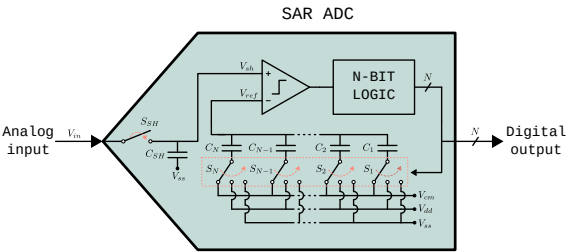
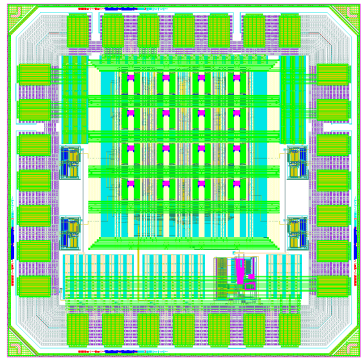
FRIDA Design Review

Kennedy Caisley

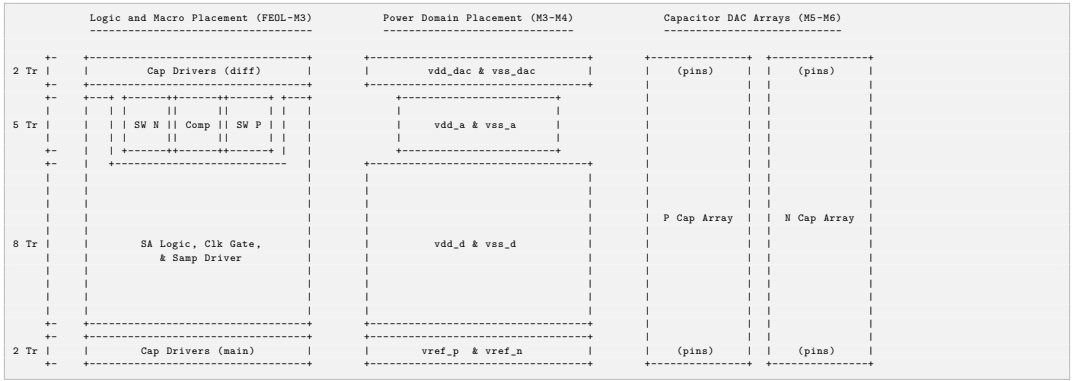
Wednesday, 28 January 2026

FRIDA Chip Overview



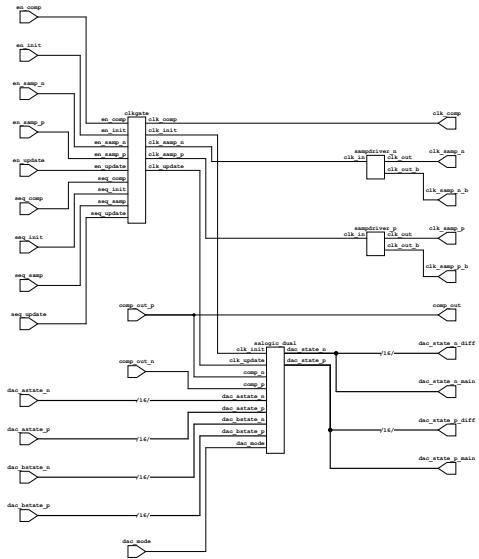


Sketch of ADC Floorplan



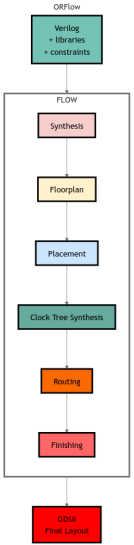
ADC Digital Block Netlist

- **salogic**: SAR sequencing FSM
- **clkgate**: Clock gating for power
- **sampdriver_p/n**: Sampling switch drivers
- Symmetric P/N halves for differential operation
- salogic and clkgate shared between halves



OpenROAD Flow Scripts (ORFS)

- **Yosys**: RTL synthesis
  - Verilog parsing and elaboration
  - Logic optimization (ABC)
  - Technology mapping to standard cells
- **OpenROAD**: Physical design
  - Floorplanning and PDN
  - Placement (global + detailed)
  - Clock tree synthesis
  - Routing (global + detailed)
  - Timing analysis (OpenSTA)
- **KLayout**: Final verification
  - DRC and LVS checks
  - GDS export



Yosys: Step 1 - RTL Synthesis

Input Files

- adc_digital.v (top module)
- clkgate.v, salogic.v, sampdriver.v
- cells_tsmc65.v (cell wrappers)

Synthesis Statistics

- Total cells: 350
- Total area: 893.88 μm^2
- Sequential elements: 49% of area
- 48 flip-flops (DFQD1, EDFD2)
- 5 clock gates (CKLNQD1LVT)

Technology Mapping (ABC)

```
Module          Cells     Area
---------------------------------
adc_digital        38    54.36
  clkgate           6    33.48
  salogic         302   798.84
  sampdriver        2     3.60
---------------------------------
Total             350   893.88
```

Key Cell Types

```
EDFD2LVT (edge DFF)      32   334.08
DFQD1LVT (DFF)           16   109.44
NR2D0LVT (NOR2)          93   133.92
BUFFD0LVT (buffer)       69    99.36
CKLNQD1LVT (clkgate)      5    32.40
```

OpenROAD Step 2.1: Floorplan

- Die and core area: 60μm $\times$ 49μm
- Row height: 2.8μm (17 rows)
- Blockages reserve space for analog macros
- I/O pin and routing grid (0.2μm pitch)

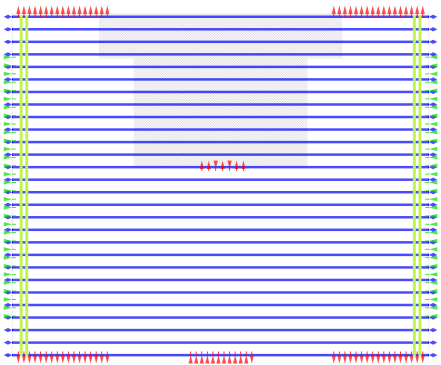
config.mk

```
export DIE_AREA = 0 0 60 49
export CORE_AREA = 0 0 60 49
export CREATE_BLOCKAGES = .../create_blockages.tcl
```

create_blockages.tcl

```
# Reserve area for comparator
create_blockage -region {17.5 27.0 42.5 49}
# Reserve areas for sampling switches
create_blockage -region {12.5 42.6 21 49}
create_blockage -region {39 42.6 47.5 49}
```

OpenROAD Step 2.2: Power Distribution Network

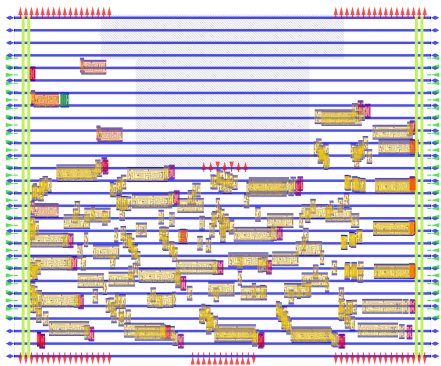


- M4 vertical stripes at edges for vdd_d/vss_d
- M1 horizontal stripes follow cell rows
- M1-M4 connections for power delivery

pdn.tcl

```
add_global_connection -net {vdd_d} \
  -inst_pattern {.*} -pin_pattern VDD -power
add_global_connection -net {vss_d} \
  -inst_pattern {.*} -pin_pattern VSS -ground

define_pdn_grid -name "Core" -pins {M4}

add_pdn_stripe -grid "Core" -layer M4 \
  -width 0.4 -offset 1.3 -nets {vdd_d}
add_pdn_stripe -grid "Core" -layer M4 \
  -width 0.4 -offset 2.1 -nets {vss_d}

add_pdn_stripe -grid "Core" -layer M1 \
  -followpins -width 0.33
add_pdn_connect -grid "Core" -layers {M1 M4}
```

OpenROAD Step 3.1: Global Placement

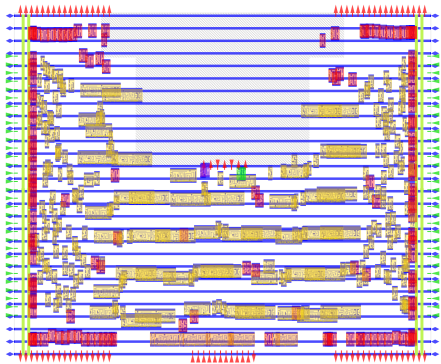


- Initial cell location doesn't consider I/O
- Instead routability-driven placement
- 60% target placement density
- Placement avoids blockages

config.mk

```
# Standard cell placement density
export PLACE_DENSITY = 0.50

# I/O pin layers
export IO_PLACER_H = M3
export IO_PLACER_V = M2
```

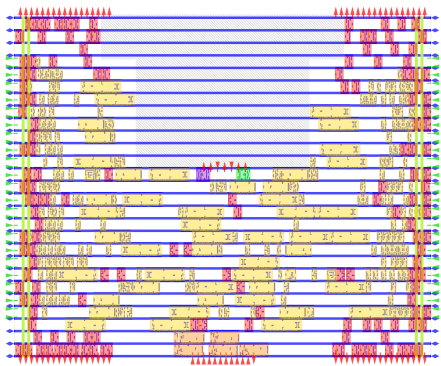
OpenROAD Step 3.2: Global Placement (I/O Aware)



- Cells migrated toward connected IO pins
- 445 movable cells, 457 total instances
- Timing-driven with Nesterov solver
- 335 iterations to converge

Metric	**Value**
Core area	2916 μm^2
Cell area	1102 μm^2
Utilization	49%
Final HPWL	5125 μm
Worst slack	4.49 ns
fmax achievable	1975 MHz

OpenROAD Step 3.3: Detailed Placement



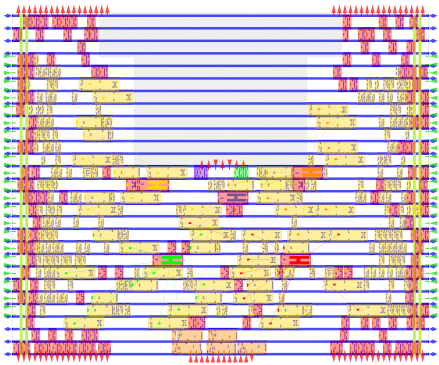
- 424 cells legalized to row sites
- Legalization adds 4% HPWL
- Optimization recovers 9.8% HPWL

Optimization Passes

```
Independent set matching    0.35%
Global swaps                2.36%
Vertical swaps              0.34%
Reordering                  0.96%
Random improvement          3.09%
Cell flipping               3.10%
```

Metric	Value
Final HPWL	4796 μm
Final area	1102 μm^2
Utilization	38%

OpenROAD Step 4.1: Clock Tree Synthesis

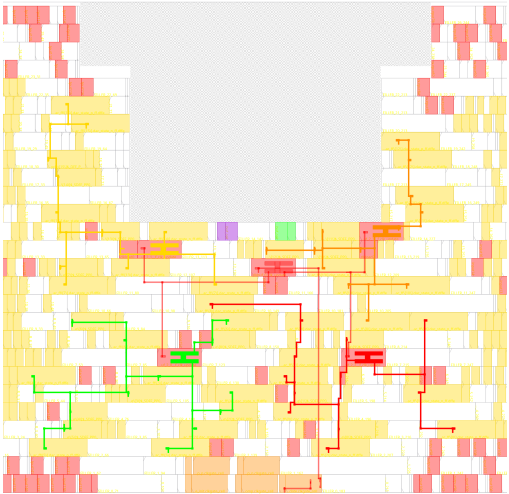


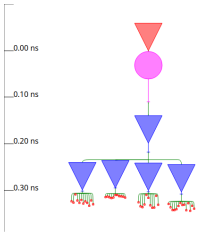
- System clock: 10 MHz (100ns period)
- Samp/comp clock: 200 MHz (5ns period)
- 50ps clock uncertainty
- Balanced tree for minimal skew

constraint.sdc

```
current_design adc_digital

# Create clock for sequencing (200 MHz)
create_clock -name seq_update \
  -period 5 [get_ports seq_update]

# Set clock uncertainties
set_clock_uncertainty 0.05 [all_clocks]
```

OpenROAD Step 4.1: Clock Tree Views

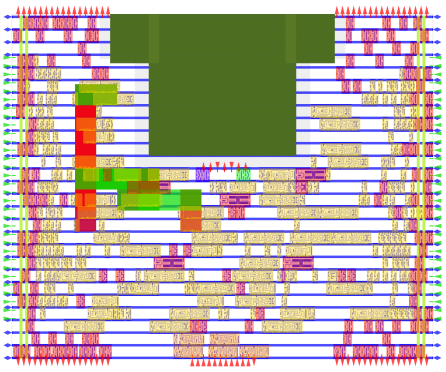




CTS Results:

- 31 BUFFD2LVT cells across 4 clock paths
- Domains unsynchronized within ADC (relies on upstream sequencer)
- Clock power: 41% of total
- Setup skew: 0.07 ns

OpenROAD Step 5.1: Global Routing



- Route guides shown for global routing
- M2-M3 routing layers only
- Routing obstructions for analog areas

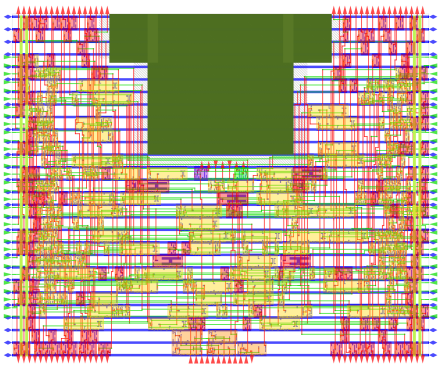
config.mk

```
export MIN_ROUTING_LAYER = M2
export MAX_ROUTING_LAYER = M3
export PRE_GLOBAL_ROUTE_TCL = \
  .../routing_blockages.tcl
```

routing_blockages.tcl

```
# Create obstructions on M1-M4 for analog
odb::dbObstruction_create $block $layer_M1 \
  $comp_llx $comp_lly $comp_urx $comp_ury
odb::dbObstruction_create $block $layer_M2 \
  $comp_llx $comp_lly $comp_urx $comp_ury
```

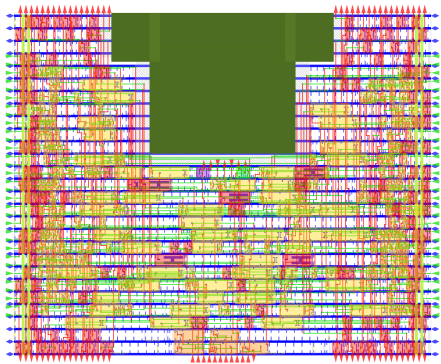
OpenROAD Step 5.2: Detailed Routing



- Final metal routing on M2/M3
- All signals routed to I/O pins
- DRC-clean routing around blockages
- **Issue:** Multicut vias on large devices

Metric	**Value**
Nets routed	506
Total wire length	5717 μm
M2 wire length	3183 μm
M3 wire length	2625 μm
Total vias	2058
Clock skew	0.07 ns
Worst slack	4.37 ns

OpenROAD Step 5.3: Fill Cells

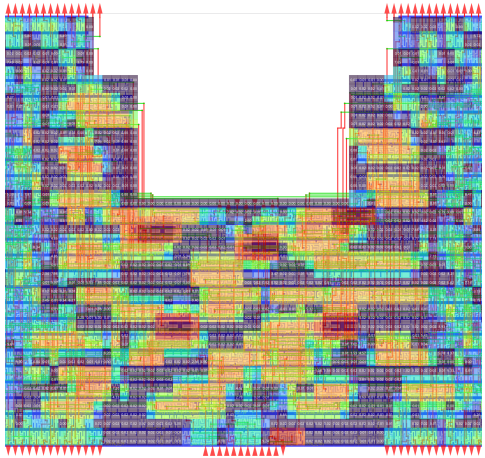


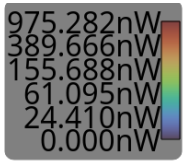
- Decap fill cells added (FEOL)
- Metal fill disabled (BEOL)
- Fills gaps in standard cell rows

config.mk

```
# Only disables metal fill
# FILL_CELLS in routing step still
# adds FEOL decap fill cells
export USE_FILL = 0
```

OpenROAD Step 6.1: Heat Maps



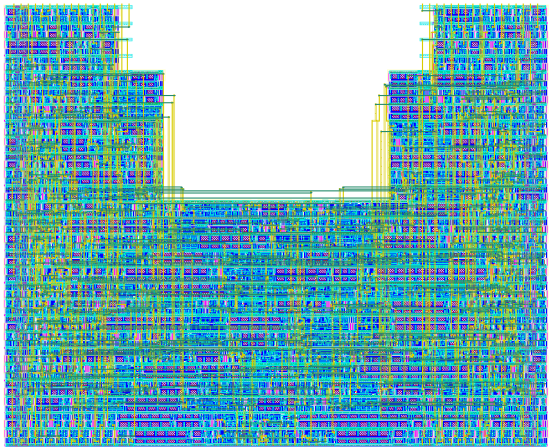


- Spatial visualizations for IR drop, thermal dissipation, placement density, pin density, and power consumption
- DRC and antenna violation views for debugging

OpenROAD GUI

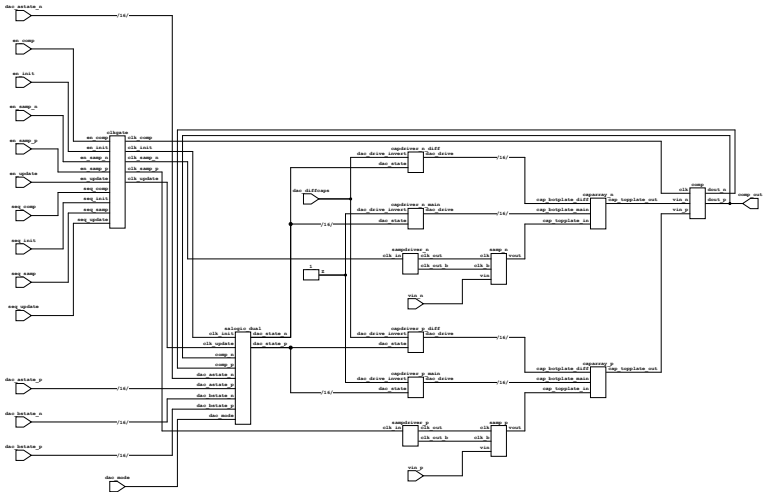
- Interactive GUI, net tracing, hierarchical cell grouping, timing reports, DRC viewer, heatmaps

ADC Digital Block Layout



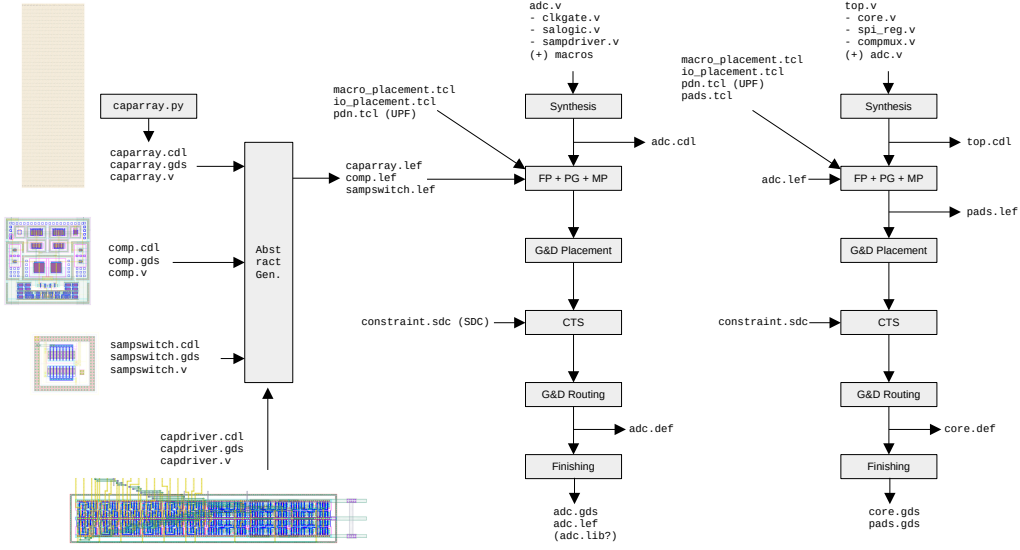
Metric	Value
Dimensions	60×49 μm
Cell count	350
Area	894 μm^2
Flip-flops	48
Clock gates	5
Wire length	5717 μm
Vias	2058
Power	18 μW

ADC Top-Level Assembly



- ADC assembled using hierarchical Verilog as well, with protected routing for sensitive analog nets

ADC Level Integration



CDAC Problem: Mismatch vs Area

- For compact SARs, CDAC area and mismatch push in opposite directions.
- Study target: fringe unit caps (1/2/3 layers), compared across 180/65/28 nm.
- Binary-weighted arrays are evaluated at 4σ (0.999936) mismatch via analytic estimates + Monte Carlo.
- DNL from capacitor mismatch adds to quantization-noise error budget.
- Beyond an overlap/margin inflection point, foreground calibration cannot recover missing codes.

Mismatch scaling used in analysis

$$\sigma_{INL_{max}} \approx \frac{1}{2}\,\sigma_{C_{unit}}\sqrt{2^N}$$

Redundancy overlap (calibration margin)

$$\beta = 2^{N/M}, \quad \Delta_i = \sum_{j=i+1}^{M-1} W_j - W_i$$

calibratable when $\Delta_i > 4\sigma_{\mathrm{equiv},i}$

Noise sanity bound

$$\sigma_n^2 \le \frac{LSB^2}{2 \times 12}$$

CDAC Strategy Stack

Strategy	Primary Benefit	Key Limitation / Tradeoff
1) Calibration	Corrects deterministic static CDAC errors after measurement	In pure binary weighting, calibratability margin is a short leash because it depends on mismatch in the same array being corrected
2) Redundancy (sub-radix, M:N expansion)	Creates overlapping decision ranges; improves tolerance to cap mismatch and comparator/reference decision noise	Does not improve linearity by itself; it creates correction room. Trades extra steps/time and more complex weights
3) Coarse-fine CDAC partition	Improves effective LSB matching without scaling full array area; weakens fixed area-vs-matching coupling	Must still satisfy sampling-capacitance requirement; split method impacts practicality/overhead
4) Switching scheme choice	Monotonic / BSS reduce switching power; CRS/VCM can improve linearity	Linearity-focused switching variants are noted but out of current exploration scope

CDAC Construction Principles:

- Use integer per-bit weights: simpler capacitor implementation and better practical matching.
- Express each bit weight as a sum of binary-scaled terms: decode stays adder-only.
- Keep total weight at a binary-scaled full-scale sum: prevents overflow in digital mapping.

Unit Fringe Capacitor

Capacitance Density:

1 layer: 0.31 $\mathrm{fF}/\mu\mathrm{m}^2$

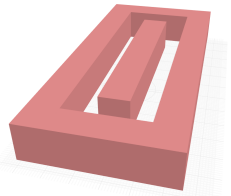
2 layers: 0.62 $\mathrm{fF}/\mu\mathrm{m}^2$

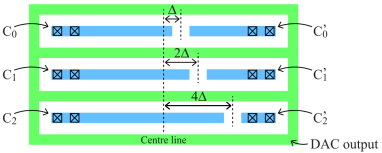
3 layers: 0.93 $\mathrm{fF}/\mu\mathrm{m}^2$

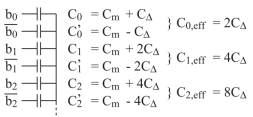
Matching: $\sigma(\Delta C/C) = 0.85\%/\sqrt{C\ [\mathrm{fF}]}$

Per-Bit Mismatch:

Bit	C_{main}	C_{diff}	uncorr	ρ=0.9
15 (768)	619 fF	4.8 fF	0.04%	0.03%
10 (64)	51.6 fF	0.4 fF	0.12%	0.11%
1 (1)	26.4 fF	25.6 fF	7.7%	2.4%

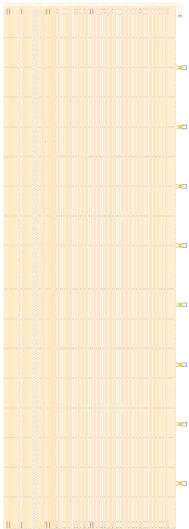






[P. Harpe, ISSCC 2018]

Capacitor Array Layout



Parameter	Value
Dimensions	17×50 μm
C_{tot}	1.88 pF
C_{eff}	1.64 pF
Unit cap (C_u)	0.8 fF
Layers	M5–M6–M7 fringe

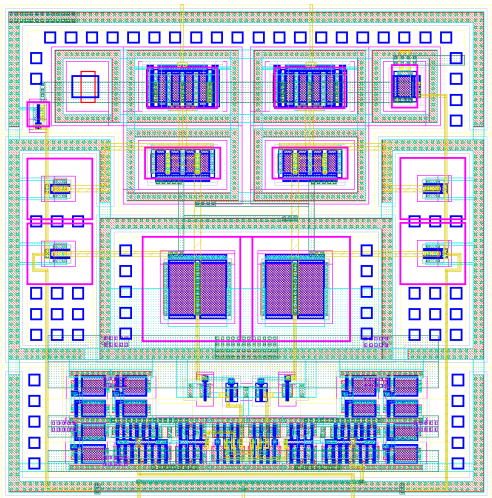
Weights: 768, 512, 320, 192, 96, 64, 32, 24, 12, 10, 5, 4, 4, 2, 1, 1

Harpe Penalty Factor:

$$\sigma_{\text{penalty}} = \sqrt{\frac{C_{\text{tot}}}{C_{\text{eff}}}} = \sqrt{\frac{1880}{1638}} \approx 1.07$$

Only 7% mismatch penalty — MSBs have $C_{\text{diff}} \ll C_{\text{main}}$

Comparator Layout



Parameter	Pre-Extract
Dimensions	18×18 μm
Topology	StrongARM latch
Input noise	∼177 μV
Avg current	∼4 μA
Peak current	∼500 μA

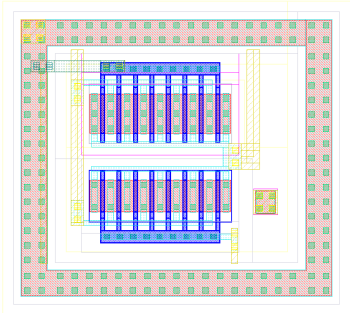
Noise contribution to ENOB:

$$V_{\mathrm{LSB}} = \frac{2V_{\mathrm{ref}}}{2^N} = \frac{1.2\ \mathrm{V}}{4096} \approx 293\ \mu\mathrm{V}$$

$$\text{Comparator noise} \approx \frac{77\ \mu\mathrm{V}}{293\ \mu\mathrm{V}} \approx 0.26\ \mathrm{LSB}$$

- Dynamic (no static power)
- Regenerative latch output

Sampling Switch Layout



Parameter	Value
Dimensions	12×12 μm
Device size	W $\approx$ 3 μm, L_{min}
R_{on}	$\sim$1 kΩ
C_{samp}	1.64 pF

1. Settling Time Requirement

Time constant: $\tau = R_{on} \cdot C = 1.64$ ns

For 12-bit accuracy, settling error < 0.5 LSB:

$$e^{-n} < \frac{0.5}{4096} \quad \Rightarrow \quad n > 9$$

$$\therefore \; t_{sample} > 9 \times 1.64 \text{ ns} \approx \mathbf{15} \text{ ns}$$

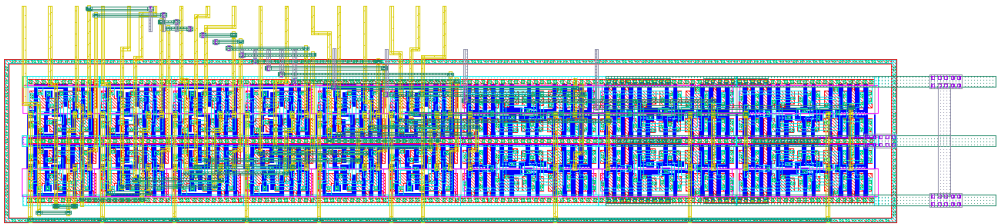
2. kT/C Sampling Noise

$$\sigma_{samp} = \sqrt{\frac{kT}{C}} = \sqrt{\frac{4.14 \times 10^{-21}}{1.64 \times 10^{-12}}} \approx 50 \ \mu\text{V}$$

With LSB $= 293$ μV:

$$\text{Noise} \approx \frac{50 \ \mu\text{V}}{293 \ \mu\text{V/LSB}} \approx 0.17 \text{ LSB} \quad \checkmark$$

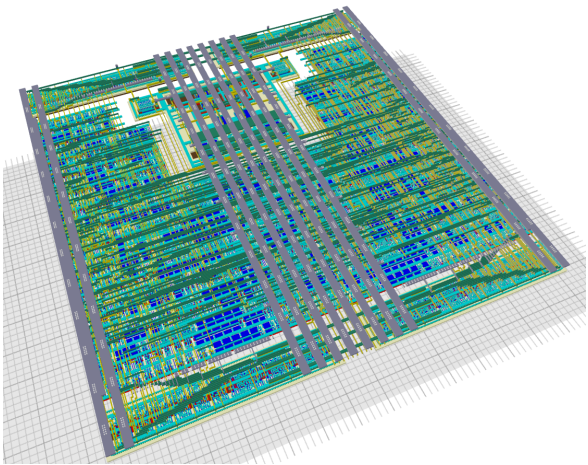
Capacitor Driver Layout



Drivers scaled in width to match corresponding capacitor size:

Bits	Output W (NMOS)	Output W (PMOS)
15–14 (MSB)	3.12 μm	4.16 μm
13–12	1.56 μm	2.08 μm
11–0 (LSBs)	0.78 μm	1.04 μm

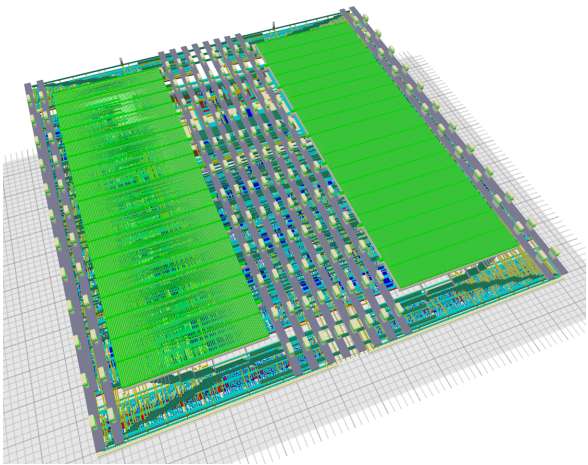
ADC Layout: Lower Metals (3D)



Layer	Usage
M1	Standard cell routing
M2–M3	Signal routing
M4	Power distribution

- Digital logic in center/bottom
- Comparator at top center
- Sampling switches on top sides
- Cap drivers at top/bottom edges

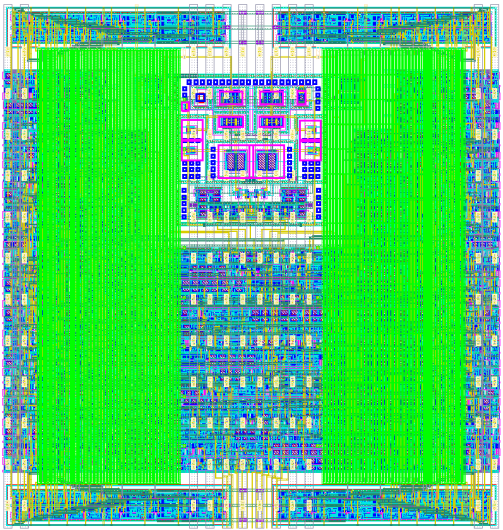
ADC Layout: With Cap Arrays (3D)



Layer	Usage
M5–M6	CDAC fringe caps

- P and N cap arrays on left/right
- Stacked above digital/analog, with shield
- Direct connection to sampling nodes
- Vias from top power grid between

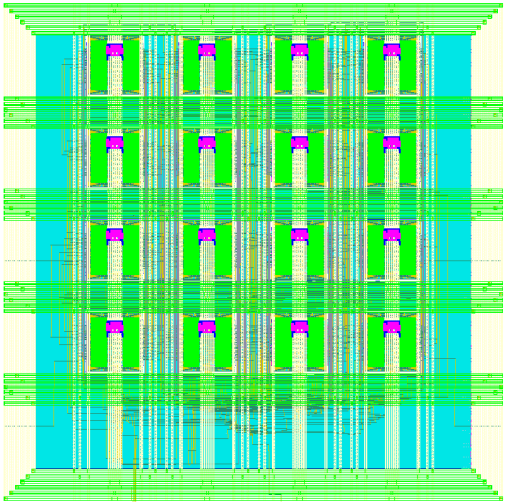
Full ADC Layout



Parameter	Value
Dimensions	60×60 μm
Area	0.0036 mm^2
Resolution	12-bit
Sample rate	10 MS/s
Power	$\sim$200 μW

- Fully differential SAR ADC
- Mixed-signal integration
- Column-parallel compatible

ADC Core Array (4×4)



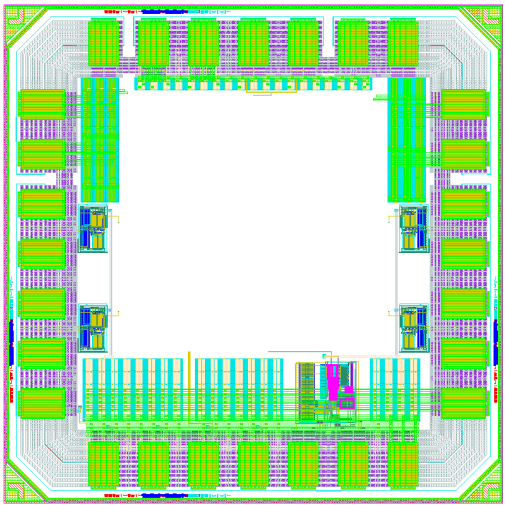
Parameter	Value
Array size	4×4 ADCs
Dimensions	240×240 μm
Power Grid	M8-M9
Total power	∼3.2 mW

- Multiplexed inputs and outputs
- Shared clock distribution

SPI Register (180 bits):

Bits	Function
[179:176]	Mux select (4b)
[175:64]	Per-ADC ctrl (7b × 16)
[63:0]	Shared DAC states (64b)

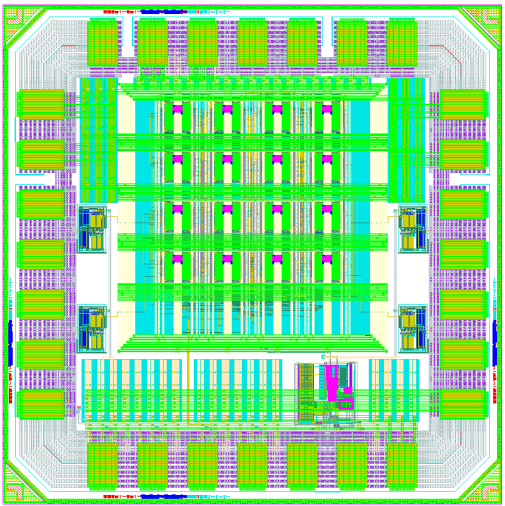
Pad Ring Layout



Pad Type	Count	Purpose
LVDS_RX	4	Sequencing clocks
LVDS_TX	1	Comparator output
CMOS_IO	5	SPI interface + reset
PASSIVE	2	Analog input (diff)
POWER	4	VDD (A, D, IO, DAC)
GROUND	4+1	VSS (A, D, IO, DAC)
Total	**21**	7 per side

- Separate analog/digital supplies
- ESD protection omitted on specific pads
- $\sim$100 pF MOSCAP decoupling/rail
- Die size 1×1 mm

Full Chip Layout (FRIDA 65A)



Pinout (# at top-left, clockwise)

1	vss_a	15	vss_io
2	vdd_dac	16	comp_out_n_pad
3	vss_dac	17	comp_out_p_pad
4	vin_p_pad	18	spi_cs_b_pad
5	vin_n_pad	19	spi_sdo_pad
6	vss_d	20	spi_sdi_pad
7	vss_d	21	spi_sclk_pad
8	vdd_d	22	rst_b_pad
9	vss_d	23	clk_samp_n_pad
10	clk_comp_p_pad	24	clk_samp_p_pad
11	clk_comp_n_pad	25	clk_init_n_pad
12	clk_logic_p_pad	26	clk_init_p_pad
13	clk_logic_n_pad	27	vss_a
14	vdd_io	28	vdd_a

Analog Automation: Sampling Switch Definition

HDL21 generator (`flow/samp/samp.py`):

```
@h.paramclass
class SampParams:
    switch_type = h.Param(dtype=SwitchType, default=SwitchType.NMOS)
    w = h.Param(dtype=int, default=10)
    l = h.Param(dtype=int, default=1)
    vth = h.Param(dtype=MosVth, default=MosVth.LOW)

@h.generator
def Samp(p: SampParams) -> h.Module:
    @h.module
    class Samp:
        din = h.Input(); dout = h.Output()
        clk = h.Input(); clk_b = h.Input()
        vdd = h.Port(); vss = h.Port()

    if p.switch_type in (SwitchType.NMOS, SwitchType.TGATE):
        Samp.mn = h.Mos(tp=MosType.NMOS, vth=p.vth, w=p.w, l=p.l)(
            d=Samp.dout, g=Samp.clk, s=Samp.din, b=Samp.vss
        )
    if p.switch_type in (SwitchType.PMOS, SwitchType.TGATE):
        Samp.mp = h.Mos(tp=MosType.PMOS, vth=p.vth, w=p.w, l=p.l)(
            d=Samp.dout, g=Samp.clk_b, s=Samp.din, b=Samp.vdd
        )
    return Samp
```

Pytest Flags (`python -m pytest flow/`):

```
--flow {netlist,simulate,measure}
--mode {min,max}
--montecarlo {yes,no}
--tech {generic,ihp130,tsmc65,tsmc28,
    tower180}
--simulator {spectre,xyce,ngspice}
--clean {yes,no}
```

Examples:

```
python -m pytest flow/ \
  --flow=netlist --mode=min --tech=
      tsmc65 -v

python -m pytest flow/samp/test_samp.py
    \
  --flow=simulate --simulator=spectre --
      tech=ihp130
```

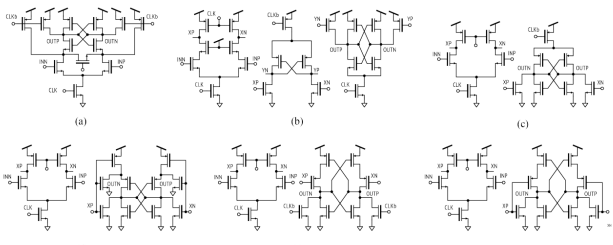
(Run `python -m pytest flow/ --help` for full pytest + project options.)

Analog Automation: Netlist Generation

```
$ python -m pytest flow/samp/test_samp.py \
    --flow=netlist --mode=max --tech=tsmc65 -v

================================================================================
Flow:       netlist
Block:      samp
Tech:       tsmc65
OutDir:     scratch
--------------------------------------------------------------------------------
switch_type  w            l            vth
--------------------------------------------------------------------------------
3x           4x           2x           2x
--------------------------------------------------------------------------------
Result:      24 netlists generated
Wall Time:   8.4ms
Errors:      [none]
```

- Cross-technology sweep: same design across foundries
- Device parameter sweep: automatic design space exploration
- Parallel generation: fast iteration on analog IP blocks
- Dual output: SPICE for simulation, Verilog for integration

Analog Automation: Comparator Topologies



Liu, "Automated and Process-Portable Generation of Data Converters," UCB/EECS-2025-21, 2025

Topology Parameters:

Parameter	Options
`preamp_diffpair`	nmos, pmos
`preamp_bias`	std, dyn
`comp_stages`	1, 2
`latch_pwrgate_ctl`	clk, sig
`latch_pwrgate_node`	ext, int
`latch_rst_ext_ctl`	clk, sig, none
`latch_rst_int_ctl`	clk, sig

- Auto netlist from `blocks/comp.py`
- Sweep device params & corners
- Can also vary topology & tech node
- Share testbench, simulation, and analysis code

Analog Automation: Generated SPICE Netlists

180nm Double-Stage (Tower):

```
.subckt comp in+ in- out+ out- clk clkb vdd vss

M_preamp_diff+ out- in+ tail vss n18lvt W=880n L=180n
M_preamp_diff- out+ in- tail vss n18lvt W=880n L=180n
M_preamp_tail tail clk vss vss n18lvt W=440n L=360n
M_preamp_rst+ out- clk vdd vdd p18lvt W=440n L=180n
M_preamp_rst- out+ clk vdd vdd p18lvt W=440n L=180n
Ma_latch+ latch- latch+ latch_vdd vdd p18lvt W=220n L=180n
Ma_latch- latch+ latch- latch_vdd vdd p18lvt W=220n L=180n
Mb_latch+ latch- latch+ latch_vss vss n18lvt W=220n L=180n
Mb_latch- latch+ latch- latch_vss vss n18lvt W=220n L=180n
M_preamp_to_latch+ latch- out- vss vss n18lvt W=220n L=180n
M_preamp_to_latch- latch+ out+ vss vss n18lvt W=220n L=180n
M_latch_ext_powergate+ latch_vdd clk vdd vdd p18lvt W=220n
M_latch_ext_powergate- latch_vdd clk vdd vdd p18lvt W=220n
M_latch_ext_rst+ latch- clkb latch_vdd vdd p18lvt W=220n
M_latch_ext_rst- latch+ clkb latch_vdd vdd p18lvt W=220n
M_latch_vss_conn+ latch_vss vdd vss vss n18lvt W=220n
M_latch_vss_conn- latch_vss vdd vss vss n18lvt W=220n
M_latch_int_rst+ latch- clkb latch_vss vss n18lvt W=220n
M_latch_int_rst- latch+ clkb latch_vss vss n18lvt W=220n
Ma_latch_out+ out- latch- vdd vdd p18lvt W=220n L=180n
Ma_latch_out- out+ latch+ vdd vdd p18lvt W=220n L=180n
Mb_latch_out+ out- latch- vss vss n18lvt W=220n L=180n
Mb_latch_out- out+ latch+ vss vss n18lvt W=220n L=180n

.ends comp
```

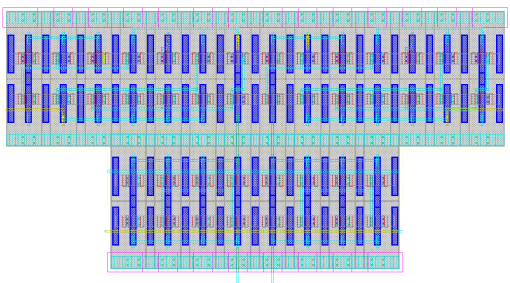
65nm Single-Stage (TSMC):

```
.subckt comp in+ in- out+ out- clk clkb vdd vss

M_preamp_diff+ out- in+ tail vss nch_lvt W=480n L=60n
M_preamp_diff- out+ in- tail vss nch_lvt W=480n L=60n
M_preamp_tail tail clk vss vss nch_lvt W=240n L=120n
M_preamp_rst+ out- clk vdd vdd pch_lvt W=240n L=60n
M_preamp_rst- out+ clk vdd vdd pch_lvt W=240n L=60n
Ma_latch+ out- out+ vdd vdd pch_lvt W=120n L=60n
Ma_latch- out+ out- vdd vdd pch_lvt W=120n L=60n
Mb_latch+ out- out+ vss vss nch_lvt W=120n L=60n
Mb_latch- out+ out- vss vss nch_lvt W=120n L=60n
M_latch_int_rst+ out- clkb vdd vdd pch_lvt W=120n L=60n
M_latch_int_rst- out+ clkb vdd vdd pch_lvt W=120n L=60n

.ends comp
```

Auto-Generated Verilog:

```
module comp (in+, in-, out+, out-, clk, clkb, vdd, vss);
    input in+, in-;
    output out+, out-;
    input clk, clkb;
    inout vdd, vss;
    wire tail;
    nch_lvt M_preamp_diff+ (.d(out-), .g(in+), .s(tail));
    nch_lvt M_preamp_diff- (.d(out+), .g(in-), .s(tail));
    // ... (structural Verilog for LVS)
endmodule
```

Analog Automation: Layout with OpenROAD



C. Wulff et al., "A Compiled 9-bit 20-MS/s 3.5-fJ/conv.-step SAR ADC in 28-nm FDSOI for BLE Receivers," IEEE JSSC, 2017

1. **Parse SPICE → Verilog**
   Structural netlist with transistor primitives
2. **Map to primitive cells**
   NMOS/PMOS → stdcell-like wrapper LEFs
3. **Partition differential halves**
   Split circuit at symmetry axis (P vs N)
4. **Explore with net weights**
   Vary placement via `specifyNetWeight` to generate layout candidates
5. **Reassemble with symmetry**
   Mirror placement coordinates across axis; lock cells with `LOCKED` status
6. **Route with guides**
   Extract guides, mirror for paired nets

Analog Automation: CDAC Weight Strategies

Five strategies in `blocks/cdac.py`:

Strategy	Description	Radix
`rdx2`	Standard binary	2.0
`subrdx2`	Sub-radix-2 rounded	<2.0
`subrdx2lim`	Sub-radix-2 w/ floor	<2.0
`subrdx2rdst`	MSB redistribution	~ 2.0
`rdx2rpt`	Binary w/ repeated	2.0

Weight calculation:

$$\beta = 2^{N/M} \qquad W_i = \max(1, \lfloor \beta^{M-1-i} \rfloor)$$

Coarse-Fine Partitioning (threshold $T = 64$):

$$W = q \cdot T + r \quad \text{where } q = \lfloor W/T \rfloor$$

Difference Capacitor Split:

- Coarse: $C_{\text{main}} = T \cdot C_u$, $m = q$
- Coarse diff: $C_{\text{diff}} = 1 \cdot C_u$, $m = q$
- Fine: $C_{\text{main}} = (T + 1 + r)$, $C_{\text{diff}} = (T + 1 - r)$

Effective: $C_{\text{eff}} = C_{\text{main}} - C_{\text{diff}}$

ADC Target Specifications

Parameter	DCD v1	CoRDIA	M	H	F (target)
Resolution	8-bit	10-bit	8-bit	10-bit	12-bit
ENOB	8.3	8.8	8.0	9.5?	11.0?
Conversion rate	6.25 MHz	2.5 MHz	4.5 MHz	10 MHz	10 MHz
ADC dimensions	40×55 μm	80×330 μm	60×800 μm	15×100 μm	50×50 μm
ADC area	0.002 mm^2	0.026 mm^2	0.048 mm^2	0.0015 mm^2	0.0025 mm^2
Power per ADC	960 μW	30 μW	700 μW	100 μW	200 μW?
FOM$_{\mathrm{csa}}$ (Hz/μm^2)	3125	95	105	5000	5000
FOM$_{\mathrm{wal}}$ (fJ/conv-step)	487	26	608	14	10

- **F** = FRIDA target: 12-bit, 10 MHz, 50×50 μm footprint
- FOM$_{\mathrm{csa}}$ = conversion rate / area; FOM$_{\mathrm{wal}}$ = energy per conversion step

Comparator Constraints in `@h.generator`

Style aligned with AlignHdl21/Fritchman thesis:

```
@h.generator
def Comp(p: CompParams) -> h.Module:
    mod = _build_comp_core(p)  # abbreviated comparator construction
    pnr = ah.PnrInput(
        module=mod,
        constraints=[
            ah.SymmetricBlocks(pairs=[("mdiff_p", "mdiff_n")]),
            ah.RouteNdr(net="tail", width_mult=2, spacing_mult=2),
        ],
    )
    pkg = h.elab(mod)
    vlsir.netlist(pkg, fmt="verilog")  # emitted for OpenROAD ingest
    return mod
```

- Constraints are declared next to schematic generation.
- Netlist export remains a compact handoff to the Hdl21 → VLSIR → OpenROAD flow.

ODB Direct vs DEF Import (Hdl21 → VLSIR → OpenROAD)

Approach	**Strengths**	**Tradeoffs**
Direct ODB generation	Full OpenDB object control (groups, regions, NDR, blockages, pre-routes)	Tighter OpenROAD/OpenDB coupling; harder portability; higher implementation burden in VLSIR bridge
DEF file generation + import	Standard interchange, easier debug/diff, cleaner interoperability across tools	DEF does not capture all analog constraints; extra sidecar commands are needed

Analog Constraints vs OpenROAD API?

Constraint Idea	Status	Endpoint	Notes

IO side / region / grouping	Supported	`set_io_pin_constraint`
		`exclude_io_pin_region`
		`place_pin`
		`place_pins`
			Includes mirrored IO-pair support for boundary placement

Block boundary / utilization / aspect ratio	Supported	`initialize_floorplan`	Fixed die/core area and utilization/aspect-ratio modes
Fixed instance placement	Supported	`place_inst`	Exact location and orientation available
Device symmetry constraints	Missing	N/A	No first-class pairwise analog symmetry intent
Net width/spacing overrides	Supported	`create_ndr` `assign_ndr`	Good fit for high-current or low-resistance nets
Shielded routing	Internal	`dbSWire::create(net, dbWireType::SHIELD, shield_net)`	Creates SHIELD special-wire objects in OpenDB; no high-level GRT/DRT auto-shield command

Route prioritization	Partial	`global_route`
		`-critical_nets_percentage`
		`set_nets_to_route`
			Timing/subset prioritization exists; generic net-weight flow is limited

Differential pair / matched routing	Missing	N/A	Requires external orchestration or post-processing

Other Outstanding Issues

- No multi-cut via policy support in the global-router stage.
- Power-grid generation is primarily ring/stripe oriented, which often doesn't work well for analog.
- LEF abstract generation and checking (soon?).
- Macros constrained to BEOL-only still appear to induce FEOL placement blockages (i.e. COVER type).
- Guard-ring structure generation and multi-voltage-domain handling unsupported at block level.
- Symmetry and differential-route intent would be to first-class OpenROAD constraints.
- Constraint intent can fragment across DEF + Tcl/Python sidecars.

Capacitor Technology Summary

Type	Density (fF/μm^2)	Key Notes
MOM / RTMOM	0.31 (1L), 0.62 (2L), 0.93 (3L)	Interdigitated multi-metal fingers; best matching; low voltage dependence; can stack above device caps. Matching note: $\sigma(\Delta C/C) \approx 0.85\% \times \sqrt{C[\mathrm{fF}]}$.
MIM / UHD MIM	1.7, 2.8, up to 5.6 (stacked)	Special dielectric between plates (e.g. Si_3N_4, Ta_2O_5); higher density; can consume a top metal option.
MOSCAP / Varactor	7–11	Poly-over-well capacitor with voltage-dependent capacitance; useful when tuning is desired (common RF use).
POD / Accumulation	$\sim$5	Poly-over-diffusion structure in accumulation mode; gate-to-bulk capacitance with source/drain tied.
POP (Poly-over-Poly)	Lower than above	Simple parallel-plate poly structure; requires poly2 availability; suitable for larger values when area is available.
NIC (N-implant cap)	Process-specific	Tower-process option; alternative capacitor primitive depending on PDK support.