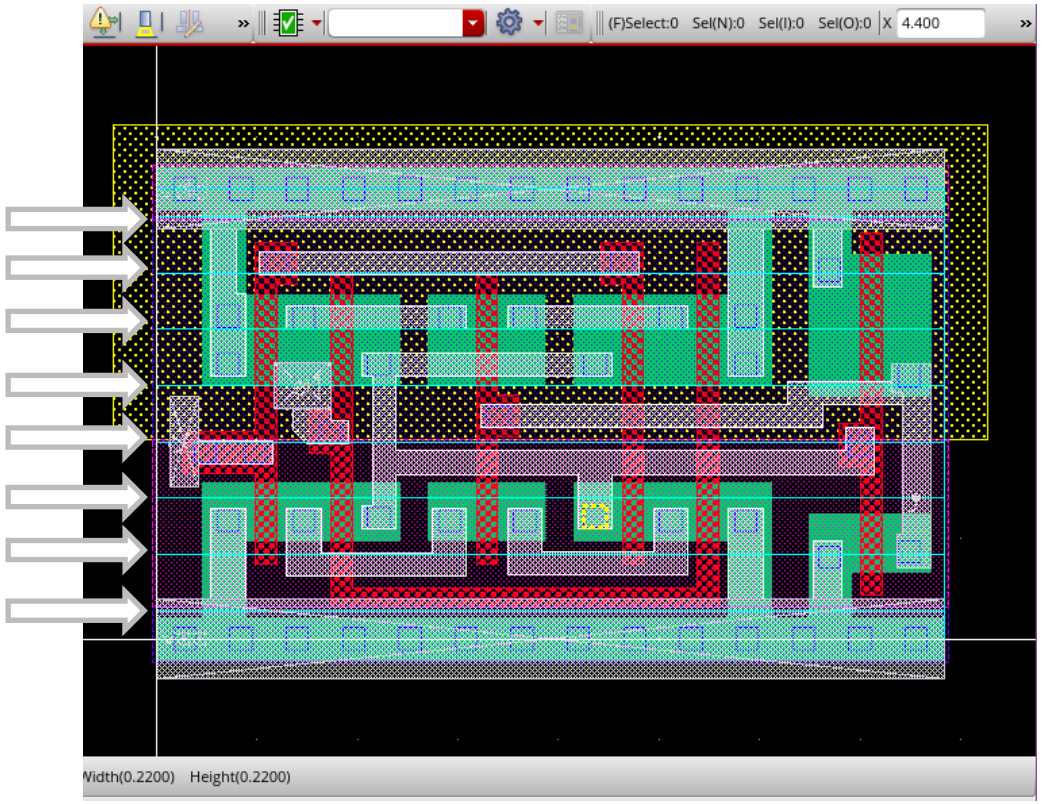



Arianna

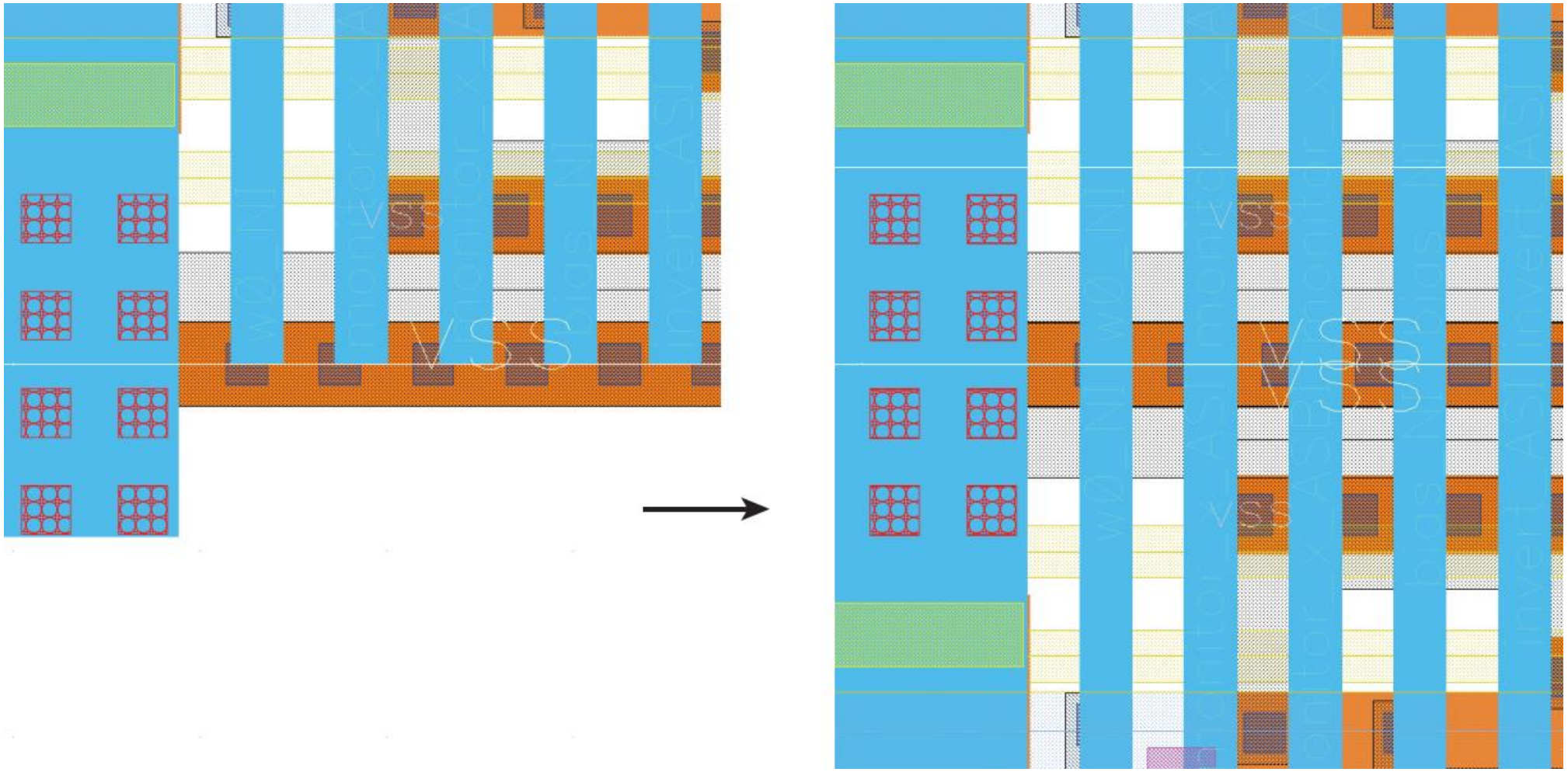

Willian

Grid and track based cells

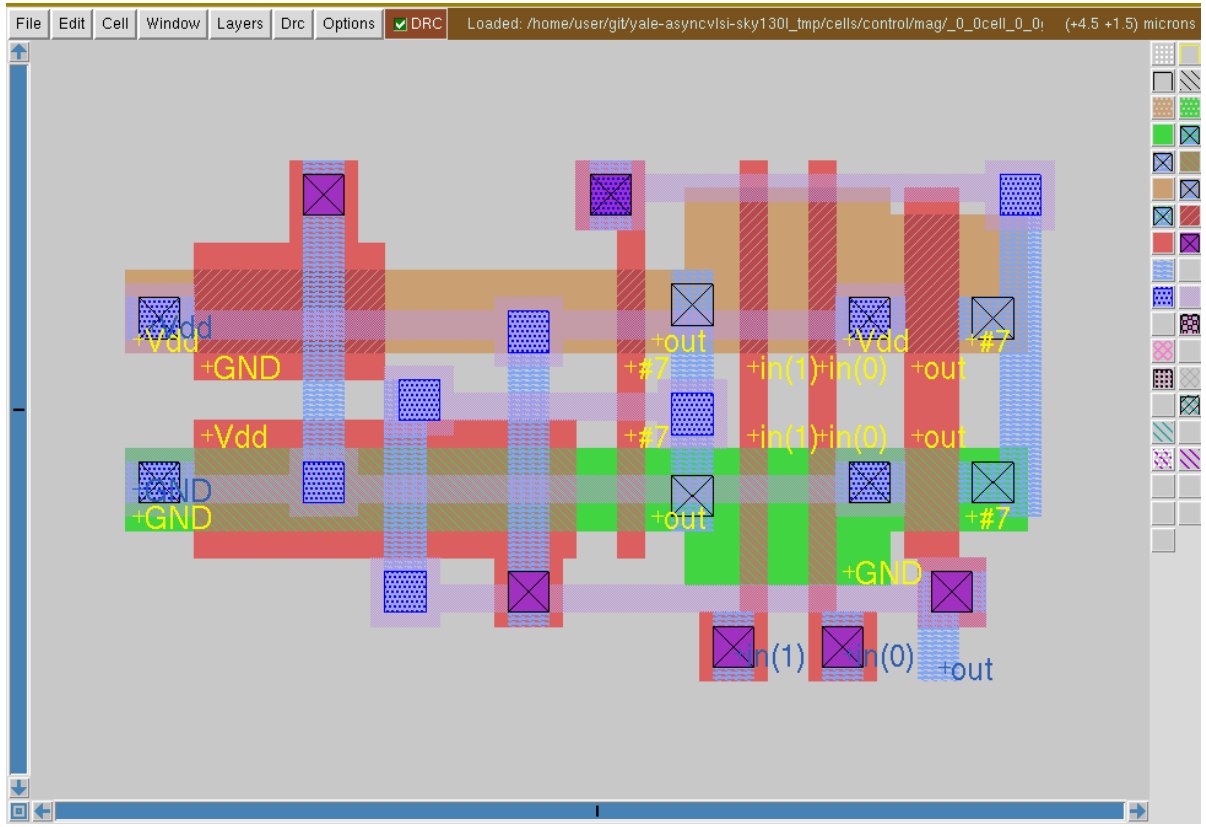


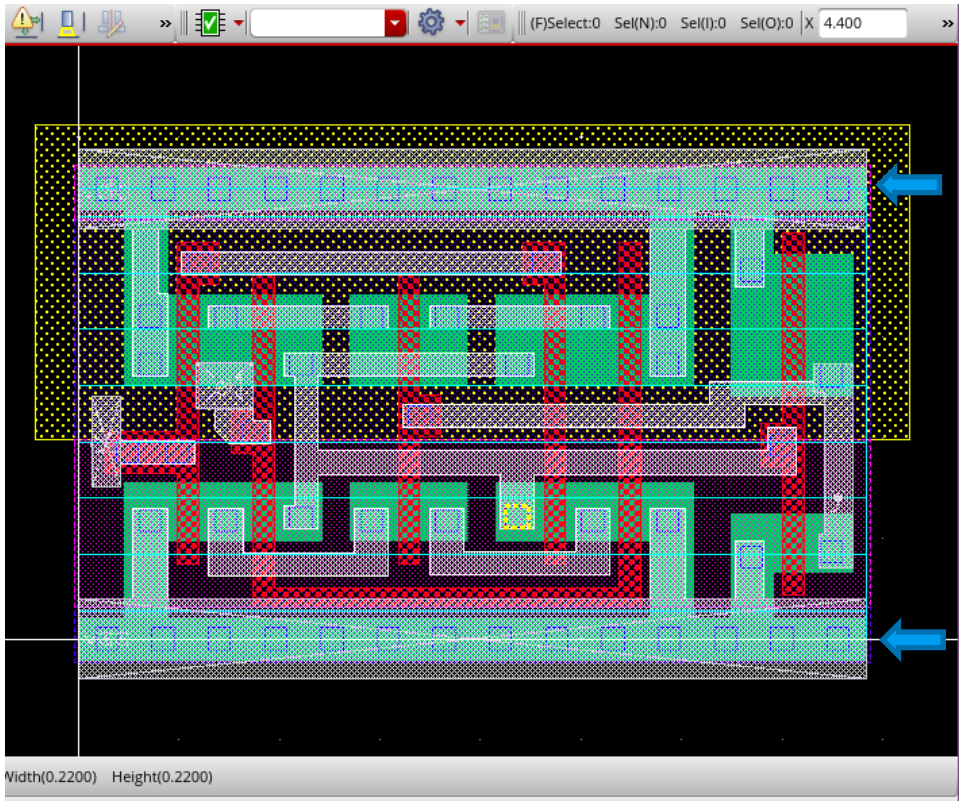


Macro tileing



Grid and track based cells





8T Cell library addon


Philipp

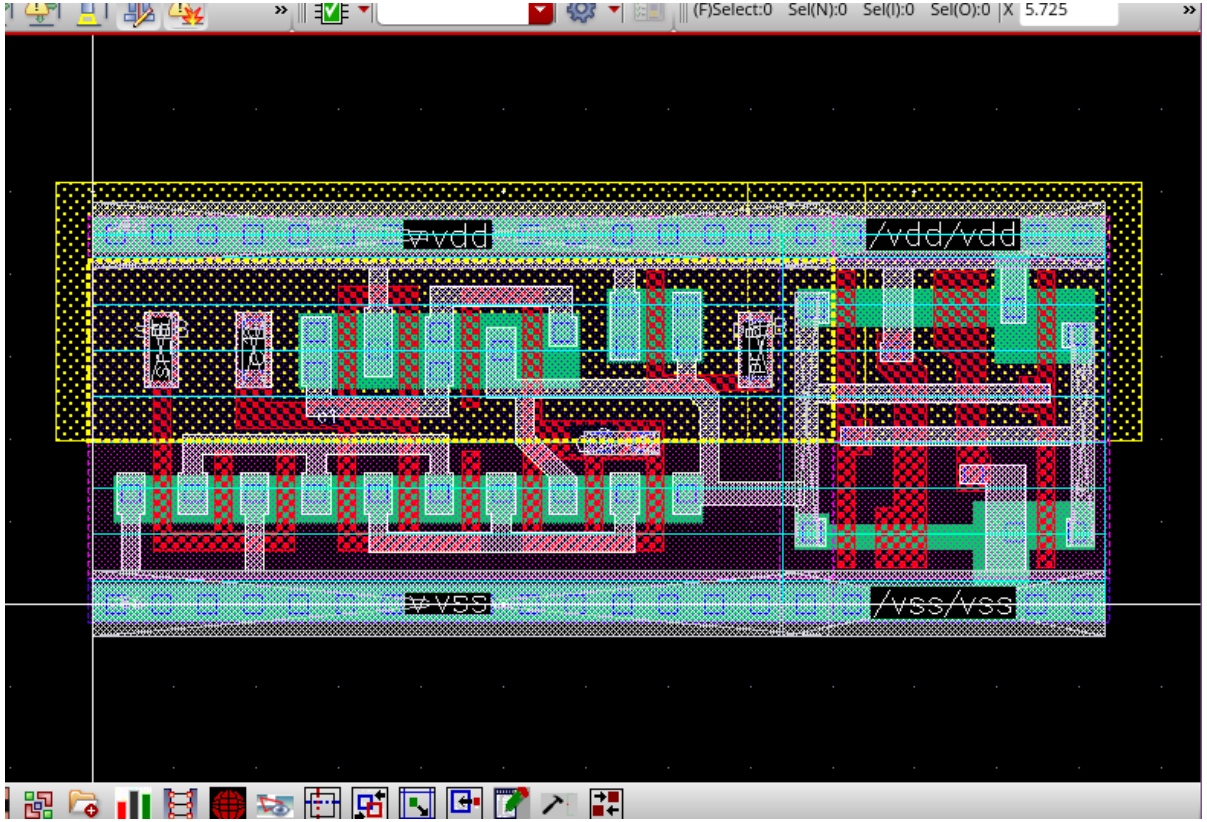

Michele

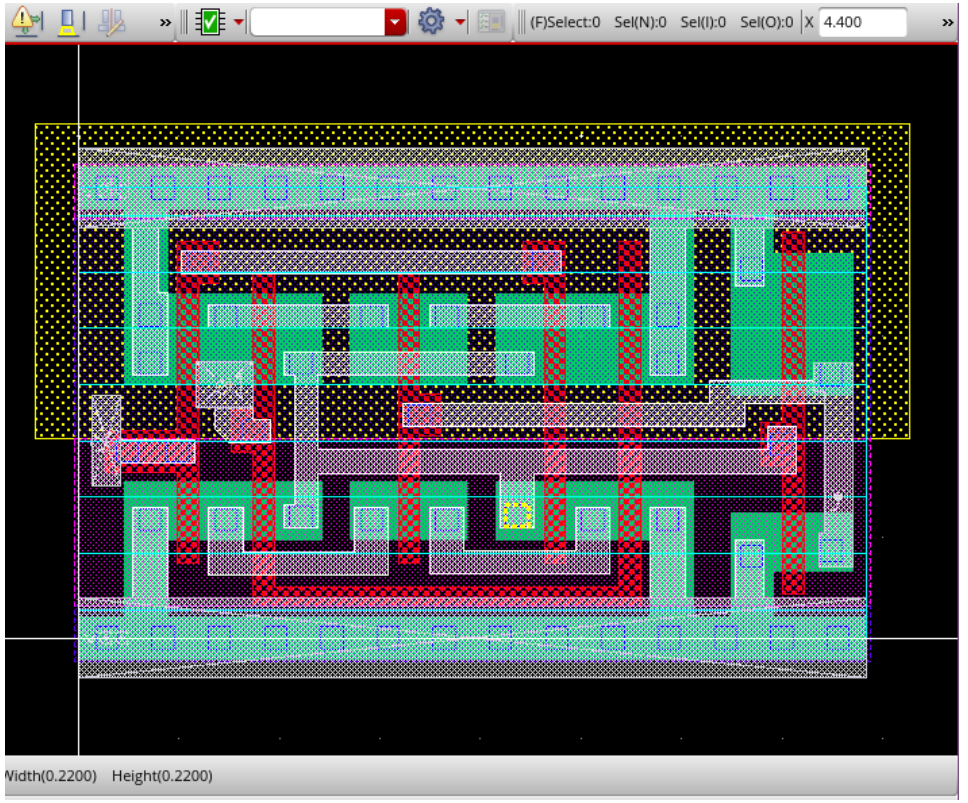

Willian


Madison


Hugh


Maxime





Place & Route


Elisabetta


Arianna

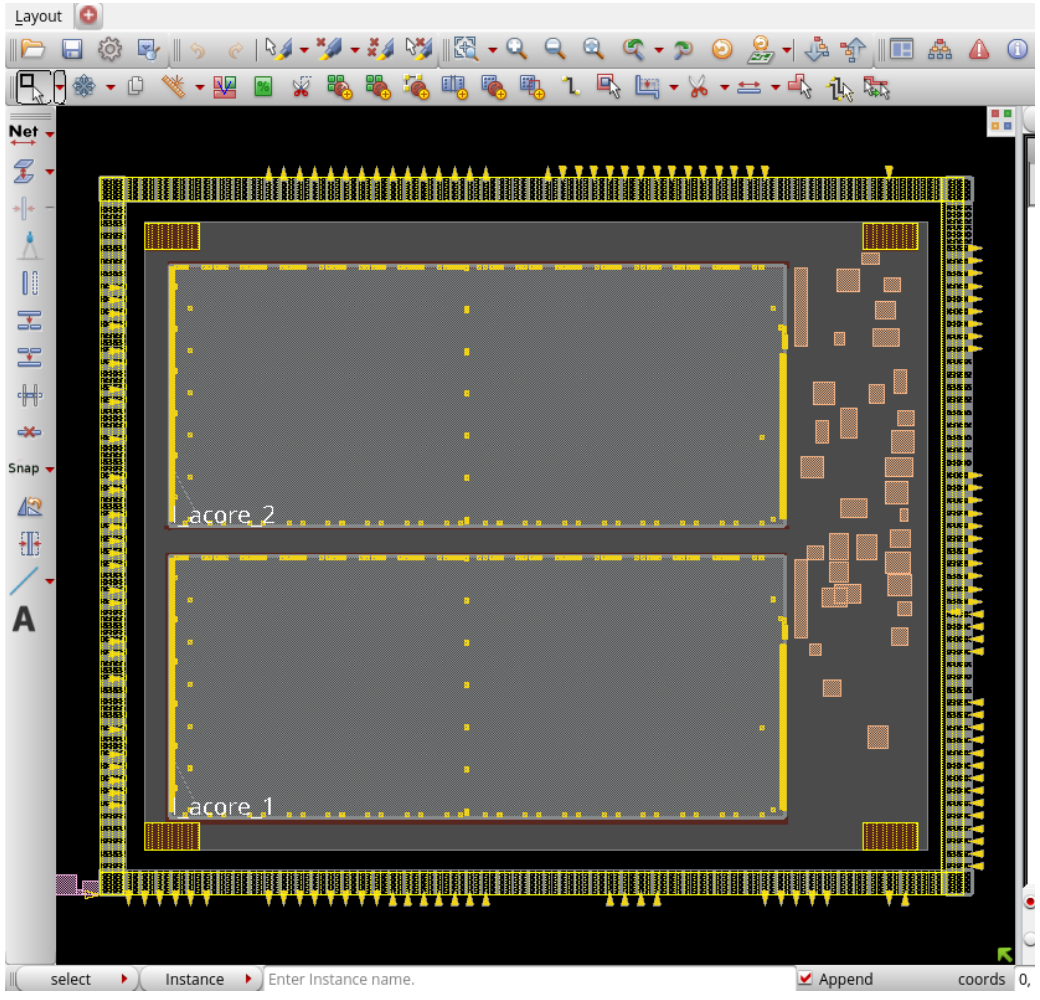

Philipp

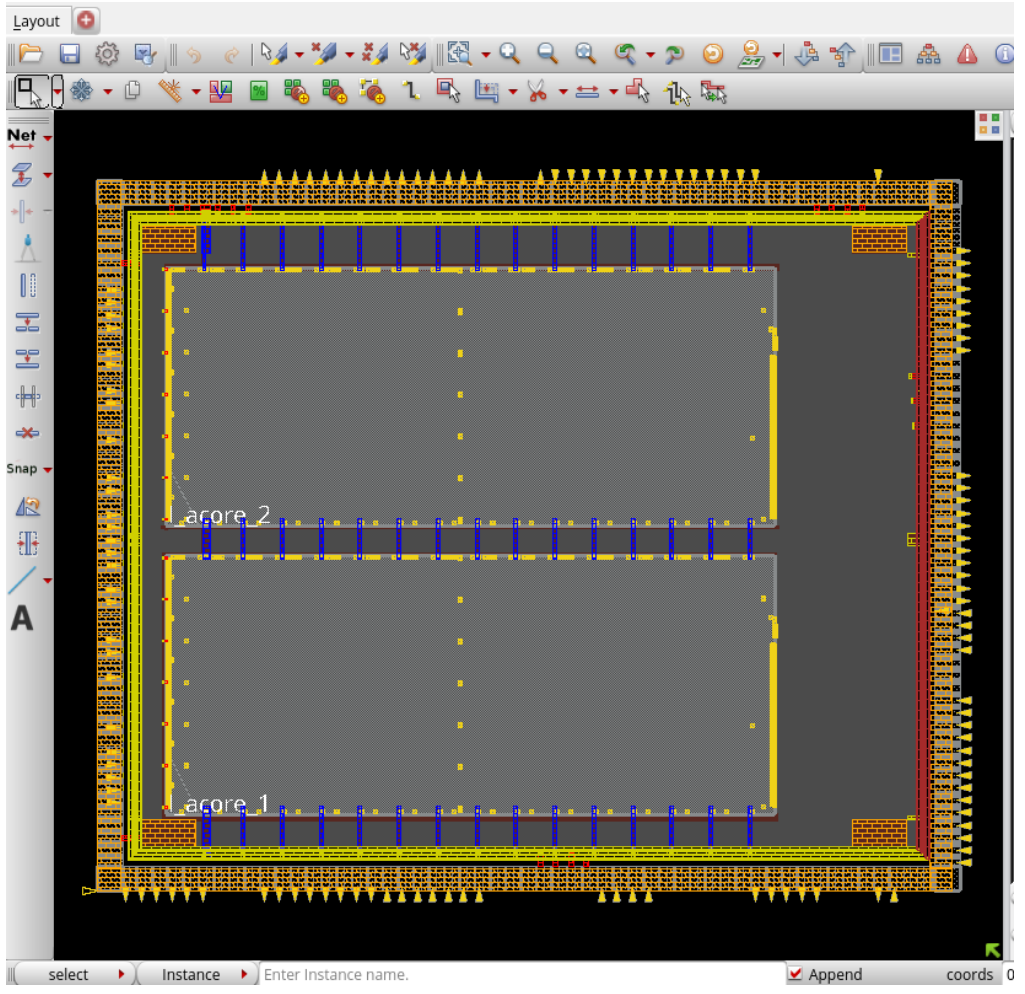

Willian


Madison


Hugh


Maxime





Place & Route


Elisabetta


Arianna

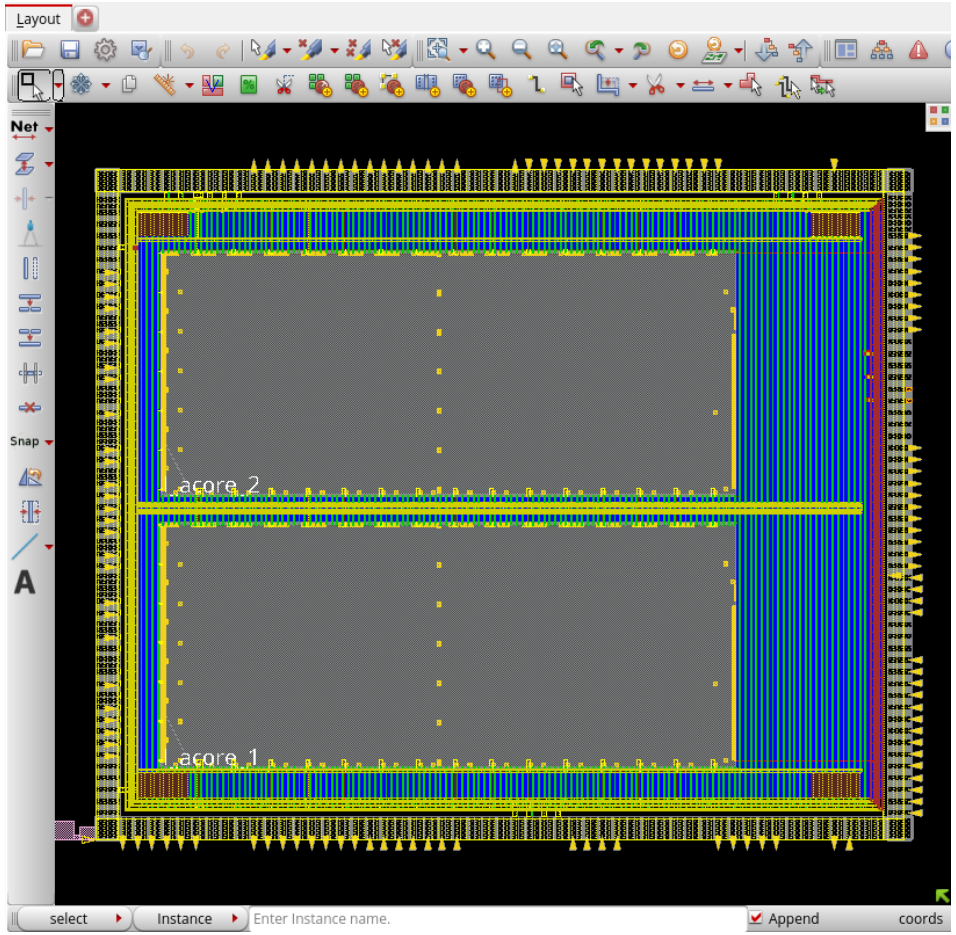

Philipp

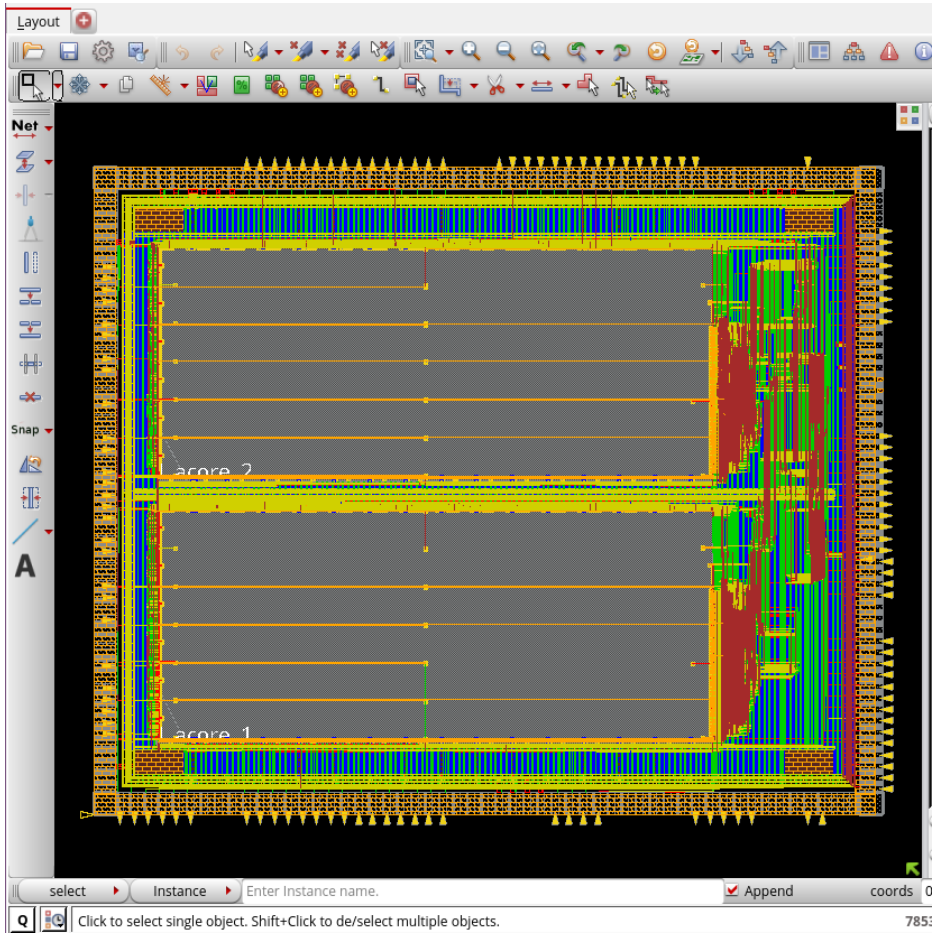

Willian


Madison


Hugh


Maxime





Virtuoso Physical Verification


Elisabetta

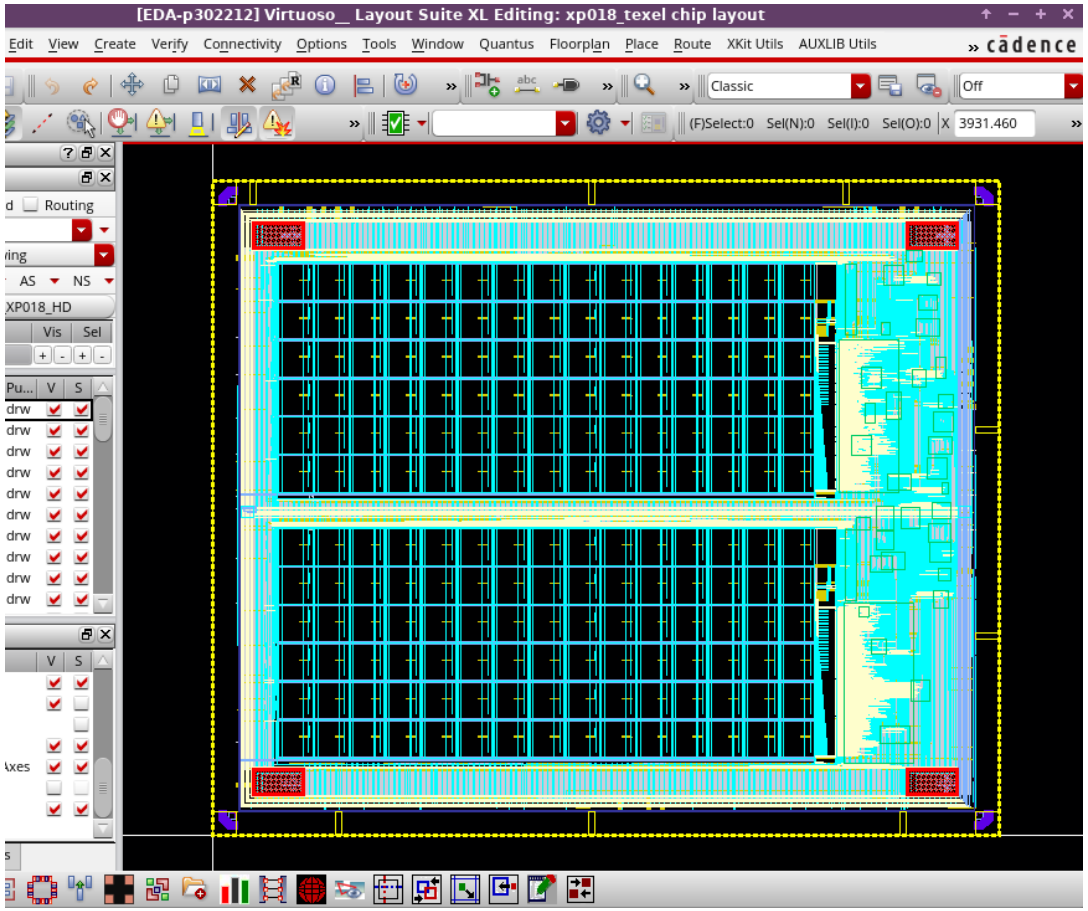

Philipp


Willian


Madison


Hugh


Maxime



Virtuoso Functional Verification


Philipp


Willian

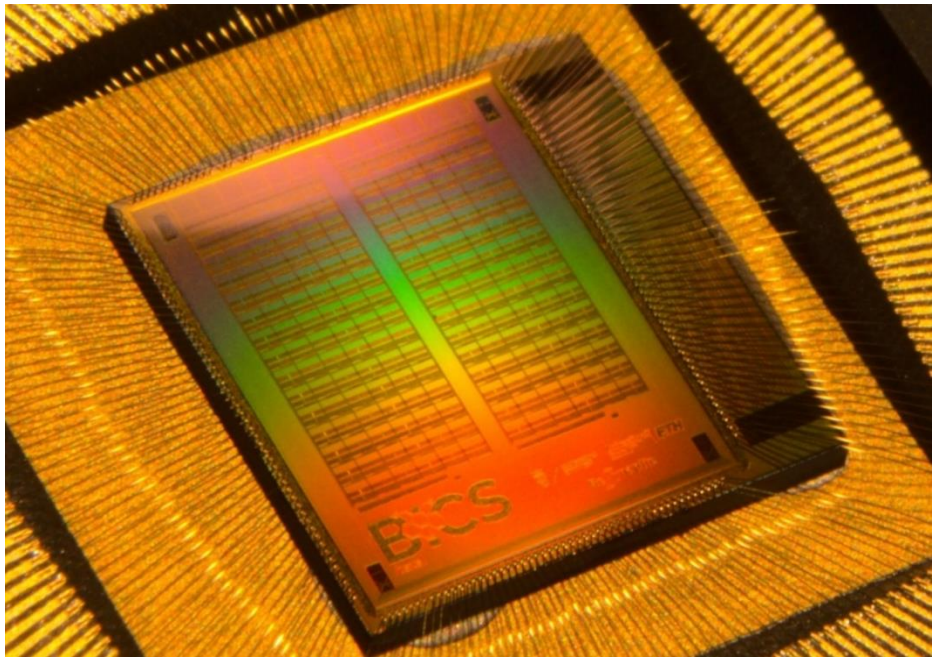

Madison

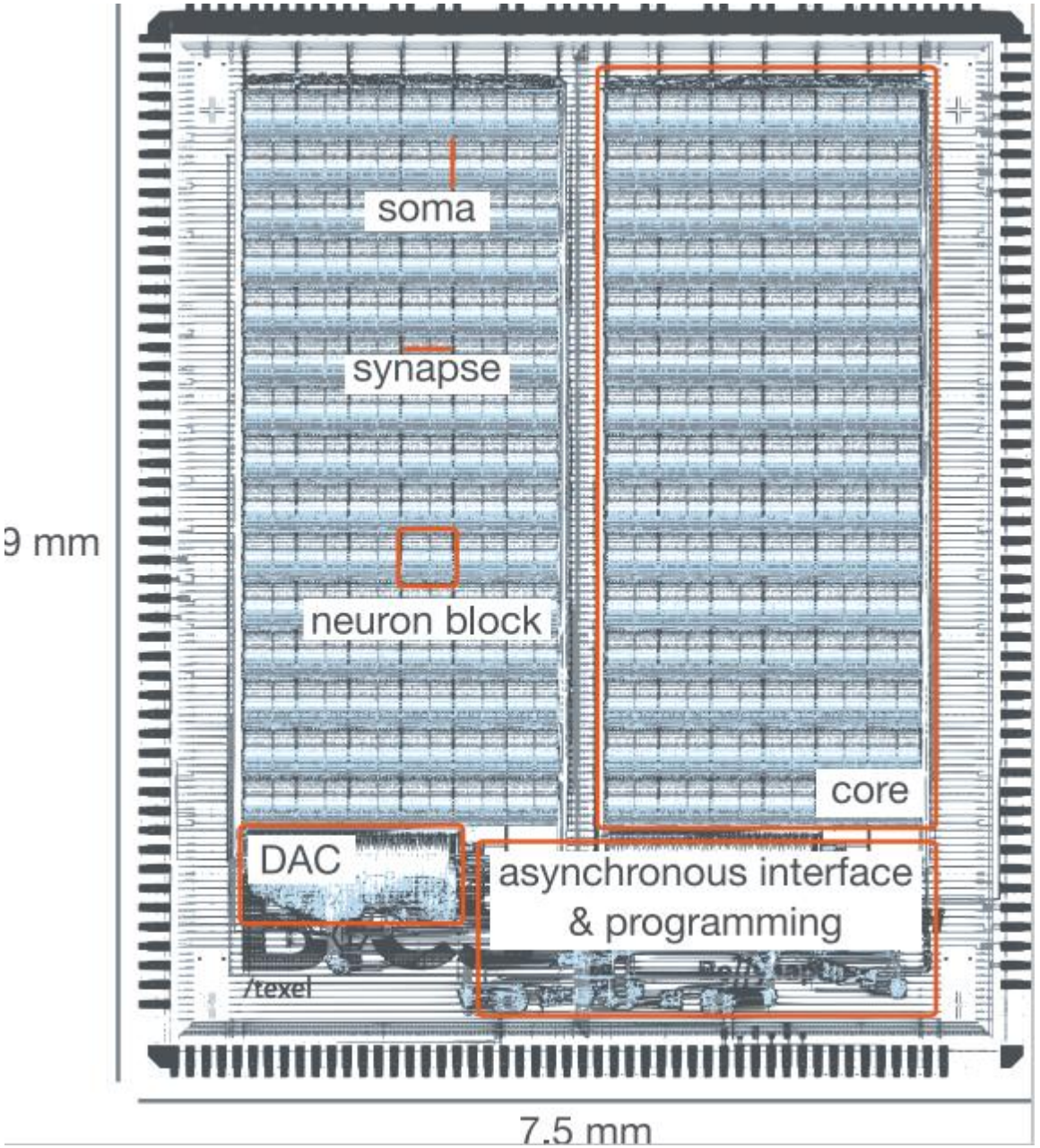

Hugh

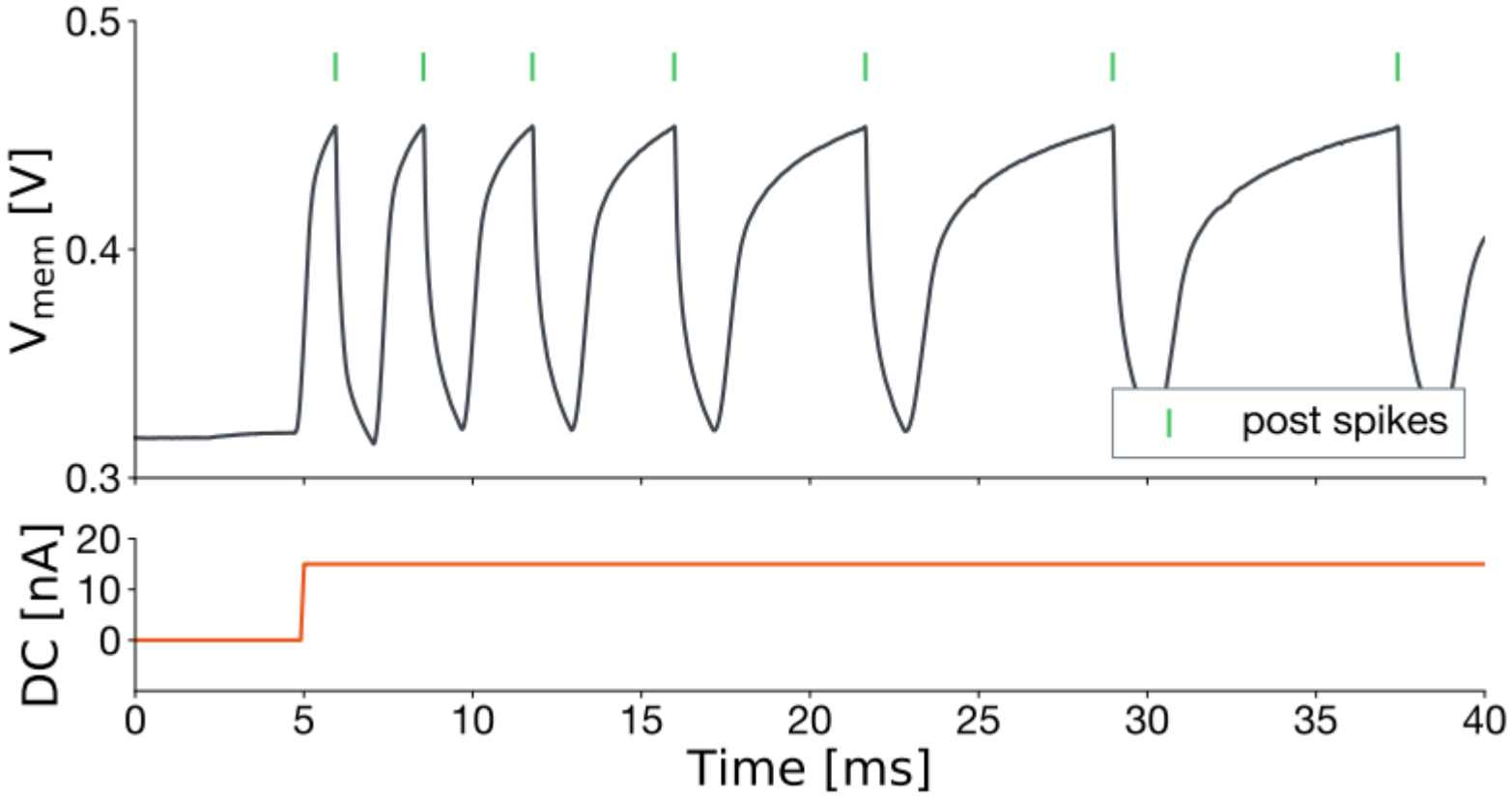

Maxime

- CSV based test-vector-sets
- Digital verilog simulations
- Mixed-signal simulation
- Full Chip SPICE event-in-event-out simulation

The final ASIC






0.5
0.4
0.3
V_{mem} [V]
post spikes
20
10
0
DC [nA]
0
5
10
15
20
25
30
35
40
Time [ms]

git.web.rug.nl/bics
or EDA Vendor Design Share Agreement

Thank you!

BICS IC Design Team


Michele Mastella


Willian Soares Girão


Madison Cotteret


Hugh Greatorex


Maxime Fabre


Philipp Klein


PI: Elisabetta Chicca


Ole Richter

BICS-Group