Strong ARM Latch - Workshop

Leibniz Institute for high performance microelectronics

Krzysztof Herman

Analog Academy

Contents

Contents

Main objectives

- Provide an analog design flow based on a real example.
- Showcase the capabilities and limitations of open source tools.
- Familiarize the participants with the SG13G2 and CMOS5L technologies of IHP.

Workshop Scope and Constraints

Technology Boundary

- Target both **SG13G2** and **CMOS5L**.
- Keep the implementation portable across both.
- Use `Metal1--Metal3` only.

Views to Produce

- StrongARM latch schematic.
- Simulation benches for **DC**, **transient**, and **AC** analysis.
- Layout and verification-ready deliverables.

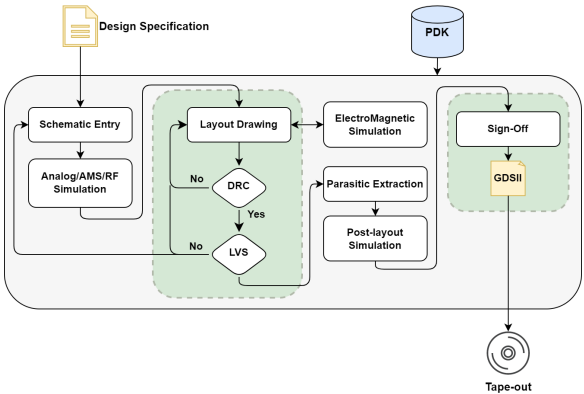
PDK Elements

- Low-voltage MOSFETs only.
- Substrate and well contact tap rings: `ptap1`, `ntap1`.
- substrate and well guard rings `dantenna`, `dpantenna`.

Design Focus

Goal: a compact, symmetric comparator core taken from schematic capture to verification with open source tools.

Open Source Tool Chain



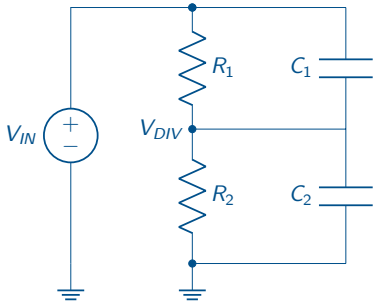
Flow: schematic → simulation → layout → verification → iterate.

Required Tools

- **xschem**: capture schematics and testbenches.
- **ngspice**: run DC, transient, and AC analysis.
- **klayout**: inspect GDS and run physical verification.
- **magic**: support layout editing and extraction.
- **cace**: automate corner analysis.

Contents

Basic xschem example → Capacitive-Resistive Divider



DcOP

```
.control
  op
.endc
```

Transient

```
.control
  tran 1u 100u
  plot Vdiv
.endc
```

Xschem Shortcuts for Fast Capture

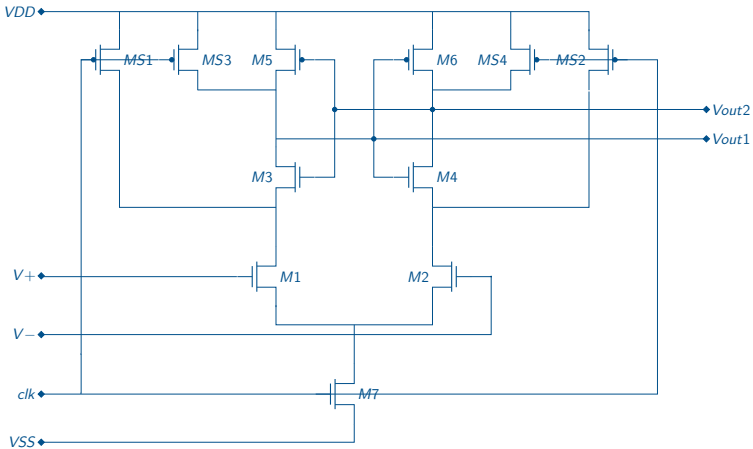
Key	Action
`q`	Open the properties window of an object.
`m`	Move the selected object.
`c`	Copy the selected object.
`Alt+R`	Rotate the selected symbol.
`Alt+F`	Flip the selected symbol.
`w`	Start wiring.
`Delete`	Remove the selected object.

Element	Function
`gnd`	ground
`vsource`	voltage source
`capa`	capacitor
`res`	resistor
`lab_wire`	wire label

Netlist inspection

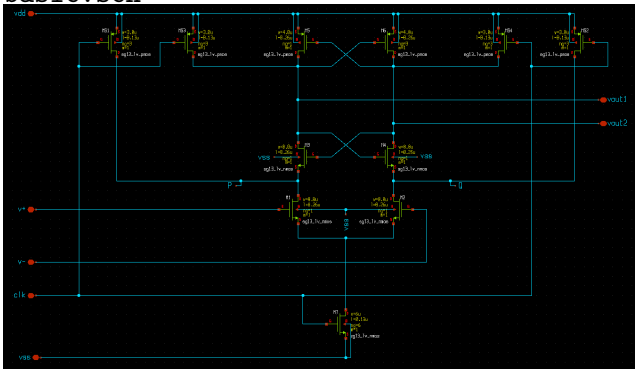
`Shift+A` → netlist

Circuit of the day: StrongARM Latch



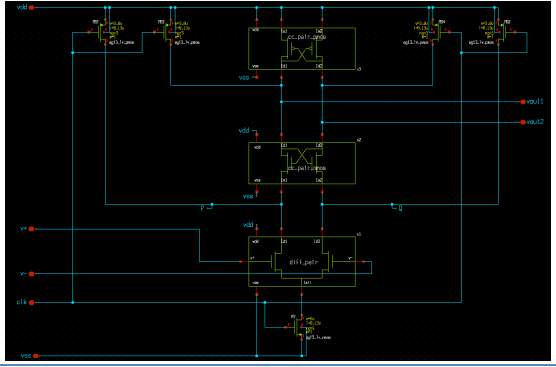
Basic architectural view

```
cd \$HOME\designs\IHP__CMP7345\CMP7345-main\string\schematic\string\xschem\
xschem CMP7345-basic.sch
```



Extended layout compatible view

```
cd \$HOME\designs\IHP__CMP7345\CMP7345-main\string\schematic\string\xschem\
xschem CMP7345-final.sch
```



Contents

Simulation Views

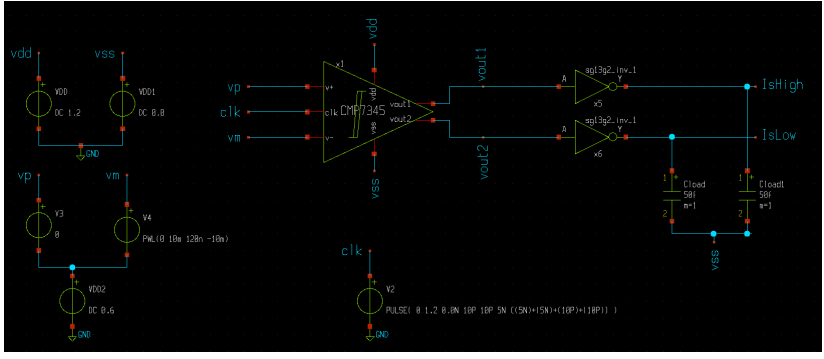
View	Typical Setup	What We Learn
DC	Sweep the input or common-mode level at a fixed clock phase.	Operating point, trip point, input range, device headroom, and the bias behavior of the divider.
Transient	Pulse `CLK` and apply a small input difference near the decision boundary.	Regeneration time, output polarity, metastability tendency, kickback, and clock-to-output delay.
AC	Linearize the divider or another biased front-end sub-block around its operating point.	Bandwidth, attenuation, and pole-zero placement before the latch enters large-signal regeneration.

Important Modeling Note

A StrongARM latch is strongly nonlinear and time-variant during evaluation, so AC analysis is meaningful for the biased linear sub-blocks, not for the full regenerative decision event.

Simulation in transient domain

```
cd $HOME$\designs\IHP__CMP7345\CMP7345-main\testbenches\tran\xschem
xschem CMP7345_tb.sch
```



PDK elements inclusion

```
.lib cornerMOSlv.lib mos_tt
.lib cornerDIO.lib dio_tt
.lib cornerRES.lib res_typ
.INCLUDE sg13g2_stdcell.spice
```

What '.lib' Does

- Loads a model library file and selects one named section from it.
- `mos_tt` picks the typical MOS corner.
- `dio_tt` picks the typical diode corner.
- `res_typ` picks the typical resistor corner.

What '.INCLUDE' Does

- Inserts the full contents of another SPICE file.
- Here it brings in `sg13g2_stdcell.spice`.
- That is needed because the testbench uses SG13G2 standard-cell inverters at the outputs.

Control Block: Setup Commands

```
.options cshunt=20f
.options rseries=1
.control
save all
shell rm tran.raw
```

Numerical stabilization and file operation

- `cshunt=20f` adds a tiny capacitance from each node to ground.
- `rseries=1` inserts a small series resistance to reduce idealized algebraic loops.
- `save all` preserves all voltages and currents for later debug and plotting.
- `shell rm tran.raw` removes an old raw database before writing a fresh one.

Control Block: Running the Transient

```
tran 10p 120n
```

TRAN command syntax

- 10p is the waveform output step used to sample the results.
- 120n is the total simulated duration.

Why the transient analysis is crucial

- A StrongARM latch is evaluated on clock edges, so time-domain simulation is the main way to observe its behavior.
- The small time step is needed to resolve fast regeneration and the buffer output transitions.
- The full time span allows several cycles so measurement can ignore startup artifacts.

Control Block: Measurement and Output

```
meas tran tdly TRIG v(clk) VAL=0.6 RISE=11 \
  TARG v(IsHigh) VAL=0.6 RISE=5
write tran.raw
```

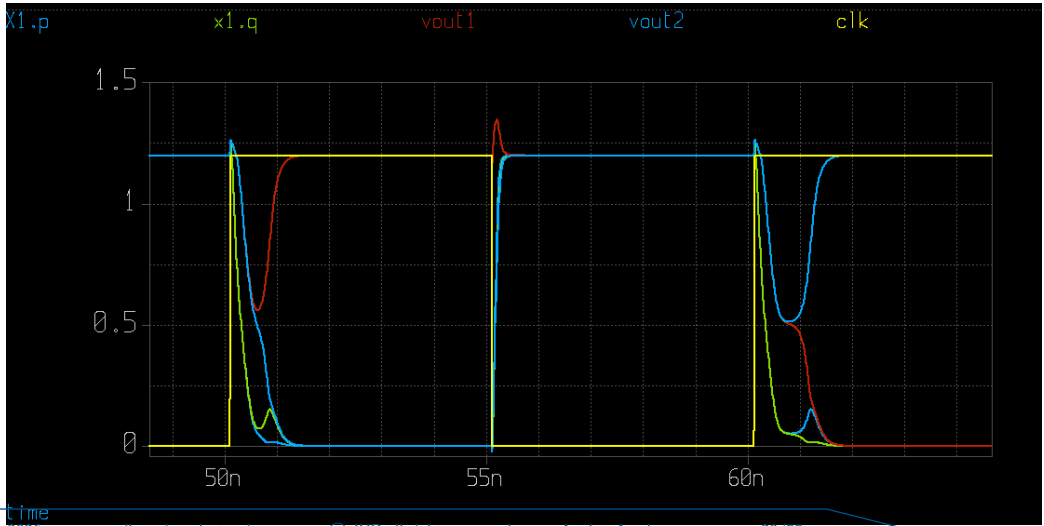
Delay Measurement

- `meas tran tdly` defines a transient measurement named `tdly`.
- The trigger event is the 11th rising crossing of `clk` through 0.6 V.
- The target event is the 5th rising crossing of `IsHigh` through 0.6 V.

Why Use Counted Crossings

- Crossing counters let us skip initial startup cycles.
- This makes the measurement more repeatable when the first few edges are not yet representative.

Transient Result



Regeneration Phase and the Role of P and Q

Regeneration Sequence

- P and Q start precharged near V_{DD}.
- After clk rises, the input pair discharges them with a small differential imbalance.
- This imbalance is the latch's first analog decision.
- Once large enough, it activates the regenerative devices and the outputs separate quickly.

How to Read the Waveform

The node that departs first between x1.p and x1.q indicates which side wins. If P and Q are not well equalized during reset, the next cycle can inherit dynamic offset.

Mismatch - theory

General CMOS Mismatch Model

- Random mismatch is usually described by Pelgrom-type relations.
- Threshold mismatch scales approximately as $\sigma(\Delta V_{TH}) \approx A_{VTH}/\sqrt{WL}$.
- Current-factor mismatch follows a similar law: $\sigma(\Delta\beta/\beta) \approx A_{\beta}/\sqrt{WL}$.
- Larger device area reduces random mismatch, but systematic mismatch is still set by layout symmetry and environment.

130 nm Rule-of-Thumb Values

- For a 130 nm analog CMOS node, a practical order of magnitude is $A_{VTH} \approx 2$ to $6\ \mathrm{mV} \cdot \mu\mathrm{m}$.
- A common design estimate for current-factor mismatch is $A_{\beta} \approx 1\%$ to $3\% \cdot \mu\mathrm{m}$.
- Minimum-length devices can therefore contribute several millivolts of offset.
- Differential pairs usually deserve the largest area.

Main Physical Sources

- local dopant fluctuation
- line-edge and gate-length variation
- oxide/thickness variation
- stress, well proximity, and temperature gradients

Comparator Implication

In a StrongARM latch, mismatch first appears as a small internal imbalance and is then amplified by positive feedback. Near zero differential input, mismatch can therefore decide the output polarity.

Mismatch in Strong ARM Latch circuit

What prof. Razavi says

- Input-pair mismatch in `M1/M2` dominates input-referred offset.
- `M3/M4` and `M5/M6` contribute less because gain develops before full regeneration.
- The article estimates these later contributions by injecting threshold mismatch and referring the error back to the input.

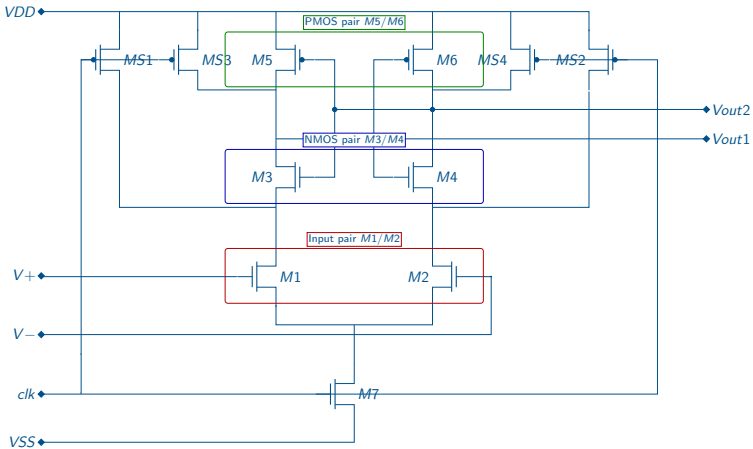
Why is statisticall analysis needed

- One transient run shows only one random mismatch realization.
- Monte Carlo repeats the same testbench over many realizations.
- The result is a distribution of offset, delay, and decision failures near zero input.

Practical Interpretation

For very small differential inputs, mismatch can determine which side wins. We therefore care about the distribution of delay and decisions, not only the nominal case.

Mismatch-Sensitive Pairs



Mismatch Enablement in Xschem

Mismatch Enable Flag

- Each MOS instance can carry an `mm_ok` parameter.
- If `mm_ok=1`, that device is included in mismatch analysis.
- If `mm_ok=0`, mismatch is not injected for that instance.

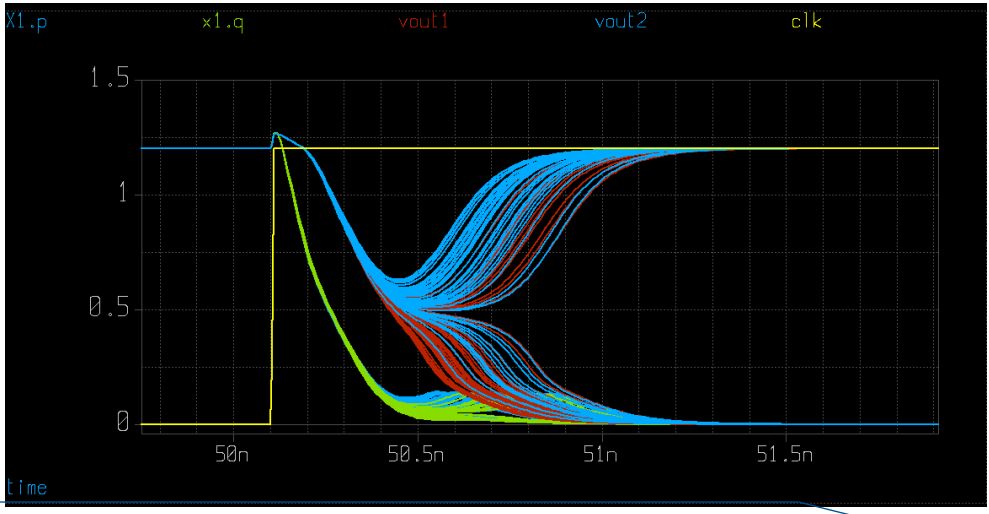
Why This Matters Here

- The highlighted pairs `M1/M2`, `M3/M4`, and `M5/M6` are the main devices of interest for offset spread.
- This lets Monte Carlo focus on comparator-core mismatch rather than every transistor in the testbench.

Mandatory Library Section

Use the mismatch-enabled model section such as `mos_tt_mismatch` in the `.lib` statement. Without the mismatch section, Monte Carlo runs only repeat the nominal corner and no random device spread is produced.

Mismatch Simulation Result



Process variation - theory

General Process Variation Model

- Process variation describes fabrication-induced shifts of device parameters.
- It is usually split into global, die-to-die variation and local, within-die variation.
- Key affected parameters are V_{TH}, L_{eff}, W_{eff}, oxide thickness, mobility, and resistance.
- Corners such as TT, FF, SS, FS, and SF approximate worst-case combinations.

Main Physical Sources

- lithography and etch bias
- implant dose and annealing variation
- oxide thickness variation
- CMP, metal thickness, and contact resistance variation

130 nm Rule-of-Thumb Effects

- Threshold voltage may shift by tens of millivolts across process corners.
- Drive current can vary significantly between slow and fast corners.
- RC delay changes due to transistor strength and interconnect resistance/capacitance shifts.
- Always verify timing and offset over PVT: process, voltage, and temperature.

Comparator Implication

In a StrongARM latch, process variation changes input-pair strength, latch regeneration speed, and clock-to-output delay. Across corners, this can shift offset, delay, and minimum resolvable input voltage.

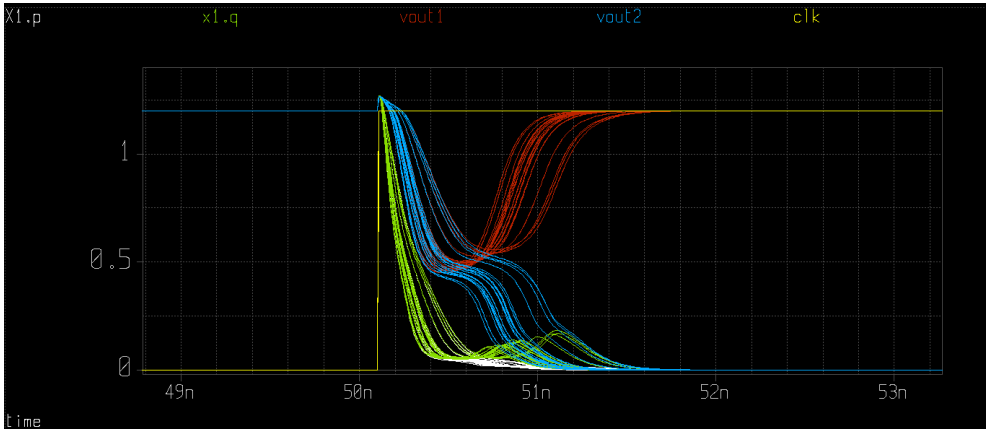
Process Variation Enablement in Xschem

```
cd $HOME$\designs\IHP__CMP7345\CMP7345-main\testbenches\tran\xschem
xschem CMP7345_process_tb.sch
```

Mandatory Library Section

Use the `stat` model section such as `mos_stat` in the `.lib` statement. Without the `stat` section, Monte Carlo runs only repeat the nominal corner and no random device spread is produced.

Process Variation Simulation Result



Kickback Noise - theory

What Kickback Noise Is

- Kickback noise is the disturbance pushed from internal comparator nodes back to the input when the clock switches.
- In a StrongARM latch, the regenerative nodes move quickly and capacitively couple charge into the input pair.
- The effect is strongest around the sampling-to-regeneration transition.
- It appears as an input-dependent transient error rather than a static offset.

Why It Matters

- It can corrupt the signal source or preceding sample-and-hold stage.
- It is often worse for small input signals because the disturbance can be comparable to the decision threshold.
- Faster regeneration usually improves speed but can increase kickback amplitude.
- Layout parasitics and source impedance strongly affect the observed waveform.

Main Physical Paths

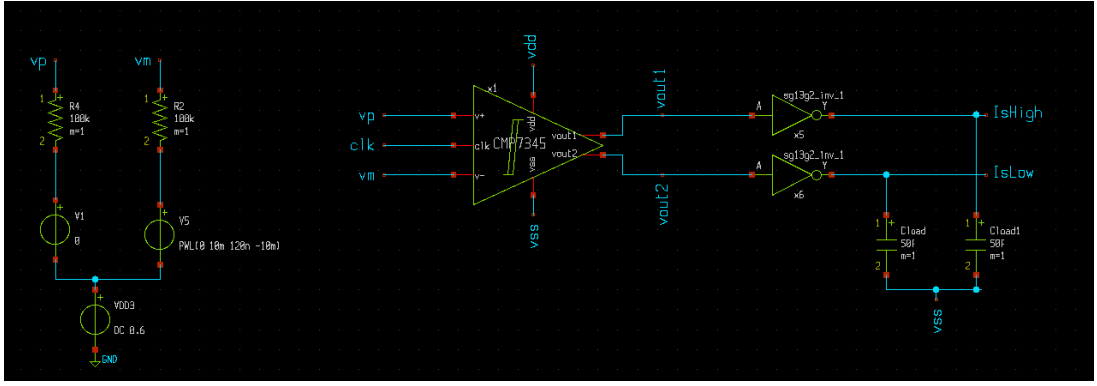
- gate-drain overlap capacitance of the input pair
- drain-bulk and interconnect parasitics
- clock feedthrough through switching devices
- charge sharing between internal and sampled nodes

Comparator Implication

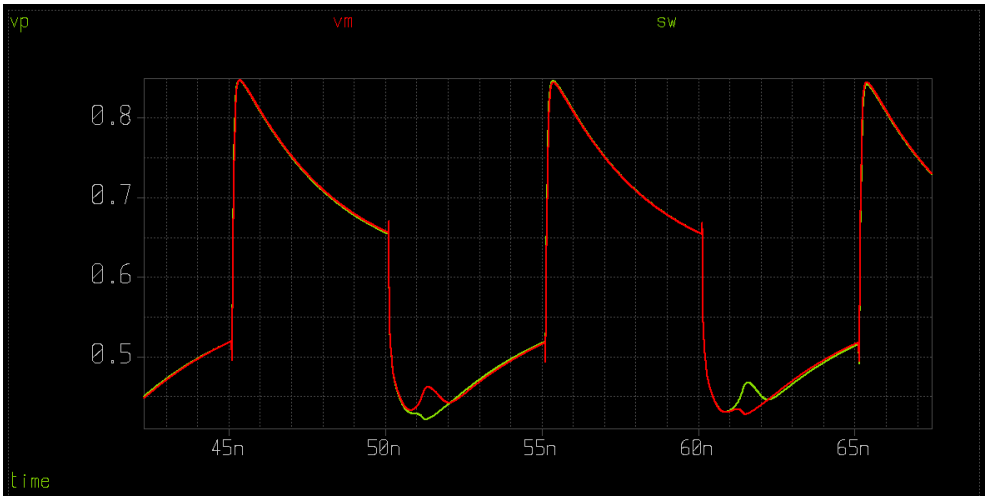
A low-offset comparator can still fail system-level accuracy if its internal switching injects too much charge back into the input network. Kickback must therefore be checked together with delay and offset.

Kickback Noise in Xschem

```
cd $HOME$\designs\IHP__CMP7345\CMP7345-main\testbenches\noise‘\xschem
xschem CMP7345_kickback_tb.sch
```



Kickback Noise Simulation Result



Contents

Klayout features

Main klayout features

- PyCell support - drag and drop.
- Hierarchical layout support.
- KLayout productivity suite.
- Advanced editing options.
- Scripting interfaces for python and ruby
- DRC and LVS support
- 2.5D Viewer

```
klayout -e
```

Layout Implementation Rules

Portable Cell Recipe

- Use only low-voltage MOS devices common to both target processes.
- Tie wells and substrate locally with `ptap1` and `ntap1`.
- Add `dantenna`/`dpantenna` for addtitional protectiona.
- Keep routing within `Metal1--Metal3`.

Comparator Matching Priorities

- Mirror the left and right device placement.
- Use common centroid device distribution and dummies.
- Equalize parasitics on the two internal regenerative nodes.
- Keep the `CLK` path compact to reduce skew.
- Plan for extraction, DRC, and LVS from the start.

Klayout key bindings for fast editing

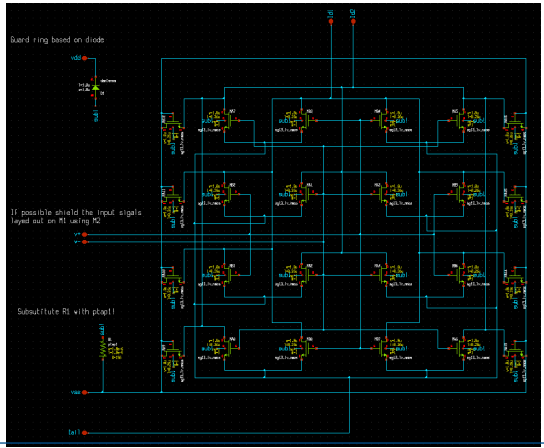
Key	Action
q	Open the properties window of an object.
m	Move the selected object.
c	Copy the selected object.
a	Align tool
o	Generates via when using path tool

Selection mode

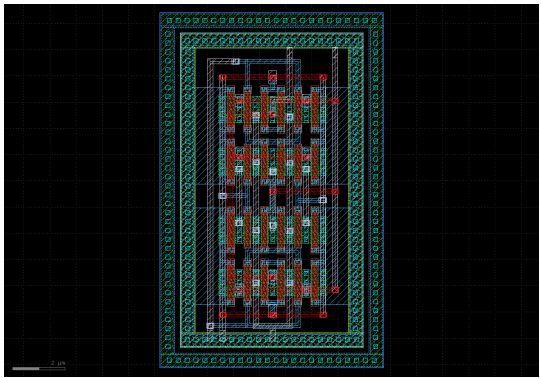
Many advanced editing operations (transformations) are available under selection.

Differential Pair: Schematic and Layout

Schematic

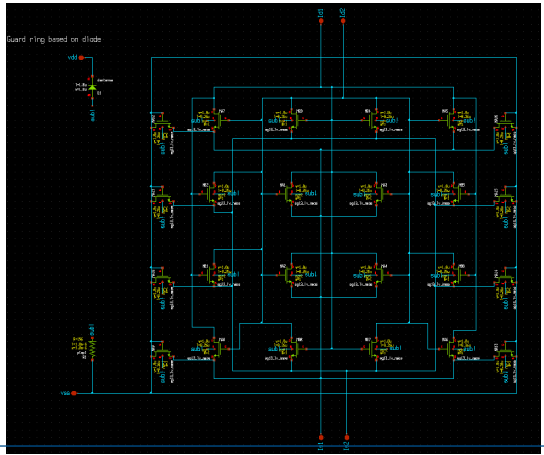


Layout

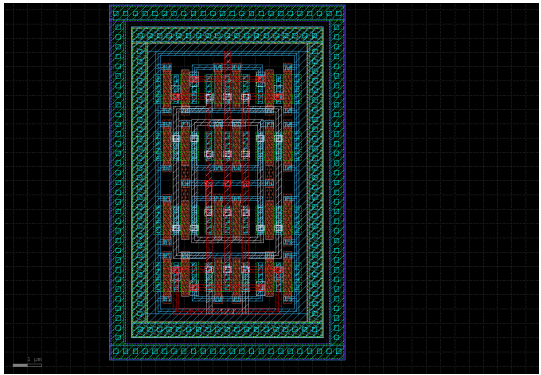


NMOS Cross-Coupled Pair: Schematic and Layout

Schematic

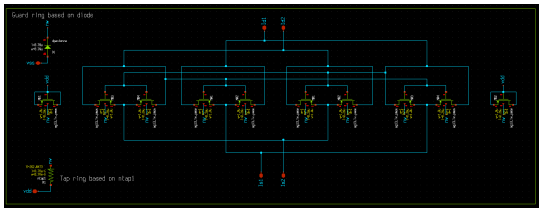


Layout

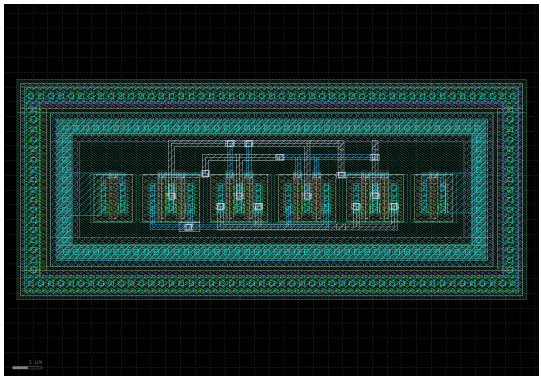


PMOS Cross-Coupled Pair: Schematic and Layout

Schematic

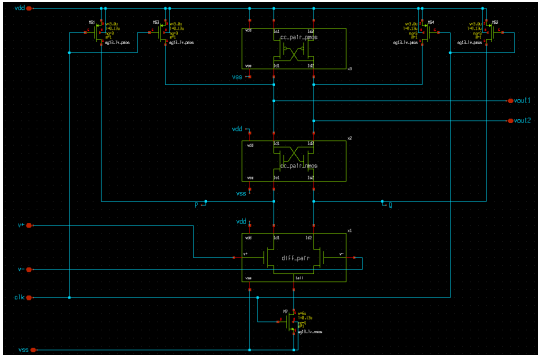


Layout

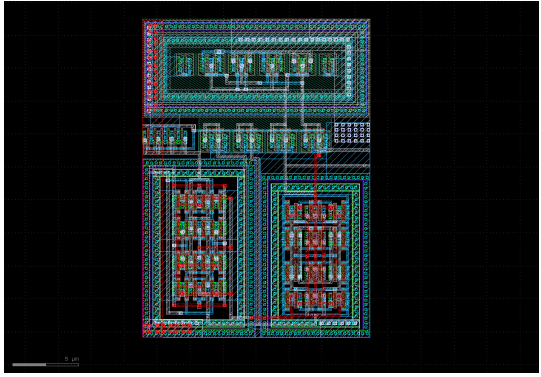


Final StrongARM Latch: Schematic and Layout

Schematic



Layout



Contents

Parasitic Extraction Flow

Extraction Command

```
cd $HOME\designsIHP__CMP7345\CMP7345-main\layout \\
kpex --pdk ihp-sg13g2 --magic --magic_cthresh=2.0 \
  --gds CMP7345.gds
```

Files to Compare

- Schematic netlist: `$HOME/IHP__CMP7345/CMP7345-main/schematic/CMP7345-final.spice`
- PEX netlist: `$HOME/IHP__CMP7345/CMP7345-main/netlist/pex/CMP7345.pex.spice`
- Goal: compare the ideal schematic view against the layout-extracted view that includes parasitic loading.

Using the Extracted View in Xschem

PEX Directory

`$HOME\designsIHP__CMP7345\CMP7345-main\netlis`

This location should contain the extracted netlist and the additional xschem symbol used to instantiate the PEX view.

PEX Testbench

`$HOME\designsIHP__CMP7345\CMP7345-main\testben`
`CMP7345_pex_tb.sch`

Include the extracted netlist here instead of the schematic-only view when running post-layout transient simulations.

Practical Check

Verify that the extracted subcircuit pin order matches the xschem symbol. If not, remap the pins or use a dedicated PEX symbol before simulation.

Contents

Physical Verification Goals

DRC

- Checks whether the layout obeys process geometry rules.
- Main rule set includes spacing, enclosure, width, and density-related checks.
- Use it early and often during layout editing.

LVS

- Checks whether layout connectivity matches the schematic netlist.
- Confirms device recognition, pin mapping, and net connectivity.
- Must pass before trusting PEX simulation results.

Comparator Relevance

For a StrongARM latch, DRC catches geometry issues on matched devices and routing, while LVS catches swapped outputs, missing taps, wrong pin order, and cross-coupled connectivity mistakes.

KLayout DRC in SG13G2

```
python3 run_drc.py --path=CMP7345.gds \
  --run_mode=deep \
  --run_dir=drc_cmp7345 \
  --no_density
```

Useful switches

- `--topcell`
- `--table`
- `--no_density`
- `--antenna`
- `--run_mode=deep|flat`

Outputs

The run creates a log and a marker database such as `CMP7345.lyrdb`:

```
klayout CMP7345.gds -m CMP7345.lyrdb
```

What to inspect first

Check input-pair symmetry, via enclosure, local taps, clock routing, and off-grid markers first.

KLayout LVS in SG13G2

```
python3 run_lvs.py \
  --layout=CMP7345.gds \
  --netlist=CMP7345-final.spice \
  --run_dir=lvs_cmp7345 \
  --run_mode=flat
```

Useful switches

- `--topcell`
- `--no_simplify`
- `--no_series_res`
- `--no_parallel_res`
- `--verbose`

Outputs

- `CMP7345.lvsdb`
- extracted netlist `CMP7345_extracted.cir`
- execution log

Comparator-specific checks

Confirm pin order, output names, tail-device extraction, well/substrate ties, and cross-coupled PMOS/NMOS connectivity before moving to PEX.

GUI Flow in KLayout

```
KLAYOUT_PATH=$PDKPATH/libs.tech/klayout:
$PDKPATH/libs.tech/klayout/tech/ klayout -e
```

GUI usage

- The SG13G2 menus expose both DRC and LVS runs directly in KLayout.
- For LVS, the active cell is used by default unless `Top Cell` is overridden.
- If the netlist path is not set manually, KLayout searches for matching `.cdl`, `.spice`, or `.cir` files near the layout.

Recommended order

Run DRC until clean, then run LVS, and only after both checks are understood proceed to post-layout extraction and simulation.

Contents

Chip Finishing Goals

Why finishing is needed

- Dummy fill improves metal-density uniformity.
- Better density control improves CMP behavior.
- This reduces polishing non-uniformity and yield risk.

Mandatory items

- A `sealring` is required in the final design.
- Fill must remain DRC-clean after insertion.
- Density is part of the submission acceptance flow.

Reference documents

Use the IHP filler documentation together with `SG13G2_os_layout_rules.pdf`. The filler spacing and sizing constraints refer back to the layout-rule manual.

Filler Generation in KLayout

GUI flow

The SG13G2 PDK menu provides filler and density tools:

- `Fill Activ/GatPoly`
- `Fill Metal`
- `Fill Top Metal`
- `Density Report`

CLI flow

```
klayout -n sg13g2 -zz -r \
  $PDK_ROOT/$PDK/libs.tech/klayout/tech/scripts/filler.py \
  -rd output_file=./design_filled.gds \
  design_nofill.gds
```

Rule note

For Metals 1–5, the documented minimum filler spacing is `0.42um`. Re-run default DRC after any density tuning.

Density Check and Alternative Flow

Post-fill verification

- Run `Density Report` first.
- Run default DRC with density checks enabled.
- For large layouts, `Run Only Density Check` can speed up debug.
- `AFil.g2` and `M1-5Fil.h` markers outside the design area may be safely waived in minimal DRC.

Alternative: `gdsfill`

```
gdsfill density <my-layout.gds>
gdsfill erase <my-layout.gds>
gdsfill fill <my-layout.gds>
```

Optional modes: `--keep-data`, `--dry-run`, and custom config files.

Recommended order

Make the layout DRC-clean, add the sealring, insert fill, run density checks, re-run full DRC, and only then do the final post-layout signoff pass.

IHP Technology Options

SG13G2

- SiGe BiCMOS platform.
- HBTs up to about **350 GHz** transit frequency.
- Low/high-voltage CMOS devices.
- MIM cap, Schottky, S-varicap, ESD options.
- **7-layer Al BEOL**: 5 thin metals plus 2 thick top metals.

SG13CMOS and SG13CMOS5L

- **SG13CMOS**: RF CMOS option without bipolar HBTs.
- **SG13CMOS5L**: open-source education CMOS option.
- Reduced device set and fewer metal layers.
- No isolated NMOS and no MIM capacitor.
- Lower-cost option for teaching shuttles.

Why it matters here

The comparator is kept portable, but the target process still sets devices, metal stack, turnaround, and MPW price.

IHP Open-Silicon MPW Access

Access models

- **Open-source MPW**: lowest-cost entry for **Apache 2.0** releases with open-source-compatible design data.
- **Private IP MPW**: same shuttle with about **20% off** the standard MPW price.
- Participation requires the IHP MPW program agreement.

What is included

- **40 bare die samples**
- wafer-rental option for measurements
- packaging services on request

Typical shuttle numbers

- **Minimum area target**: **90 mm^2** shared across customers
- **SG13CMOS5L**: about **1500 EUR/mm^2** with lower target pricing after area threshold
- **SG13G2**: listed around **2800 EUR/mm^2**

Typical turnaround

SG13CMOS: about **4 months**
SG13G2: about **6 months**

Contents

Thank you for your attention!

IHP GmbH – Innovations for High Performance Microelectronics
Im Technologiepark 25
15236 Frankfurt (Oder)
Tel. +49 (0) 335 5625 380
herman@ihp-microelectronics.com

Leibniz Association