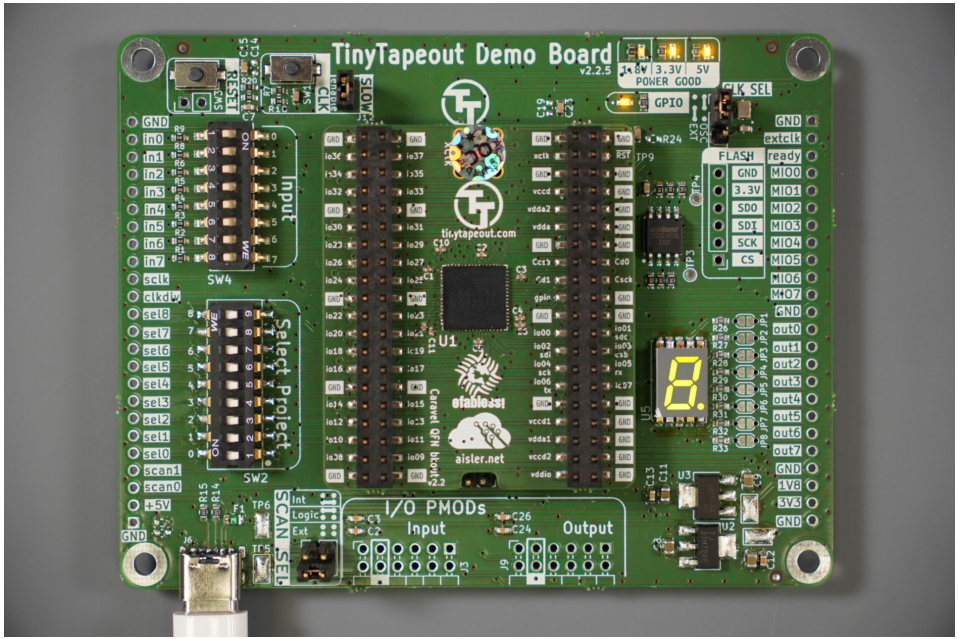
SRAM22: An Open-Source SRAM Compiler for SkyWater 130nm

Rahul Kumar, **Rohan Kumar**, Ethan Wu, Allison Husain, Connor Lu, Viansa Schmulbach, Borivoje Nikolić

Note: some slides/figures removed to avoid pre-publication.

Motivation

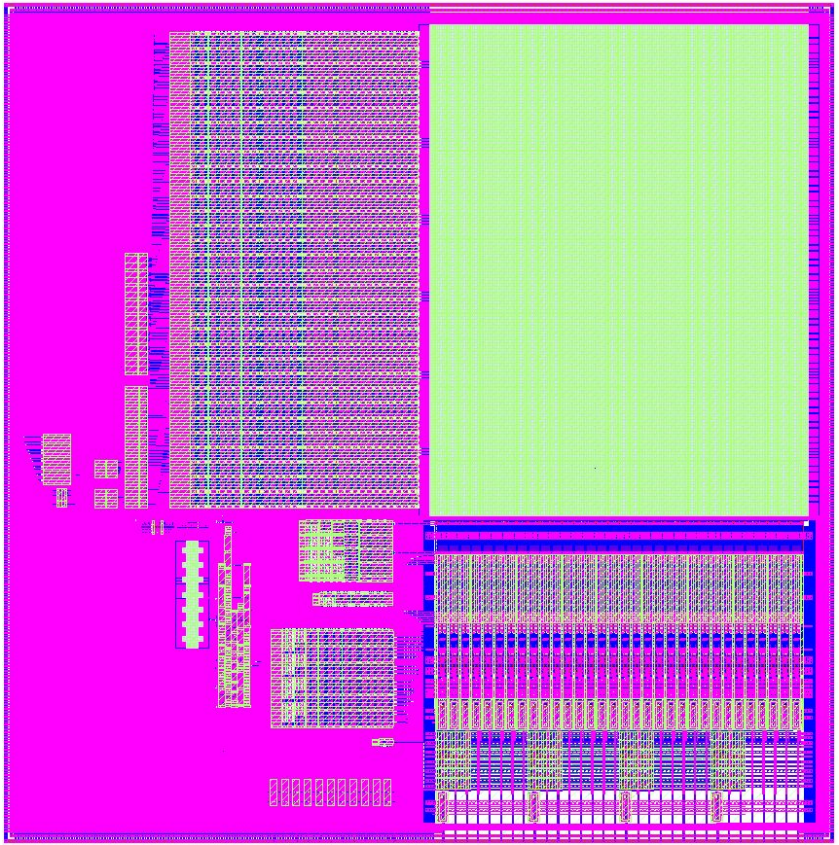
- SKY130 is a great platform for reproducible research and education
- However, SKY130 lacks well-characterized and easy-to-integrate open-source SRAM macros



SRAM22 Overview

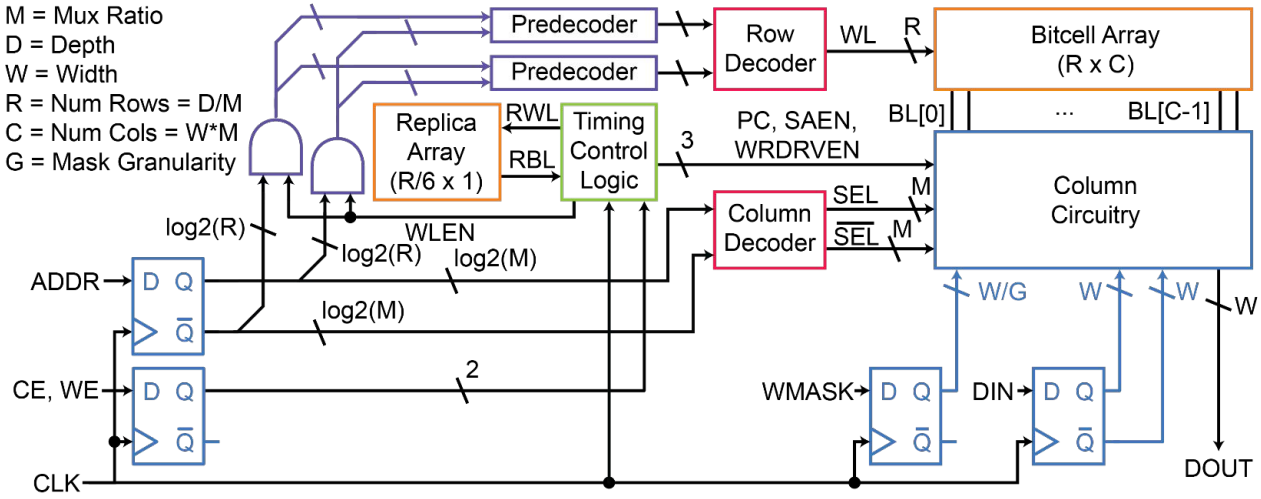
SRAM22 is an SRAM generator for SKY130.

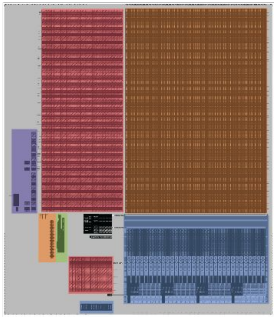
- Generates single-port SRAMs
- Only uses up to metal 2
- Replica timing circuitry
- Pins on metal 1 along bottom edge
- Codified design and verification flow



SRAM Architecture

- Inputs saved into registers at rising edge of clk
- Row decoder decodes upper bits of address
- Col decoder decodes lower bits
- Timing control circuit controls wordline pulse width, sense amps, precharge, etc.

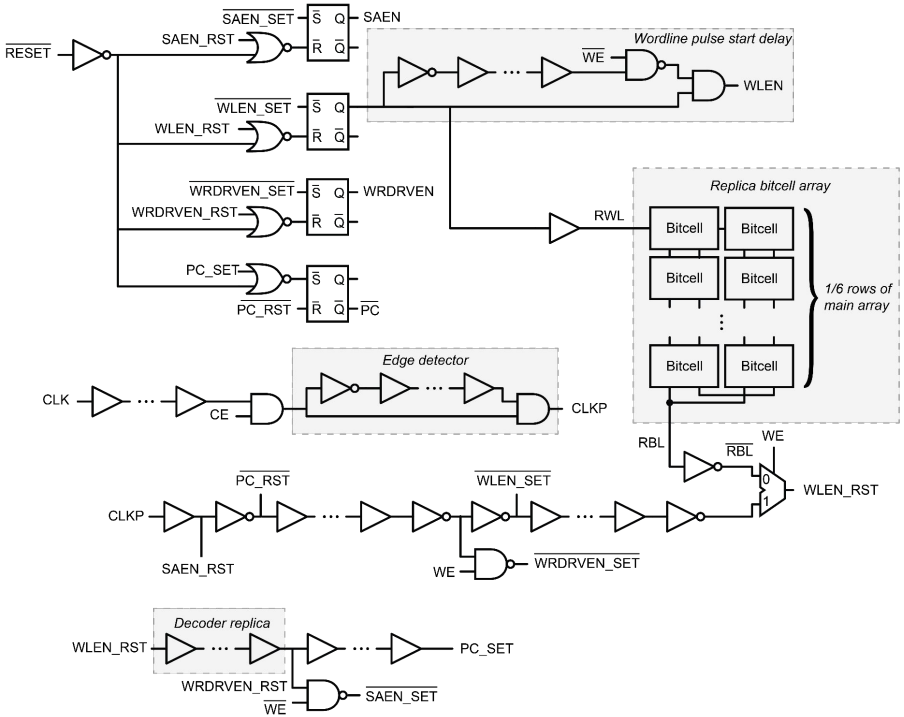




D = 2048, W = 32,
M = 8, G= 8

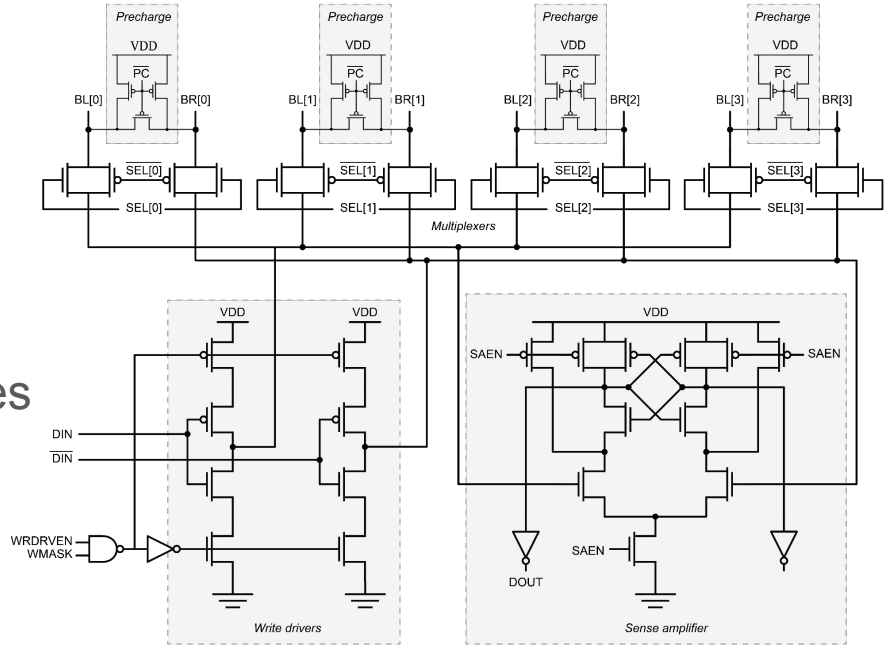
Replica Bitline Timing

- Edge-triggered pulse generator starts read/write operation
- Sense amp enable timing determined by replica bitcell array
- Replica bitcell array has ~1/3 the number of cells vs main bitline, so discharges faster
- Main bitline discharges by only ~0.3V, reducing energy dissipation compared to full swing



Column Circuitry

- Precharges bitlines before/after read/write operations
- Implements column muxing: only 1 of every 4 or 8 columns is accessed on any given cycle
- Drives bitline/complement during writes
- Compares bitline/complement during reads by using a StrongARM sense amplifier

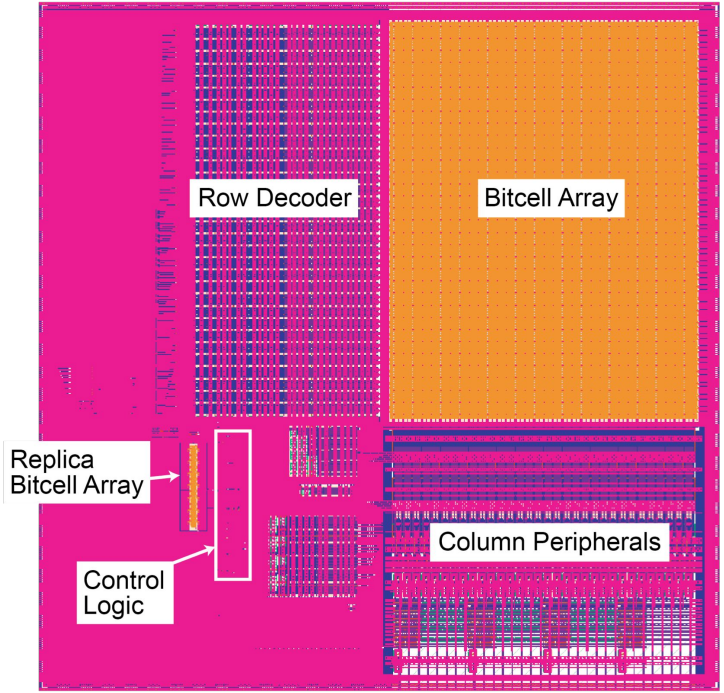


Generator Flow

- SRAM22 uses pre-extracted per-cell wordline and bitline capacitances
- Estimate total worldline and bitline capacitance
- Size row/col decoders, column peripherals, and control logic replicas

Generator Flow (cont.)

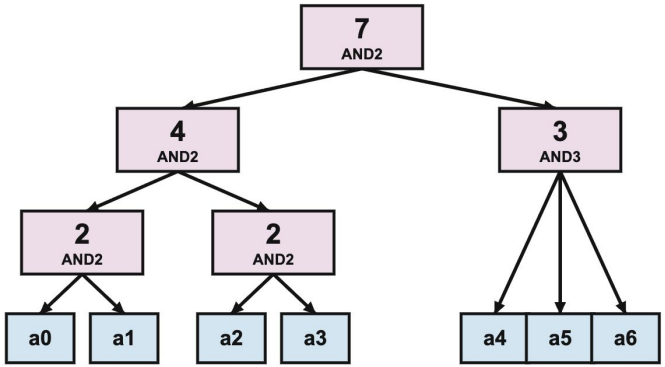
- Based on determined sizing, lay out SRAM
- Export SPICE, GDS, LEF, LIB
- Run DRC, LVS, simulations



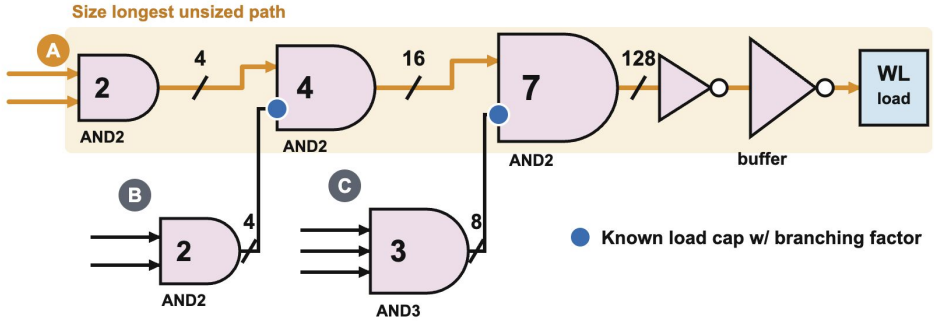
Decoder Sizing

- Decoder converts binary to one-hot via a tree of AND gates
- Programmatically compute tree structure:
  - Final stage must be AND2
  - Intermediate stages are AND3 if the number of bits in the stage is 3 or 6+
- Size each path in the decoder tree using known load capacitances
  - Wordline capacitance
  - Input capacitance of previously-sized decoder gates

Pass 1: Tree Structure

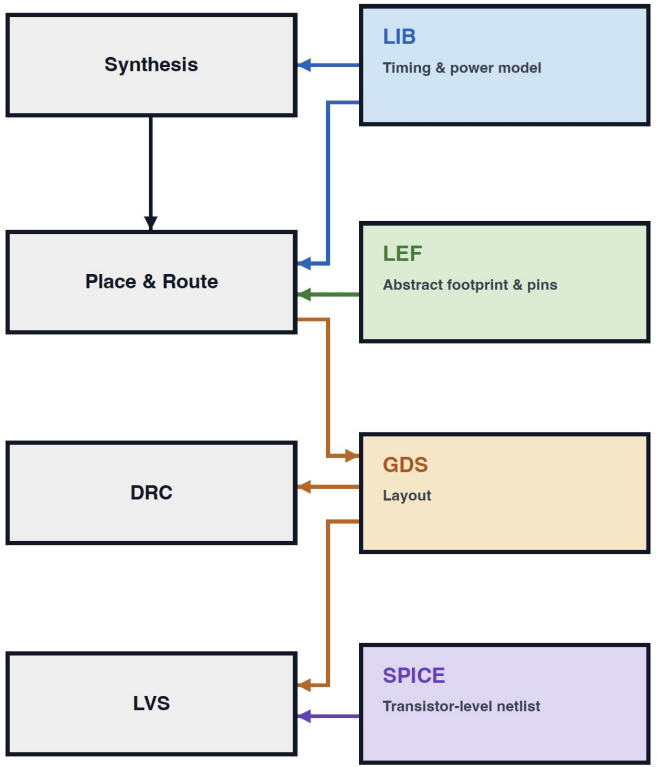


Pass 2: Iterative Path Sizing



Collateral Export

- SRAM22 exports:
  - SPICE
  - GDS
  - LEF
  - LIB (generated via commercial tools, or, if no access to commercial tools, by interpolating between pre-characterized data points)
- SRAM22 has been used in a handful of tapeouts (including at least one outside of Berkeley)



Verification

- SRAM22 automates functionality and behavioral checks
- For example, verifies that bitline differential voltage is at least 150 mV when sense amp is triggered
- SRAM22 has also been verified in silicon; we expect to publish more details in the near future

Conclusion

- There is still a lot more work to be done!
  - Improve documentation
  - Integrate with open source digital flows
  - Improve peripheral density
- If you are interested in using or contributing to SRAM22, check out the repo
  - Source: https://github.com/ucb-substrate/sram22
  - Generated macros: https://github.com/ucb-substrate/sram22_sky130_macros