ESE 5700 Project 1, Part 2: The 16 × 4 SRAM Macro

Panos Dimtsoudis (panosdim) Dhruva Jindal (djindal)

10/20/26

Team and contributions

Task	panosdim	djindal
2.1 Array layout, DRC, LVS	X	
2.1 Extraction, C_{BL} and C_{WL}	X	
2.2 Decoder and wordline timing	X	
2.3 Column circuits		X
2.4 Controller		X
2.5 Verification	X	
Metrics and FOM		X
Report writing	X	X

AI Usage Disclosure

Claude was used to help organize the report and to generate tables, figure placeholders, and figure captions. It was also used to write MATLAB scripts that produced plots from CSV data obtained from our simulations. All circuit designs, simulations, extracted data, and reported results were produced and verified by the authors.

Summary of results

Unless stated otherwise, the results use transistor-level schematic periphery connected to the extracted array at $V_{DD} = 0.8\,\mathrm{V}$. Values are rounded to two significant figures.

Quantity	Value	Section
Supply V_{DD}	0.8 V	–
Array area (µm^2)	34	2.1
Density factor D (from Part 1)	79	2.1, Metrics
C_{BL} per column, worst case (fF)	4.7	2.1
C_{WL} per row (fF)	3.1	2.1
Decoder (2-2 predecode): ADDR→WL rise / fall (ps)	129 / 89	2.2
Minimum WL pulse, read / write, unit test (ps)	140 / 70	2.2c
Minimum WL pulse, read / write, full macro (ps)	180 / 135	2.2c
WL pulse provided (ps)	200	2.2c, 2.4
Read path: $\overline{\text{BL}}$ swing / margin (unit test)	0.50 V / 80 mV	2.3
t_{access} (ps)	440	Metrics
Write time (ps)	300	Metrics
$T_{CLK,min}$ (ns)	0.70	2.5, Metrics
P_{write} (µW)	140	Metrics
Total area, array + periphery (µm^2)	110	Metrics
FOM$_{SRAM}$ (µm^2 · µW · ns)	1.1×10^4	Metrics

1 Array layout and extraction (2.1)

1.1 Hierarchy

The array is built in two levels from the Part 1 cell `sram6t_v2` (PU = PG = PD = 2 fins, 0.578 × 0.912 µm). `sram_row` places four cells side by side and brings out BL⟨3:0⟩, $\overline{\text{BL}}$⟨3:0⟩, WL, VDD and VSS. `sram_array` stacks sixteen rows and brings out WL⟨15:0⟩, BL⟨3:0⟩, $\overline{\text{BL}}$⟨3:0⟩, VDD and VSS. Both levels have a schematic (an iterated instance, `I<0:3>` and `R<0:15>`), a symbol and a layout, and each was taken through DRC and LVS before the next level was built.

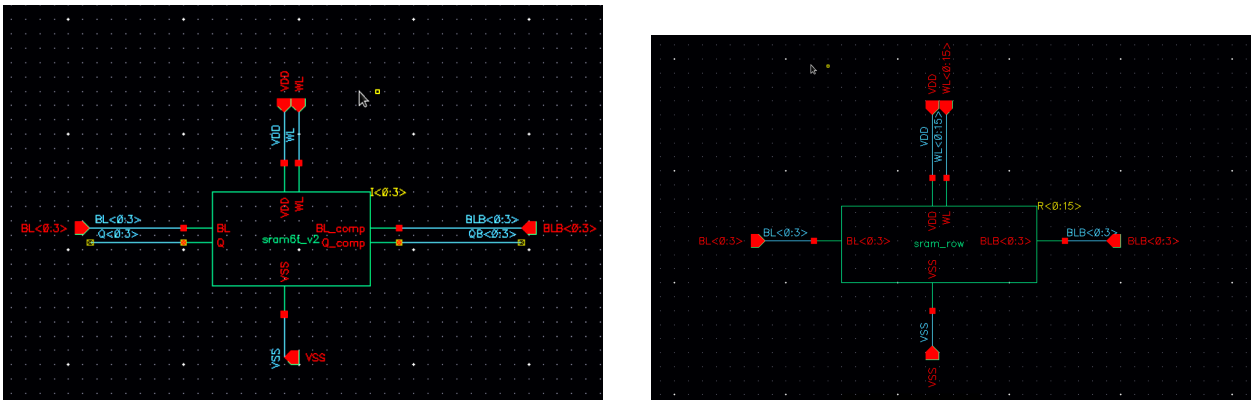


Figure 1: Left: `sram_row`, four cells sharing WL, VDD and VSS, one bitline pair per cell; the labels $Q\langle 3{:}0\rangle$ and $\overline{Q}\langle 3{:}0\rangle$ name the storage nodes for probing. Right: `sram_array`, sixteen rows sharing the four bitline pairs, one wordline per row.

1.2 Floorplan and tiling

The cell was drawn so that VDD, VSS and WL run horizontally from edge to edge and BL and $\overline{\text{BL}}$ run vertically from top to bottom (Table 1). The VDD and VSS rails, and the substrate and well taps on them, are centred on the cell's top and bottom edges, so half of each rail lies inside the cell.

Net	Layer	Direction	Shared by
WL$\langle k \rangle$	M1	horizontal	the 4 cells of row k
VDD, VSS rails (with taps)	M1	horizontal	the 2 rows that meet at the rail
BL$\langle j \rangle$, $\overline{\text{BL}}\langle j \rangle$	M2	vertical	the 16 cells of column j
VSS strap (column 0), VDD strap (column 3)	M2	vertical	all rails of that supply

Table 1: Layers and routing direction of the array nets.

Rows. Rows alternate between R0 and MX (mirrored about the horizontal axis) at a pitch of 0.912 µm. Even row k is placed at R0 with its origin at $y = 0.912k$; odd row k is placed at MX with its origin at $y = 0.912(k + 1)$, so that it occupies the same 0.912 µm slot upside down. Two neighbouring rows therefore face each other with the same supply: rows $2m$ and $2m + 1$ share one VDD rail, and rows $2m + 1$ and $2m + 2$ share one VSS rail. Each rail and each row of taps serves two rows of cells, and the array has 9 VSS rails and 8 VDD rails instead of 16 of each. No separate tap cells are needed: the shared edge taps pass DRC as placed.

Columns. Columns are placed at R0 with a pitch of 0.578 µm. We did not mirror the columns. Because VDD, VSS and WL run horizontally, the only shapes on a cell's left and right edges are the bitlines, and the bitlines of two columns are different signals that cannot be shared. Mirroring the columns would place $\overline{\text{BL}}\langle j \rangle$ next to $\overline{\text{BL}}\langle j + 1 \rangle$ instead of next to BL$\langle j + 1 \rangle$ without saving any area. Keeping every column in the same orientation also gives every bitline the same neighbours, so all four columns see the same coupling.

Power straps. Each row-pair rail is a separate M1 shape, so the rails have to be tied together for LVS and for a real supply. Two vertical M2 straps do this. The VSS strap runs along the left edge of column 0 and the VDD strap along the right edge of column 3. Both sit in the empty margin between the cell edge and the nearest bitline, so they add no area. Each VSS rail connects to the left strap and each VDD rail to the right strap with an M2_M1 via, so no stub has to cross the other supply's strap. The VDD and VSS pins sit on the straps, the WL$\langle k \rangle$ pins on M1 and the bitline pins on M2. Both straps are 32 nm wide. The VSS strap spans $x = 0$ to 0.032 µm and the VDD strap $x = 2.280$ to 2.312 µm. Both run from $y = -0.016$ to 14.608 µm, 16 nm past the bottom and top cell edges, where the outermost rails (centred on those edges) also extend.

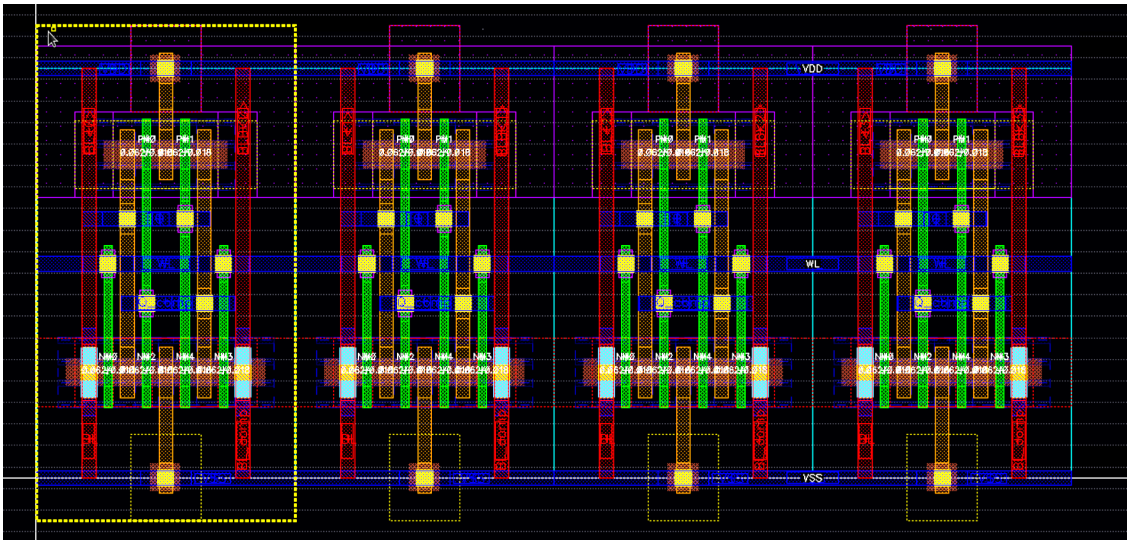


Figure 2: `sram_row`: four cells at R0 on a 0.578 µm pitch. WL, VDD and VSS run edge to edge through all four cells; each cell has its own BL and $\overline{\text{BL}}$ on M2.

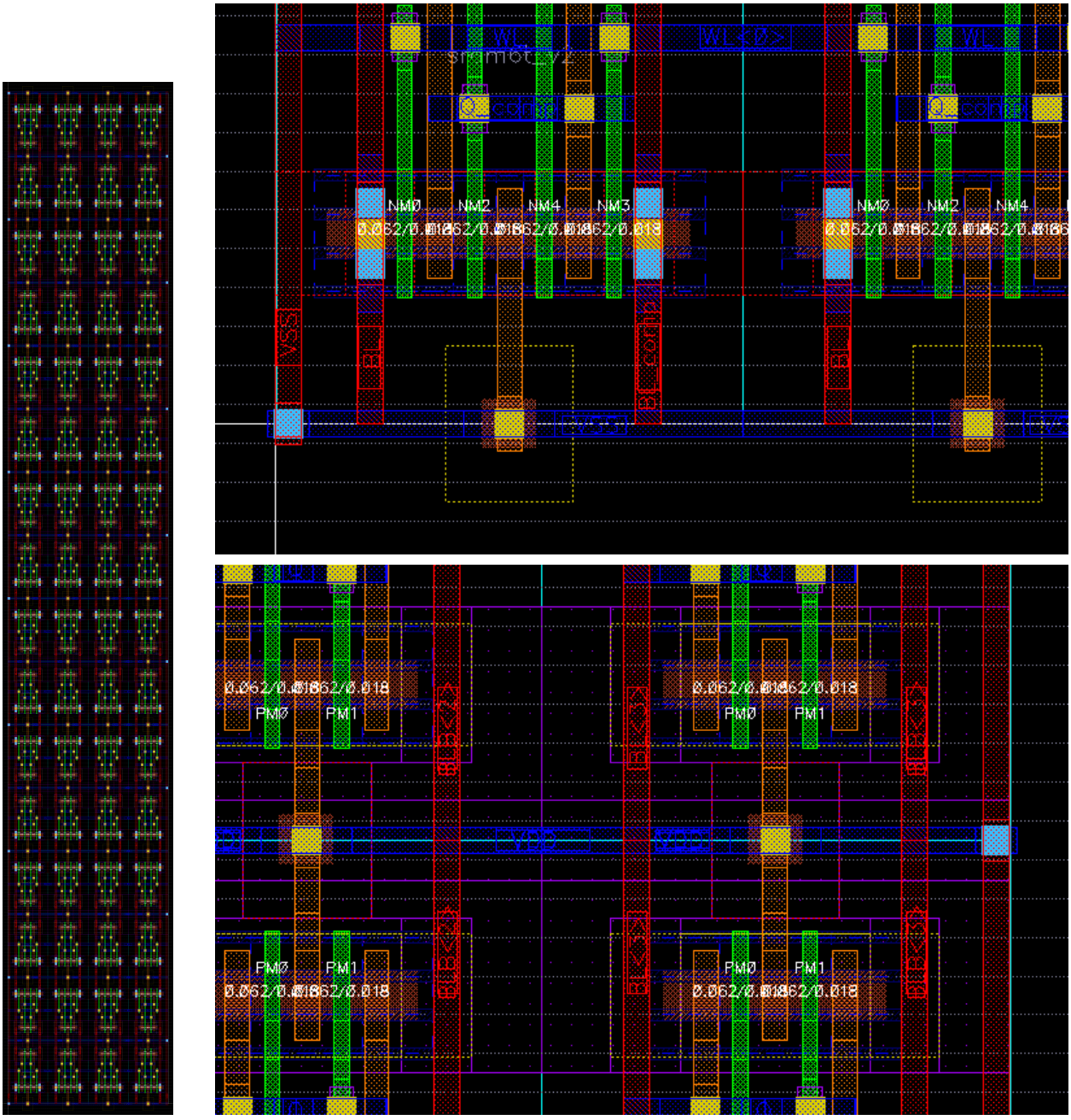


Figure 3: Left: the full 16×4 array. Right: the VSS strap inside column 0 (top) and the VDD strap inside column 3 (bottom), each connected to its rails through M2_M1 vias.

1.3 Area

The cells tile with no gaps, so the cell region is exactly $4 \times 0.578 = 2.312\,\mu\text{m}$ wide and $16 \times 0.912 = 14.592\,\mu\text{m}$ tall. The straps sit inside it, so the array is no larger than its cells:

$$A_{array} = 2.312 \times 14.592 = 33.74\,\mu\text{m}^2 = 64 \times 0.5271\,\mu\text{m}^2 \approx 3.4 \times 10^5\,F^2 \quad (F = 10\,\text{nm}). \qquad (1)$$

The cells fill 100% of the array. Our first version placed both straps outside the cells, on the left, which added a $0.202\,\mu\text{m}$ column: $2.514 \times 14.592 = 36.68\,\mu\text{m}^2$, with the cells filling 92%. Moving

the straps into the first and last columns removed that 8%. The cost is extra capacitance on the two bitlines next to the straps (Section 1.6). The strap ends and the outer halves of the two edge rails extend 16 nm past the top and bottom of the cell region. This adds 0.2% to the height, which we neglect.

1.4 DRC and LVS

Both `sram_row` and `sram_array` are DRC clean, and their layout and schematic netlists match. LVS on the array also reports 16 "unattached port label" warnings, all from port labels inside the subcells: 6 in `sram6t_v2` (Q, WL, VDD, VSS, BL, $\overline{\text{BL}}$) and 10 in `sram_row` (VDD, VSS and the eight bitlines). In this run, the warnings come from labels inside the hierarchical subcells. Renaming the labels, drawing metal under them and enabling automatic cell matching did not remove them. The extracted and schematic netlists still match.

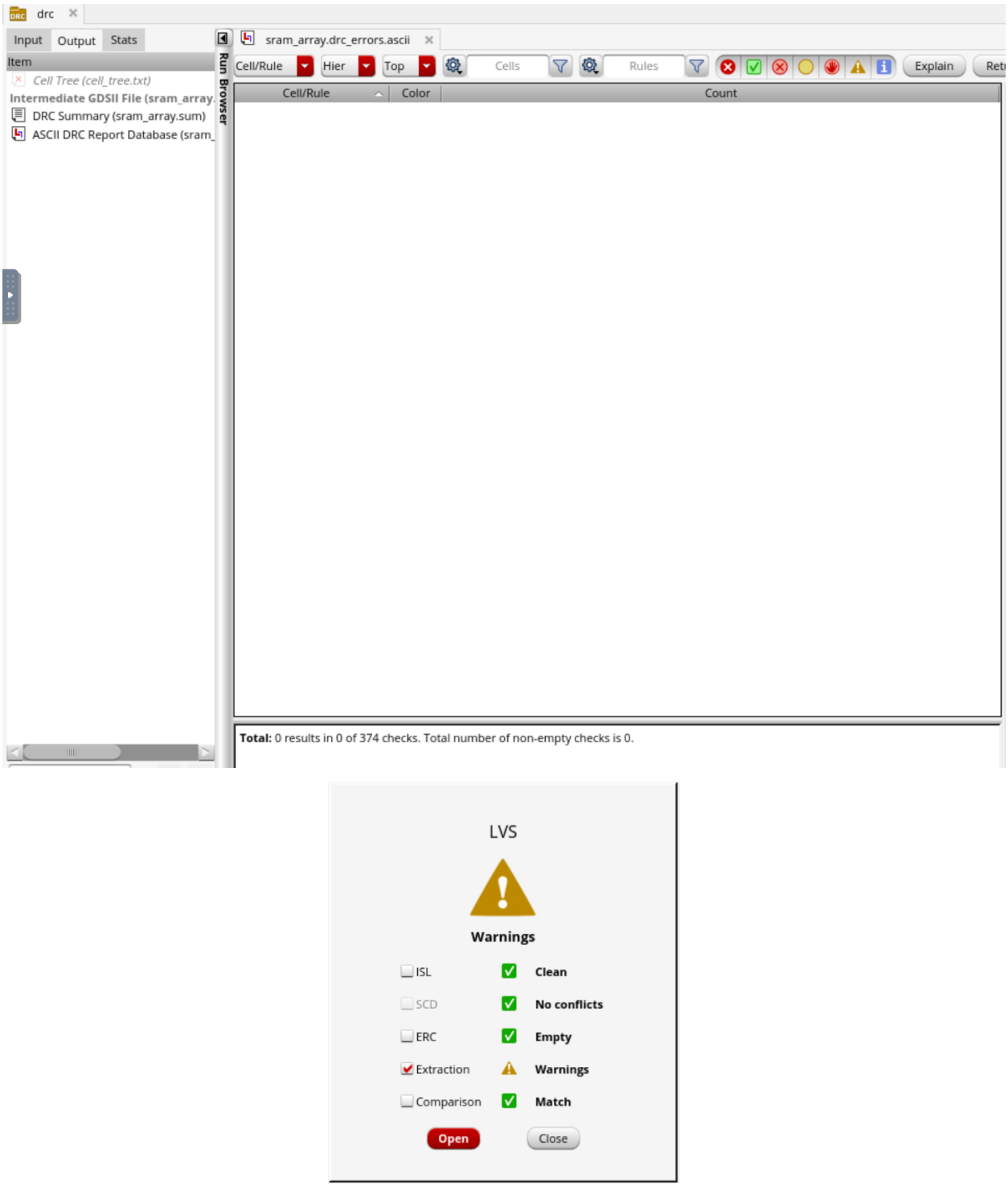


Figure 4: Pegasus results for the full array. Top: DRC, 0 results in 374 checks. Bottom: LVS, with ISL clean, no SCD conflicts, ERC empty and the comparison a match; the extraction warnings are the unattached port labels described above.

1.5 Extraction

The array was extracted with Quantus (`cds_ff_mpt_qrc`, typical corner, RC coupled, reference node VSS) into `av_extracted`. The extracted view contains the expected devices and nets (Table 2). The resistor and capacitor counts are 64 times the single cell's, plus 347 coupling capacitors between neighbouring cells and to the straps. The 154 signal nets are exactly the 128 internal nodes (Q, $\overline{Q}$ of 64 cells), 16 wordlines, 8 bitlines and the two supplies, so no net is split or merged. The 64 floating nets are one per cell, as in the single-cell extraction.

Quantity	`sram_array`	64× `sram6t_v2`
Devices (`n1svt` / `p1svt`)	256 / 128	256 / 128
Resistors	5,730	5,760
Capacitors	15,963	15,616
Signal nets	154	–
Floating nets	64	64

Table 2: Quantus summary for the array, against 64 copies of the single cell.

```
--------------------------------------------------------------------------------
EXTRACTION stage summary:
--------------------------------------------------------------------------------
 Design data:
    LVS nets:                    218
    signal nets:                 154
    floating nets:               64
    ground net name:             VSS
--------------------------------------------------------------------------------
OUTPUT stage summary:
--------------------------------------------------------------------------------
 Reduction statistics:
    Dangling Rs removed:         0
    Merged Parallel Rs:          0
    Mfactor Merged devices:      0
    Rs shorted by min_res option: 1408
    Merged Parallel Self Caps:   0
    CCs filtered by minC option:  10712
    Shorted Incomplete Nets:     0
 Output statistics:
    R: 5730
    C: 15963
    L: 0

    Summary for proj1_lib/sram_array/av_extracted

    instance count totals:

        lib             cell            view                    total
        analogLib       pcapacitor      symbol                  15963
        analogLib       presistor       symbol                   5730
        cds_ff_mpt      n1svt           symbol                    256
        cds_ff_mpt      p1svt           symbol                    128

    Extracted view creation has completed.
--------------------------------------------------------------------------------
INFO (LBRCXM-702): Run ended: Fri Oct 16 19:58:38 2026
INFO (LBRCXM-708): *****  Quantus terminated normally  *****
```

Figure 5: Summary sections of the Quantus log for `sram_array`: 154 signal nets and 64 floating nets, 5,730 R and 15,963 C, and the instance counts (256 `n1svt`, 128 `p1svt`).

Connectivity check. Quantus issued 24 MRGVIA-131093 warnings (the log lists at most 24). Twelve are at the VSS strap vias and twelve at the M2_M1 vias of BL⟨0⟩ in rows 0–2, the bitline next to the VSS strap. In each case the via's M2 has the other mask colour from the wire's M2,

which Quantus says "may create opens". The same warning appeared on the bitline vias in Part 1 without producing a netlist mismatch. We checked the array in two ways. First, an open bitline via would cut cells off BL⟨0⟩ and lower its capacitance, but the measured C_{BL} is higher than the per-cell estimate (Table 4). Second, we ran a DC operating point on the extracted array (test bench in Section 1.6) and compared row 0, whose VSS strap via and BL⟨0⟩ vias are flagged, with row 15.

In this test, the DC solution places every unselected cell at its metastable point ($Q = \overline{Q} \approx V_M$), because the cells are symmetric and the solver starts from a symmetric guess. Both inverters of each such cell conduct, which draws hundreds of microamps from VDD to VSS in total. We therefore measured the total bitline current, which only the selected row can draw, using Kirchhoff's current law at the array: $I_{bit} = -(I_{VDD} + I_{VSS})$.

- **VSS path (read):** WL on, all bitlines at 0.8 V. Bitline current flows through the pass gates and pull-downs into the row's VSS rail.
- **VDD path:** WL on, all bitlines at 0 V. The pull-ups fight the pass gates, and the current comes from the row's VDD rail.

Test	Row 0 (vias flagged)	Row 15	Difference
VSS path, I_{bit} (µA)	23.4	21.8	7%
VDD path, I_{bit} (µA)	−144	−133	9%

Table 3: Connectivity check on the extracted array.

An open via would leave a rail or bitline floating and the row's current near zero. Instead, row 0 draws slightly more than row 15 in both tests, which shows that the straps reach their rails and BL⟨0⟩ reaches its cells. The 7–9% difference may be caused by voltage drop along the straps during the artificial metastable-current test.

1.6 Bitline and wordline capacitance

The test bench `tb_array` (Figure 6) instantiates one `sram_array`, bound through a config view to `av_extracted`. Every line has its own ideal source: WL⟨0⟩, WL⟨15⟩ and WL⟨14:1⟩ (iterated), BL⟨0⟩, BL⟨3:1⟩ and $\overline{\text{BL}}$⟨3:0⟩, plus VDD and VSS. For the capacitance, all wordlines are held at 0 V (the cells hold) and all bitlines at 0.8 V (precharged). One line at a time is driven with a 1 V AC source on top of its DC level, and the capacitance is read from the imaginary part of the source current at 1 MHz:

$$C = \frac{|\operatorname{Im} I|}{2\pi f}, \qquad f = 1\,\text{MHz}. \tag{2}$$

All other lines sit at their DC level and are therefore AC grounds, so the result is the total capacitance of the driven line, including its coupling to every other line. Driving one line at a time keeps the coupling between the driven bitline and wordline from cancelling part of the measurement.

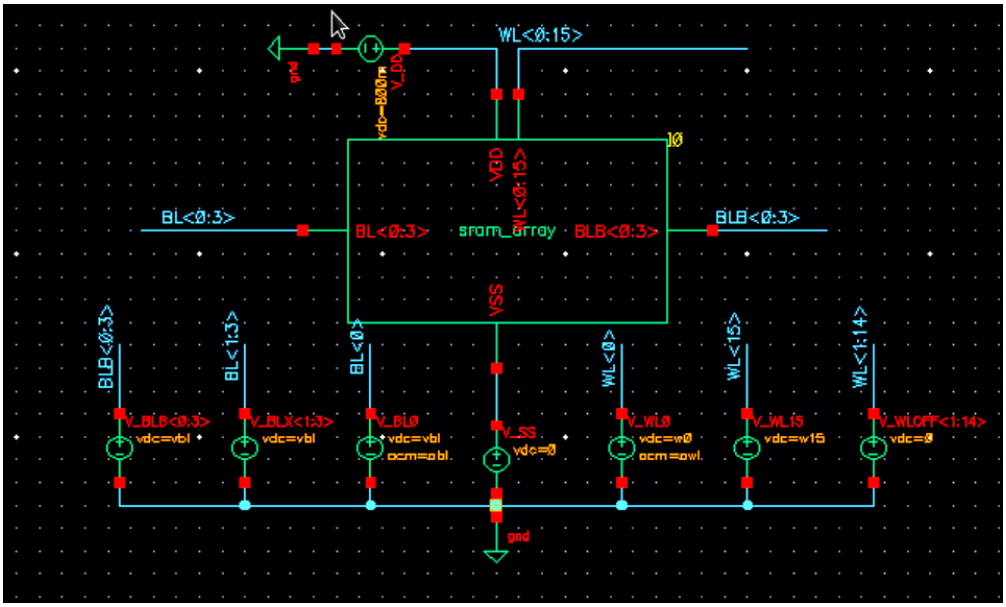


Figure 6: Test bench `tb_array`: the extracted array with one source per line. The same bench, with different variable values, gives the connectivity check (DC) and the capacitances (AC).

Quantity	Extracted array	Part 1 per cell $\times n$	Difference
C_{BL}, BL$\langle 0\rangle$, next to the VSS strap (fF)	4.723	$16 \times 0.268 = 4.288$	+10%
C_{WL}, WL$\langle 0\rangle$ (fF)	3.127	$4 \times 0.779 = 3.118$	+0.3%
BL$\langle 0\rangle$–WL$\langle 0\rangle$ coupling (fF)	0.140	–	–

Table 4: Extracted array capacitances against the scaled Part 1 per-cell values. Shown with extra digits so the small differences are visible.

Wire estimate. The Part 1 per-cell values already include the cell's own piece of each line: the WL crosses the full cell width and the bitlines its full height. Because the cells abut with no gaps and the pins are at the array edge, the array adds no wire to either line beyond what the cells contain, so the wire term in the estimate is zero. The wordline agrees: C_{WL} matches 4× the cell value to 0.3%. C_{BL} of the edge bitline BL$\langle 0\rangle$ is 10% (0.44 fF) above 16× the cell value. Most of this is coupling to the VSS strap, which runs beside it for the full 14.6 µm. With the straps outside the cells, the same bitline measured 4.366 fF (+1.8%), so interior bitlines, which have no strap beside them, stay close to 16× the cell value. The same applies to $\overline{\text{BL}}\langle 3\rangle$ next to the VDD strap. Because the straps carry DC supplies, this coupling adds load but no noise. The bitline-to-wordline coupling, 0.14 fF for one crossing, is the charge the wordline edge injects into the bitline at the selected cell.

Loads for the rest of Part 2. We use the worst-case edge bitline, $C_{BL} = 4.7$ fF (4.72 fF in calculations), for the column circuits and the read timing, and $C_{WL} = 3.1$ fF (3.13 fF) as the load of each wordline driver in the decoder. The read path senses only $\overline{\text{BL}}$ (Section 3), so of the two edge bitlines only $\overline{\text{BL}}\langle 3\rangle$, next to the VDD strap, slows the read. The full-macro runs confirm this: column 3 is the slowest column to read a 1 (Section 5).

2 Row decoder and wordline timing (2.2)

2.1 Logical effort in this kit

All periphery gates are built from the unit inverter `inv_u`: a 2-fin NMOS and a 3-fin PMOS, $L = 18\,\text{nm}$. From the Part 1 device data, an NMOS fin drives $19.9/15.1 = 1.32$ times the current of a PMOS fin at $|V_{GS}| = |V_{DS}| = 0.8\,\text{V}$, so equal rise and fall need a PMOS 1.32 times wider. With two fins as the minimum, 3:2 = 1.5 is the closest ratio.

The test bench `tb_inv` (Figure 7) measures the unit inverter's input capacitance and its delay. C_{in} uses the same AC method as C_{BL}, at an input bias of 0 V and 0.8 V. The delays use a pulse source shaped by one inverter, then the inverter under test, loaded by one (FO1) or four (FO4) unit inverters, measured at $V_{DD}/2$. With $d = \tau(h + p_{inv})$, the two fan-outs give τ and p_{inv}:

$$\tau = \frac{d_{FO4} - d_{FO1}}{3} = \frac{35.6 - 17.9}{3} = 5.9\,\text{ps}, \qquad p_{inv} = \frac{d_{FO1}}{\tau} - 1 = 2.0. \tag{3}$$

Quantity (`inv_u`, 2n/3p fins)	Value
C_{in} at $V_{in} = 0$ / 0.8 V (fF)	0.532 / 0.486
C_{in}, average (fF)	0.509 (0.10 fF per fin)
FO1 delay, rise / fall (ps)	17.3 / 18.5
FO4 delay, rise / fall (ps)	34.6 / 36.7
τ (ps)	5.9
p_{inv}	2.0
Best stage effort ρ, from $\rho(\ln\rho - 1) = p_{inv}$	4.3

Table 5: Logical-effort calibration of the unit inverter.

Rise and fall differ by less than 7%, so the 2:3 ratio gives reasonably balanced delays. The measured parasitic delay is about twice the textbook value, which raises the best stage effort from about 4 to 4.3. With this unit, the logical efforts are $g_{NANDk} = (2k + 3)/5$ (NAND2 1.4, NAND3 1.8, NAND4 2.2) and $g_{NOR2} = (2 + 3 \cdot 2)/5 = 1.6$, with $p = k\,p_{inv}$ for a k-input gate.

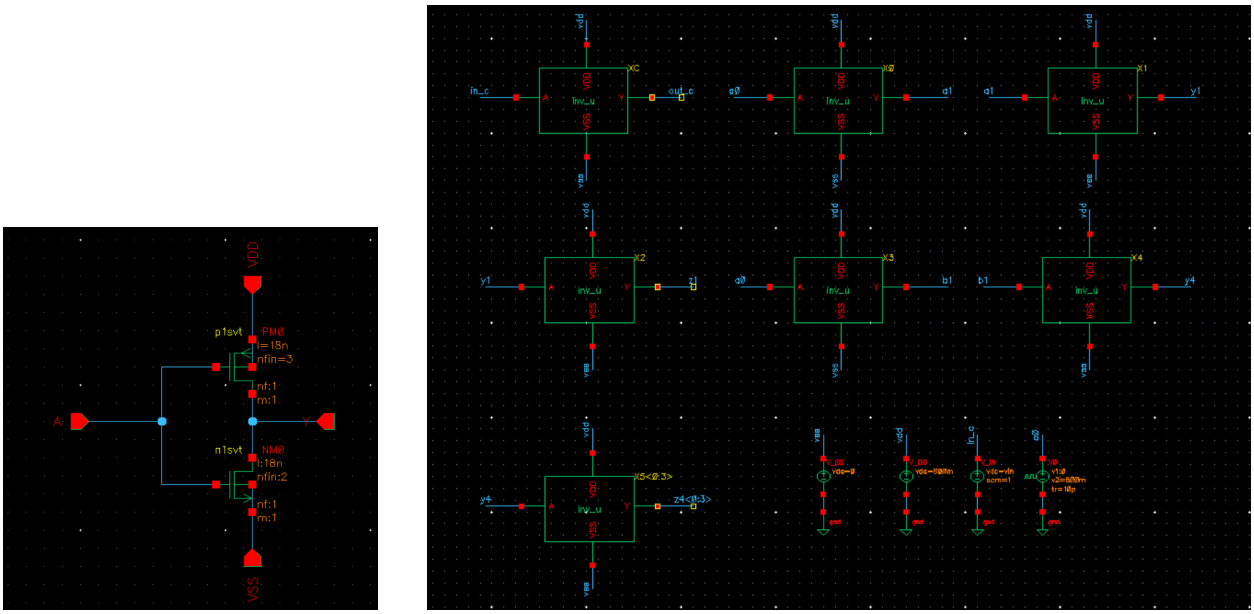


Figure 7: Left: the unit inverter. Right: `tb_inv`, with the AC input-capacitance bench and the FO1 and FO4 delay chains.

2.2 Path effort (2.2a)

The path runs from an address input to one wordline. Each address literal (true or complement) is driven by a unit-size buffer, so the address-input capacitance is $C_{in,addr} = 0.509\,\text{fF}$. The load is the extracted wordline, $C_{WL} = 3.13\,\text{fF}$ (Section 1). Each literal reaches 8 of the 16 rows: directly in a flat decoder, and through 2 predecode gates that each feed 4 rows in a 2-2 predecode. So

$$H = \frac{C_{WL}}{C_{in,addr}} = \frac{3.13}{0.509} = 6.1, \qquad B = 8, \qquad F = G\,B\,H \approx 50\,G. \tag{4}$$

With G between 2 and 2.2, $F \approx 100$ and the best number of stages is $\hat{N} = \ln F / \ln \rho \approx 3.2$: three stages, counting the literal buffer, at a stage effort of about 4.8.

2.3 Implementations compared (2.2b)

We evaluated three decoders, all starting with the unit literal buffer:

1. **Flat:** INV, then a NAND4 per row, then an inverter as the wordline driver.
2. **2-2 predecode, NAND rows:** INV, NAND2, INV (predecode), then a NAND2 and an inverter per row.
3. **2-2 predecode, NOR rows:** INV, NAND2 (predecode, active-low outputs), then a NOR2 per row that drives the wordline directly.

For each, the ideal logical-effort sizes were rounded to legal fin counts (at least 2), and the path delay was re-evaluated with the rounded sizes (`scripts/le_decoder.py`).

Design	N	G	B	H	F	$\hat{f}$	P (τ)	D ideal (ps)	D sized (ps)
Flat: INV, NAND4, INV	3	2.20	8	6.1	108	4.76	12.1	156	158
2-2 predecode: INV, NAND2, INV, NAND2, INV	5	1.96	8	6.1	96	2.49	14.2	157	157
2-2 predecode: INV, NAND2, NOR2	3	2.24	8	6.1	110	4.79	10.1	145	143

Table 6: Logical-effort comparison. D ideal uses continuous sizes; D sized uses the fin counts of Table 7.

Design	Predecode gate	Predecode INV	Row gate	WL driver
Flat	–	–	NAND4 2/2	INV 3/4
Predecode, NAND rows	NAND2 3/3	INV 5/6	NAND2 4/3	INV 5/7
Predecode, NOR rows	NAND2 6/5	–	NOR2 3/7 (row gate and driver)	

Table 7: Fin counts (NMOS/PMOS, per transistor). The literal buffer is `inv_u` (2/3) in every design. NAND NMOS are in series and PMOS in parallel; NOR NMOS are in parallel and PMOS in series. Total fins, including the 8 literal buffers: flat 408, predecode with NAND rows 640, predecode with NOR rows 536.

The wordline is a small load, only six unit inverters, so the branching effort dominates and the best path is short. The flat decoder has the right number of stages, but its NAND4 is already at the minimum size and carries the largest parasitic delay ($4p_{inv} = 8.1\tau$, using the unrounded value of p_{inv}). The predecoder with NAND rows has the lowest G but five stages, and its extra parasitic delay cancels the gain. The predecoder with NOR rows has three stages and the smallest total parasitic delay (10.1τ), so it is predicted to be fastest: 143 ps against 158 ps for the flat decoder, at 31% more fins.

Simulation. The test bench `tb_dec` (Figure 8) drives both schematic decoders from the same address signals and loads each with its own copy of the extracted array (through a config view, `sram_array` → `av_extracted`), with all bitlines held at 0.8 V. A single pulse source, shaped by two `inv_u` stages, switches all four address bits together between 0000 and 1111. We use this as a stress case because every literal changes: WL⟨15⟩ rises while WL⟨0⟩ falls, and then the reverse. The delay is measured at $V_{DD}/2$ from $\overline{\text{ADDR}}\langle 0\rangle$, the input whose buffer drives the true literals that select row 15, to WL⟨15⟩.

Decoder	WL rise (ps)	WL fall (ps)	Average (ps)	Predicted (ps)	Error
Flat: INV, NAND4, INV	204	102	153	158	+3%
2-2 predecode, NOR rows	129	89	109	143	+32%

Table 8: Simulated address-to-wordline delay of the two decoders driving the extracted array, against the logical-effort prediction (sized gates).

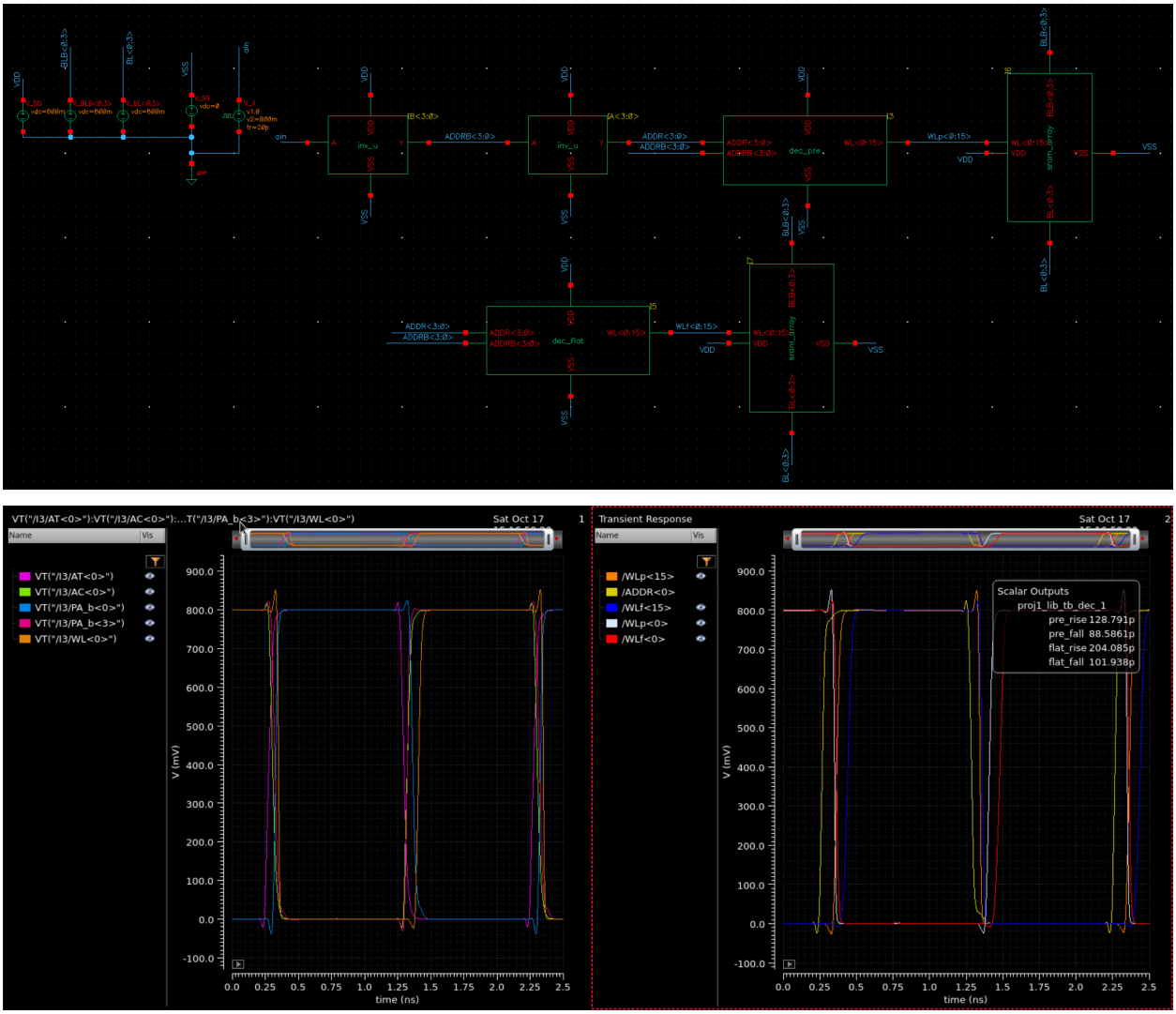


Figure 8: Top: `tb_dec`. Bottom left: internal nodes of the predecoder (literals `AT<0>` and `AC<0>`, predecoded lines `PA_b<0>` and `PA_b<3>`, and `WL<0>`). Bottom right: ADDR⟨0⟩ and wordlines 0 and 15 of the predecoder (WLp) and the flat decoder (WLf), with the measured delays.

Logical effort ranks the two designs correctly. For the flat decoder it is within 3% of the simulated average. For the predecoder it is pessimistic by about a third, mostly because it charges each k-input gate a parasitic delay of $k\,p_{inv}$, which overestimates the NAND2 and NOR2 sized here. The simulation also shows the rise/fall imbalance that an averaged estimate hides. The flat decoder's WL rise (204 ps) is twice its fall, because a selected row must pull its NAND4 output down through four stacked 2-fin NMOS. The predecoder's rise is 129 ps. The WL rise starts every access, so it is the edge that matters. **We choose the 2-2 predecoder with NOR2 rows** (Figure 9): its WL rise is 37% faster than the flat decoder's, at 31% more fins.

Body connections. Every NMOS body is tied to VSS and every PMOS body to VDD, as in a layout where each device type shares one well. A first run had the bodies of the stacked transistors tied to their own sources, which removes the body effect. That run was about 4% faster on the WL rise (124 against 129 ps for the predecoder, 196 against 204 ps for the flat decoder) and unchanged

on the fall. Only the rising edge goes through a stack (the NAND4 or NAND2 pull-down and the NOR2 pull-up), and the body effect is weak in a FinFET, so the difference is small and does not change the choice.

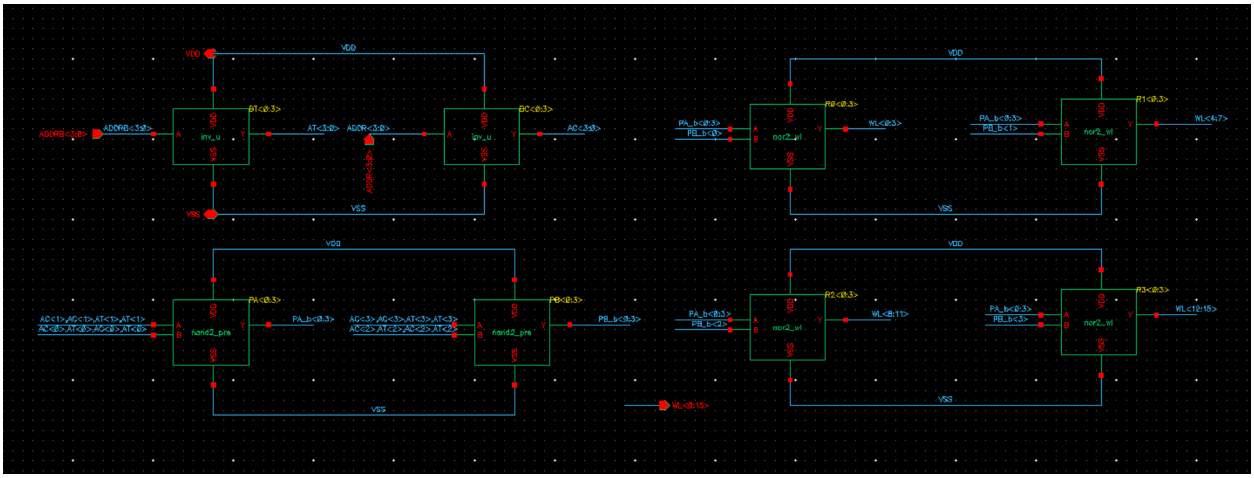


Figure 9: The chosen decoder, `dec_pre`. Top left: the literal buffers (`inv_u`), which make the true (`AT`) and complement (`AC`) literals. Bottom left: the two groups of four NAND2 predecoders, `PA` for A_1A_0 and `PB` for A_3A_2, with active-low outputs. Right: the 16 NOR2 row gates in four iterated groups, which drive the wordlines directly.

Wordline enable. The wordline-enable pulse enters the group-A predecoder (A_1, A_0) as a third input, so its four gates become NAND3s. While WLE is low, every group-A predecoded line is high and every NOR2 row output is low, so no wordline can glitch while the address changes. WLE loads only four gates. The NAND3 (`nand3_pre`) and the gated decoder (`dec_wle`) are shown in Section 4 (Figure 15). In the full macro the wordline follows WLE by 91–97 ps on the rising edge and 96–103 ps on the falling edge.

2.4 Wordline pulse widths (2.2c)

Read. The selected cell discharges one bitline with I_{cell} while the other stays at V_{DD}. The wordline must stay high until the differential reaches what the read path needs, ΔV_{SA}:

$$t_{WL,read} \geq \frac{C_{BL}\,\Delta V_{SA}}{I_{cell}} = \frac{4.72\,\text{fF}}{21.8\,\mu\text{A}}\,\Delta V_{SA} = 0.22\,\text{ps/mV} \times \Delta V_{SA}. \tag{5}$$

For a sense amplifier that needs 100 mV this would be 22 ps. Our read path is a skewed inverter (Section 3) whose effective trip point is 0.30 V, so $\Delta V_{SA} = 0.50$ V and the bound is 110 ps. I_{cell} falls as the bitline drops and the read inverter adds its own delay, so the unit test needs more: RD crosses $V_{DD}/2$ 139 ps after WL rises (Table 14), and with a 170 ps pulse the bitline ends 80 mV past the trip point. We therefore take $t_{WL,read} \geq 140$ ps, and 170 ps with margin.

Write. The wordline must stay high until both storage nodes have crossed the switching point. In the Part 1 functional check (ideal write devices, 4.0 fF bitline load, the high bitline left floating) the write settled in 131 ps. With the real write driver and $C_{BL} = 4.72$ fF, the slower storage node crosses $V_{DD}/2$ 47 ps after WL rises and settles within 10% at 69 ps (Table 12), so $t_{WL,write} \geq 70$ ps, and about 100 ps with margin. The read needs the longer pulse, so the read sets the wordline width: one pulse of about 170 ps serves both operations.

In the full macro. The controller (Section 4) gives a wordline pulse of 200–205 ps. In the macro each bitline also carries the precharge, write-driver and read-inverter devices, and the array is the extracted one, with its parasitic resistance and coupling capacitance. There the read bitline crosses the 0.30 V trip point 169–181 ps after the wordline rises, against 110 ps in the unit test. The same macro with the schematic array, which we ran first by mistake, crossed it after 110 ps, so the extra time comes from the array parasitics. The minimum read pulse in the macro is therefore about 180 ps, and the 200–205 ps pulse leaves about 21 ps; the bitline ends at 0.23–0.27 V when the wordline closes. A write completes (both storage nodes past $V_{DD}/2$) 299–304 ps after CLK, about 135 ps after the wordline rises. That includes 15–34 ps of waiting for the write driver at $T = 0.7$ ns (Section 4), so the cell itself needs about 100 ps. The read still sets the width. Figure 10 shows one read in the macro at $T_{CLK,min}$ with the quantities of the specification's diagram.

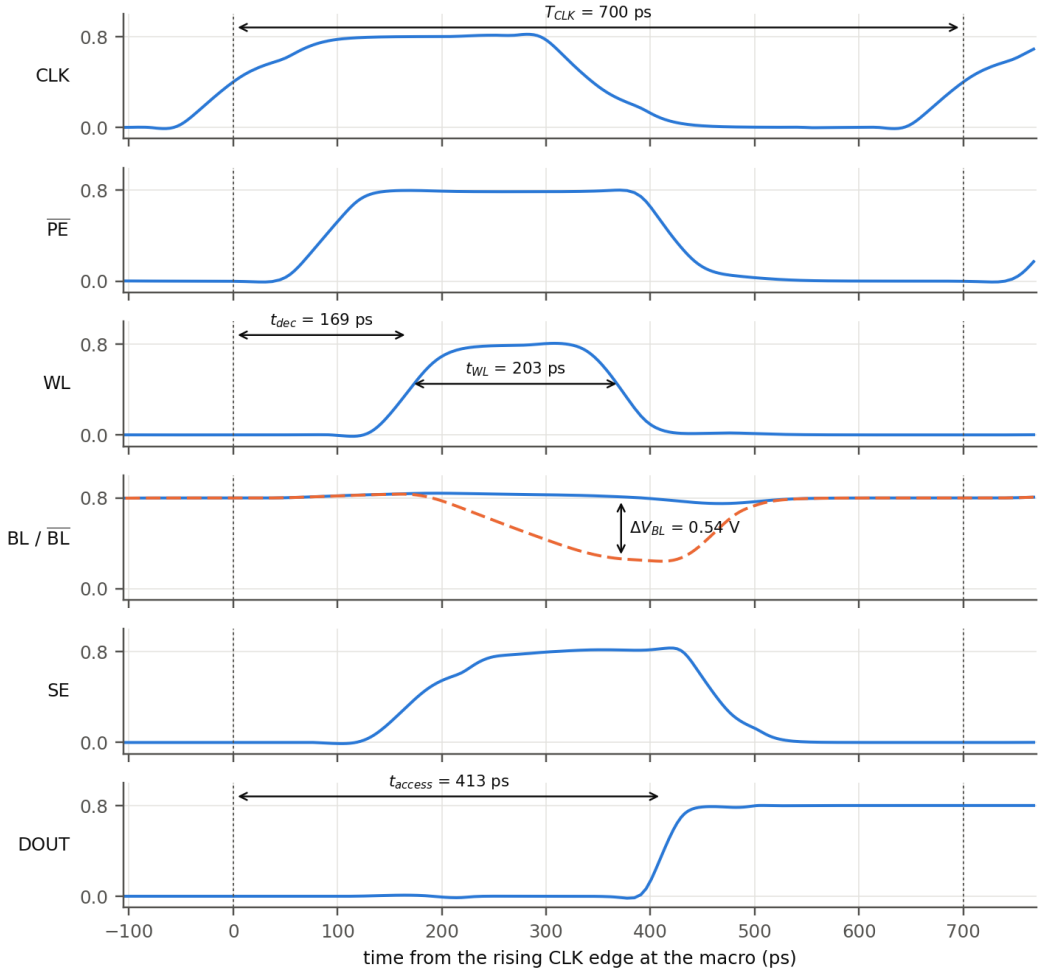


Figure 10: One read in the full macro with the extracted array, $T = 0.7$ ns (cycle 67: row 15 returns 0101 after the write of 0101). Times are from the rising CLK edge at the macro pin. The wordline rises $t_{dec} = 169$ ps after CLK and stays open for $t_{WL} = 203$ ps; $\overline{\text{BL}}\langle 0\rangle$ has swung $\Delta V_{BL} = 0.54$ V when it closes; DOUT$\langle 0\rangle$ rises at $t_{access} = 413$ ps. The precharge ($\overline{\text{PE}}$ low) starts when CLK falls, after the wordline has closed, and SE closes the output latch before the precharge restores $\overline{\text{BL}}$.

3 Column circuits (2.3)

Each column has three circuits on its bitline pair: a precharge (`pre_col`), a write driver (`wd_col`) and a read path with an output latch (`rd_col`). All three are transistor-level schematics with symbols. Each was unit-tested with the extracted worst-case bitline capacitance, $C_{BL} = 4.72\,\text{fF}$, as a capacitor on each bitline. The write-driver and read-path benches also put one `sram6t_v2` cell on the bitlines, so that they exercise a real write and a real read. Every NMOS body is tied to VSS and every PMOS body to VDD. The control inputs are $\overline{\text{PE}}$ (precharge, active low), WEN (write enable) and SE (sense enable, which opens the output latch).

Block	Devices (fins)	Unit-test result on 4.72 fF
Precharge	2 PMOS (4), PMOS equalizer (2)	BL from 0 to 98% V_{DD} in 89 ps
Write driver	2 tri-state inverters, NMOS 6, PMOS 4	write time 47 ps from WL
Read path	skewed inverter P 6 / N 2, TG latch	WL to DOUT 198 ps

Table 9: Column circuits per column.

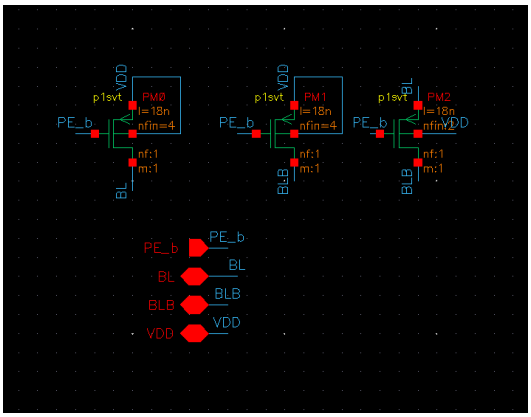
3.1 Precharge

`pre_col` has three PMOS gated by $\overline{\text{PE}}$: PM0 from VDD to BL and PM1 from VDD to $\overline{\text{BL}}$, 4 fins each, and an equalizer PM2 between BL and $\overline{\text{BL}}$, 2 fins. All three are PMOS, so all of them are off while $\overline{\text{PE}}$ is high during an access. The worst case the precharge must recover from is a write, which leaves one bitline at 0 V. The bench `tb_pre` starts from that state (initial conditions BL = 0, $\overline{\text{BL}} = 0.8\,\text{V}$) and pulls $\overline{\text{PE}}$ low at 100 ps. The bitline capacitors are a design variable, so the same bench also checks a bitline 20% larger than extracted (5.66 fF), the case 2.4 asks about.

Quantity (from $\overline{\text{PE}}$ falling)	$C_{BL} = 4.72\,\text{fF}$	5.66 fF (+20%)	Change
BL from 0 to 90% V_{DD} (ps)	65.0	75.3	+16%
BL from 0 to 98% V_{DD} (ps)	88.7	102.6	+16%
BL and $\overline{\text{BL}}$ within 10 mV (ps)	76.8	88.6	+15%
$\overline{\text{BL}}$ minimum during equalization (V)	0.731	0.732	–

Table 10: Precharge unit test.

A hand estimate for PM0 alone, with an average current of about 40 µA, gives $4.72\,\text{fF} \times 0.72\,\text{V}/40\,\mu\text{A} \approx 85\,\text{ps}$ to 90%. The simulation is faster (65 ps) because the equalizer gives BL a head start by sharing charge with $\overline{\text{BL}}$, which dips to 0.73 V and then recovers with BL. A 20% larger bitline slows the precharge by only 16%, because the 20 ps edge of $\overline{\text{PE}}$ and the PMOS's own drain capacitance do not scale with C_{BL}. Even the slower case, about 100 ps, is a small part of the clock-low phase.



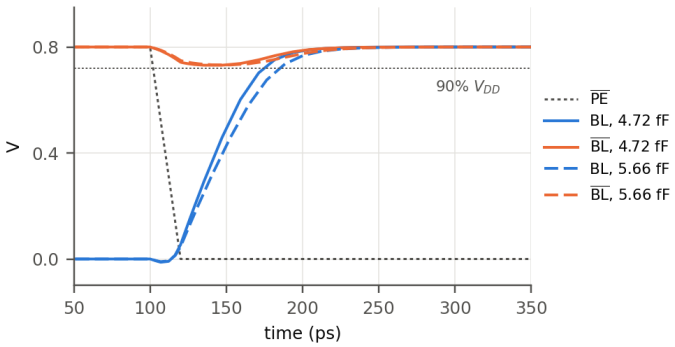


Figure 11: Left: `pre_col`. Right: `tb_pre`, BL recovering from 0 V and $\overline{\text{BL}}$ dipping during equalization, for the extracted bitline (solid) and one 20% larger (dashed).

3.2 Write driver

`wd_col` drives both bitlines actively while WEN is high: BL to DIN and $\overline{\text{BL}}$ to its complement, in both directions. In the Part 1 functional check the bitline that should stay high was left floating and sagged to about 0.54 V during the write; here it is held at V_{DD} by the driver's pull-up. While WEN is low both outputs are high-impedance, so the precharge owns the bitlines. Each output is a tri-state inverter. The enable transistors sit next to the bitline, so a disabled driver adds only one drain per device type to each bitline.

Instance	Type	Connection	Fins
ID, IE	`inv_u`	DIN → DIN_b, WEN → WEN_b	2/3
PD_BL, PE_BL	PMOS in series	VDD – (DIN_b) – (WEN_b) – BL	4, 4
NE_BL, ND_BL	NMOS in series	BL – (WEN) – (DIN_b) – VSS	6, 6
PD_BLB, PE_BLB	PMOS in series	VDD – (DIN) – (WEN_b) – $\overline{\text{BL}}$	4, 4
NE_BLB, ND_BLB	NMOS in series	$\overline{\text{BL}}$ – (WEN) – (DIN) – VSS	6, 6

Table 11: `wd_col` devices. Gate signals in parentheses.

The bench `tb_wd` starts with both bitlines precharged and the cell holding the opposite of DIN. WEN rises at 100 ps and WL at 200 ps. Run 1 writes a 0 over a stored 1; Run 2 writes a 1 over a stored 0. The cell's write trip point, 0.275 V, is the bitline level at which the cell flips (Part 1, Section 1.4c).

Quantity	Run 1 (write 0)	Run 2 (write 1)
Driven bitline below the 0.275 V trip, from WEN (ps)	48.1	48.1
Driven bitline below 0.08 V, from WEN (ps)	76.7	76.5
Faster storage node crosses $V_{DD}/2$, from WL (ps)	31.1	31.1
Write time: slower node crosses $V_{DD}/2$, from WL (ps)	**47.1**	**47.1**
Slower node within 10% of its final value, from WL (ps)	69.4	69.3
Undriven bitline minimum (V)	0.724	0.724

Table 12: Write-driver unit test.

The two runs agree to within 0.2 ps, as they should for a symmetric cell and driver. The undriven bitline sags by only 76 mV, against 260 mV with a floating bitline in Part 1, and it pulls the rising storage node up through its pass gate. That is why the write now settles in 69 ps instead of 131 ps. When WL opens, the driven bitline bumps up to about 90 mV as the cell's high node dumps charge into it, far below the trip point.

We kept NMOS 6 and PMOS 4 fins although the driven bitline takes 77 ps to reach 0.08 V. The write only needs that bitline below the trip point before WL opens, which takes 48 ps from WEN. In the macro, WEN rises with the clock and WL only after the wordline-enable delay and the decoder, so WL lags WEN by more than that. Larger drivers would also add drain capacitance to every bitline and slow every read.

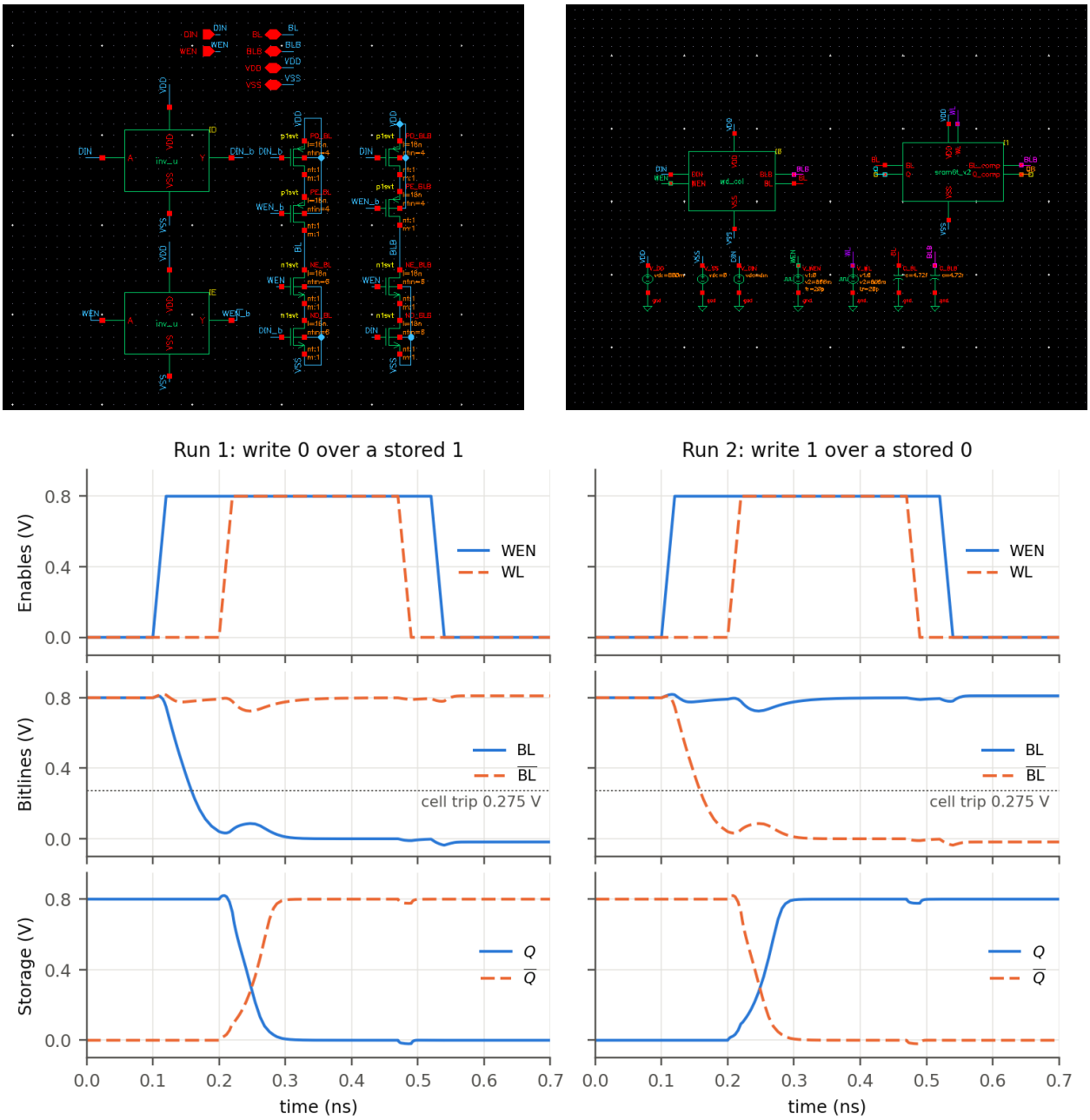


Figure 12: Top: `wd_col` and its bench `tb_wd`. Bottom: both write directions, with the cell's write trip point marked on the bitlines.

3.3 Read path and output latch

We read single-ended on $\overline{\text{BL}}$ with a skewed inverter, RI (PMOS 6 fins, NMOS 2), instead of a differential sense amplifier. When the cell holds $Q = 1$, $\overline{Q} = 0$ pulls $\overline{\text{BL}}$ down, so the inverter output RD equals Q and DOUT needs no extra inversion. We chose this circuit because it avoids the matched differential pair and offset requirement of a sense amplifier and is simpler to time. Its drawbacks are the larger bitline swing, the longer wordline pulse, and sensitivity to the inverter trip point. RD feeds a transmission-gate latch built from two `inv_u`. While SE is high, TG1 passes

RD to the latch; when SE falls, TG1 opens and TG2 closes the loop, so DOUT holds through the next precharge until the next read.

Instance	Type	Connection	Fins
RI_P, RI_N	skewed inverter	$\overline{\text{BL}} \rightarrow$ RD	P 6, N 2
IS	`inv_u`	SE $\rightarrow$ SE_b	2/3
TG1	transmission gate	RD $\leftrightarrow$ X, on while SE = 1	2/2
LA, LB	`inv_u`	X $\rightarrow$ XB $\rightarrow$ DOUT	2/3
TG2	transmission gate	DOUT $\leftrightarrow$ X, on while SE = 0	2/2

Table 13: `rd_col` devices.

The bench `tb_rd` holds both bitlines at 0.8 V initially, with DOUT at 0. WL is high from 110 to 280 ps at its 50% points and SE from 160 to 330 ps. A 4-fin PMOS stands in for the precharge on $\overline{\text{BL}}$ and turns on at 400 ps, so the bench also checks that DOUT holds once $\overline{\text{BL}}$ is restored. Run 1 reads a stored 1, so $\overline{\text{BL}}$ discharges; Run 2 reads a stored 0, so BL discharges and $\overline{\text{BL}}$ stays high.

Quantity	Run 1 (read 1)	Run 2 (read 0)
Effective trip: $\overline{\text{BL}}$ when RD crosses $V_{DD}/2$ (V)	0.299	–
WL rises to RD crosses $V_{DD}/2$ (ps)	138.9	–
WL rises to DOUT crosses $V_{DD}/2$ (ps)	198.4	–
$\overline{\text{BL}}$ when WL falls (V)	0.218	0.806
DOUT at 1.1 ns, after precharge (V)	0.800	0.000
Read disturb on the low storage node (V)	0.13	0.13

Table 14: Read-path unit test.

Required differential and margin. The read inverter switches when $\overline{\text{BL}}$ has fallen from 0.8 V to its effective trip point, 0.30 V, so the read path needs a 0.50 V **swing** on $\overline{\text{BL}}$. With the 170 ps wordline pulse of the bench, $\overline{\text{BL}}$ reaches 0.218 V before WL closes: a **margin of about** 80 mV below the trip point, and RD has crossed $V_{DD}/2$ about 30 ps before WL falls. The read disturb on the low storage node peaks at 0.13 V, the same as in Part 1, far below the cell's switching point.

Why the trip point is 0.30 V. A square-law estimate with the Part 1 device data puts the static switching point of a 6/2 inverter near 0.45 V. The measured 0.30 V is a dynamic trip point. RD starts to rise when $\overline{\text{BL}}$ is near 0.5 V, but close to its switching point the PMOS has only about 0.15 V of overdrive and must charge about 1 fF on RD (the inverter's drains, TG1 and the latch input). By the time RD reaches $V_{DD}/2$, $\overline{\text{BL}}$ has fallen to 0.30 V. Weakening the pull-down with a second, always-on NMOS in series did not help (trip 0.302 V), because near the switching point the original NMOS is barely on and the series device drops almost nothing, so we removed it. Raising the trip point substantially would need a much larger PMOS, which loads the bitline, or a sense amplifier.

Loaded bitline. In Run 2, BL carries only the cell and 4.72 fF and falls at about 4.1 mV/ps; in Run 1, $\overline{\text{BL}}$ also carries the read inverter's gates and the precharge drain and falls at about 3.4 mV/ps. The read periphery therefore adds roughly 20% to the bitline capacitance. In the

macro each bitline also carries the write driver's drains, so the read timing used for the controller (2.4) comes from the full macro, not from this unit test.

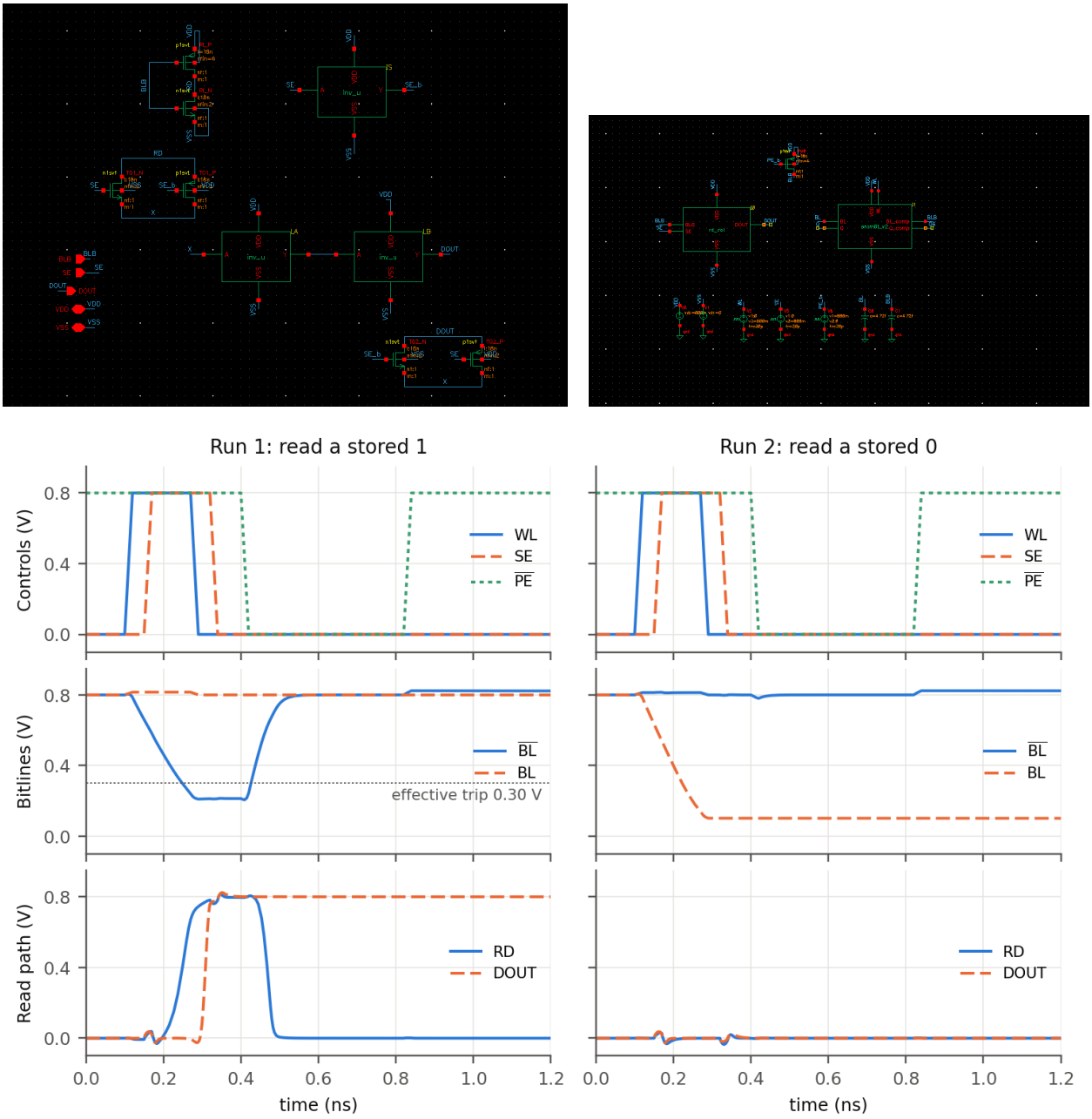


Figure 13: Top: `rd_col` and its bench `tb_rd`. Bottom: reading a stored 1 (left) and a stored 0 (right). DOUT keeps its value after SE falls and the precharge restores $\overline{\text{BL}}$.

4 Controller (2.4)

Inputs. Every input (ADDR⟨3:0⟩, WE, DIN⟨3:0⟩) passes through a D latch, `latch_lo`, that is transparent while CLK is low and holds while CLK is high, so the inputs are sampled on the rising edge and stay constant for the whole access. The latch is the same transmission-gate latch as in

rd_col with the clock phases swapped (TG1 on while CLK = 0), and it provides both polarities: the address latches give the decoder ADDR and $\overline{\text{ADDR}}$, and the WE latch gives WE_l and its complement WE_lb.

Pulse generation. The controller ctrl derives all four control signals from the rising CLK edge with a delay chain of 24 inv_u, drawn as one iterated instance CH<1:24> whose input is labelled CLK,n<1:23> and whose output is n<1:24>. Tap $n\langle k\rangle$ is CLK delayed by k inverters (true for even k, inverted for odd k), so each pulse is an AND of the clock with a later, inverted tap, and retiming a pulse only means relabelling a tap:

$$\overline{\text{PE}} = \text{CLK (buffered)}, \qquad \text{WLE} = \text{CLK} \cdot n\langle 9\rangle,$$
$$\text{WEN} = \text{WE_l} \cdot \text{CLK} \cdot n\langle 17\rangle, \qquad \text{SE} = \text{WE_lb} \cdot n\langle 2\rangle \cdot n\langle 17\rangle.$$

The precharge is off while CLK is high and on while it is low. WLE opens the wordline shortly after the edge and closes it after about 200 ps. WEN drives the write drivers on writes only; SE opens the output latch on reads only and closes it before the precharge restores the bitlines.

Signal	Gate	Driver	Load (4 columns)
$\overline{\text{PE}}$	inv_u	inv_drv_f	4.0 fF
WLE	nand2_pre (CLK, $n\langle 9\rangle$)	3 × inv_drv_f	5.6 fF
WEN	3 × nand3_c (CLK, WE_l, $n\langle 17\rangle$)	2 × inv_drv_f	6.8 fF
SE	nand3_c (WE_lb, $n\langle 2\rangle$, $n\langle 17\rangle$)	inv_drv_f	3.6 fF

Table 15: Controller gates. nand3_c is a minimum NAND3 (2 fins per transistor); parallel copies are iterated instances on the same nets.

Wordline enable in the decoder. WLE gates the group-A predecoder: in dec_wle (a copy of dec_pre) the four NAND2 of PA<0:3> become nand3_pre (NMOS 9 fins in series, PMOS 5 in parallel, the same pull-down strength as the 6-fin NAND2) with WLE as the third input. While WLE is low every $\overline{P_A}$ line is high, so every NOR2 output, and therefore every wordline, is low while the address latches are open.

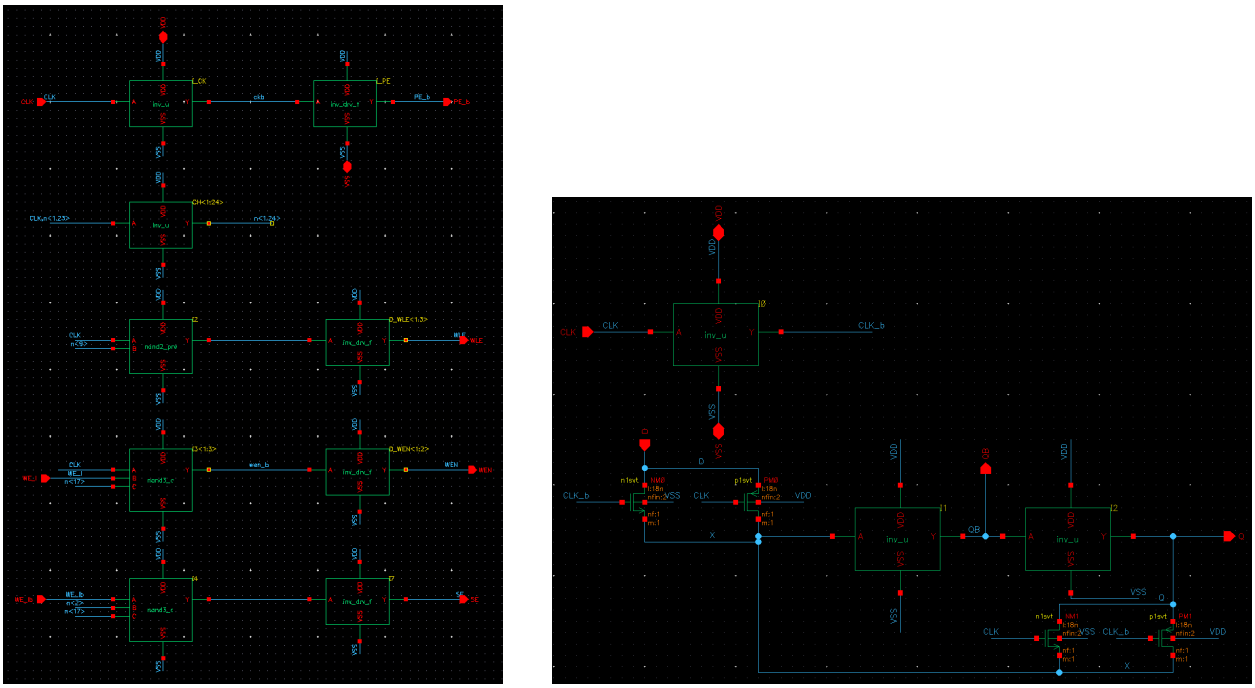


Figure 14: Left: the controller, with the 24-inverter delay chain `CH<1:24>`, the pulse gates and their drivers. Right: the input latch, transparent while CLK is low.

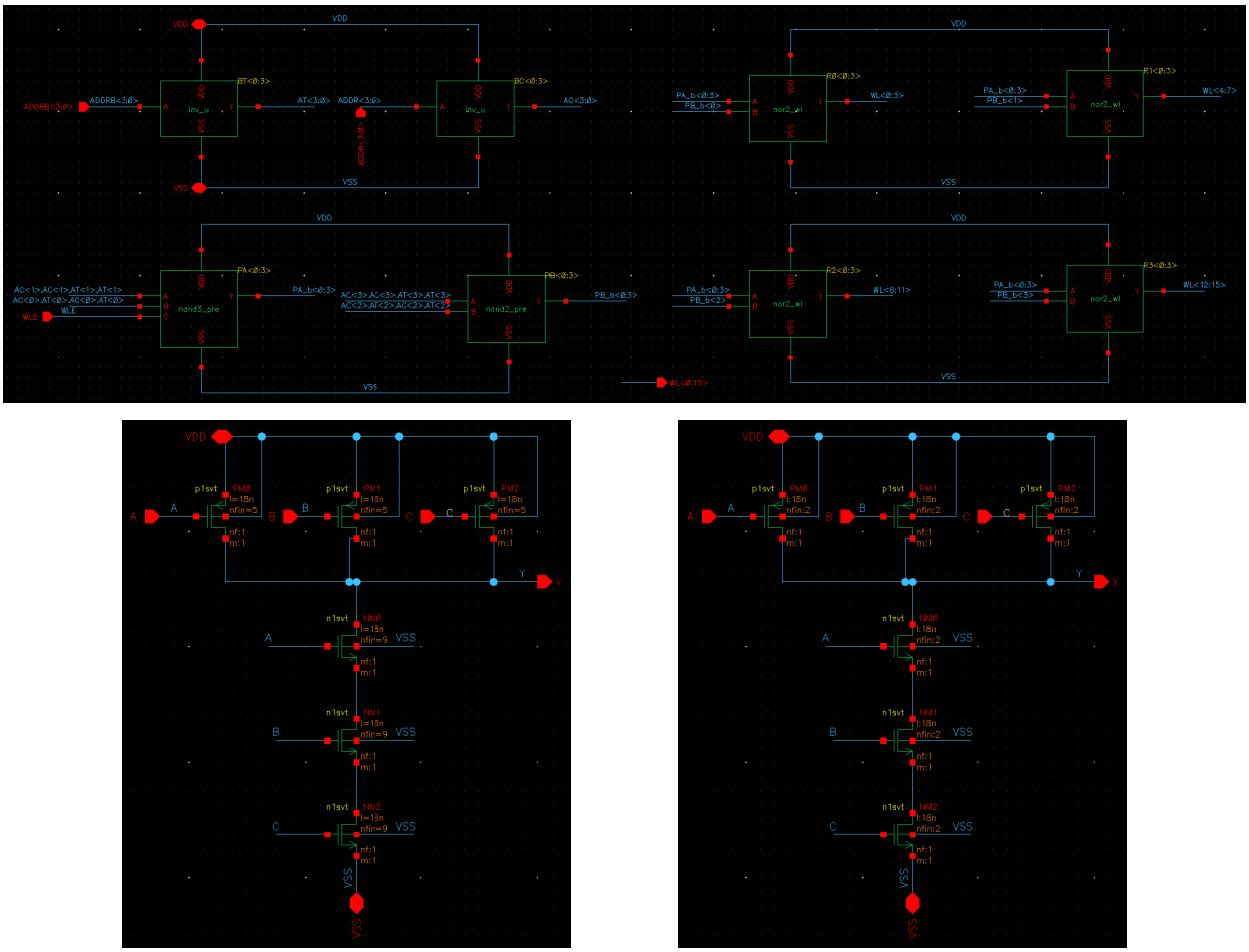


Figure 15: Top: `dec_wle`, the chosen decoder with WLE gating the group-A predecoder. Bottom left: `nand3_pre` (NMOS 9 fins in series, PMOS 5 fins in parallel). Bottom right: `nand3_c`, the minimum NAND3 used for WEN and SE in the controller.

Unit test. `tb_ctrl` drives `ctrl` with a 1 ns clock and loads each output with the capacitance of four columns (Table 15). The first version tapped $n\langle 2\rangle$, $n\langle 13\rangle$ and $n\langle 21\rangle$ with single drivers: WLE rose 135 ps after CLK, SE fell after CLK had already fallen, and WEN rose at 196 ps, too late for the write driver. Starting WLE and WEN from CLK itself, moving the taps and adding parallel drivers gave the final timing:

Edge, from CLK rising (ps)	Read (WE = 0)	Write (WE = 1)
$\overline{\text{PE}}$ rises (precharge off)	61	61
WLE rises / falls	57 / 254	57 / 254
WEN rises / falls	–	88 / 445
SE rises / falls	150 / 451	–

Table 16: `tb_ctrl`, $T = 1$ ns.

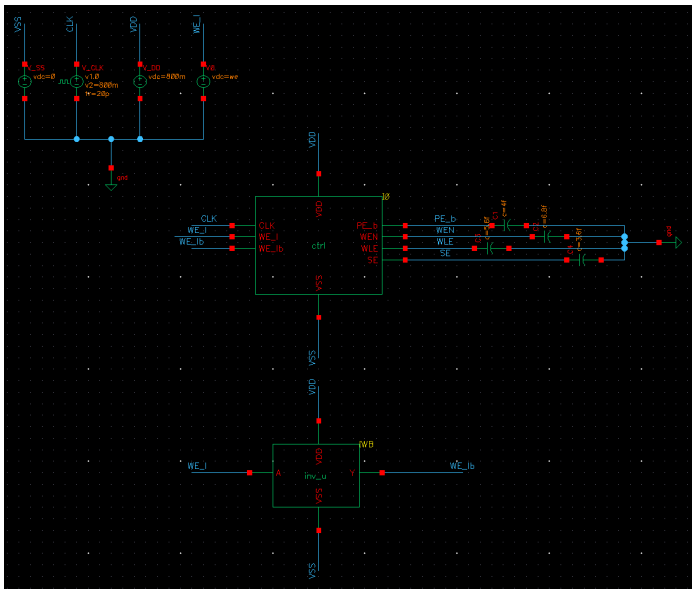


Figure 16: `tb_ctrl`: `ctrl` driven by a 1 ns clock with 20 ps edges, WE_l set by the design variable `we` (0 for a read, 1 for a write) and inverted by an `inv_u` for WE_lb, and each output loaded with the capacitance of four columns.

Timing in the macro and margins. Table 17 gives the margin on each edge, measured in the full macro with the extracted array (Section 5) at $T_{CLK,min} = 0.7$ ns. All times are taken from the rising CLK edge at the macro pin, and each margin is the worst over the run. The write time comes from the power run at the same period (Metrics).

Edge	Required	Provided	Margin
Precharge off before WL opens	$\overline{\text{PE}}$ up before WL up	89 vs 167 ps	+78 ps
WL open long enough for a read	bitline at 0.30 V: $\leq$ 181 ps after WL up	WL open 202–205 ps	+22 ps
Write completes before WL closes	both nodes past $V_{DD}/2$	304 vs 369 ps	+65 ps
Precharge starts after WL closes	$\overline{\text{PE}}$ down after WL down	415 vs 378 ps	+37 ps
DOUT valid before SE closes	DOUT at 442 ps	SE closes at 459–465 ps	+17 ps
Latch closes before RD flips back	SE down before RD down	RD falls 22–25 ps later	+22 ps
Precharge done before the next cycle	98% before the next edge	556 vs 700 ps	+144 ps
DOUT valid before the next edge	$t_{access} < T$	442 vs 700 ps	+258 ps

Table 17: Timing margins in the full macro (extracted array) at $T = 0.7$ ns.

Three points need explaining. First, the read edges are the tight ones. The extracted array discharges the bitline more slowly than the unit test (Section 2.4, "Wordline pulse widths"), so the bitline reaches the read inverter's trip point only 22 ps before the wordline closes, and DOUT is valid 17 ps before SE closes. At 0.7 ns SE closes through its $n\langle 2\rangle$ input, about 110 ps after CLK falls, so this margin shrinks with the period.

Second, the latch race: after CLK falls, the precharge pulls $\overline{\text{BL}}$ back up and the read inverter output RD falls again, and the latch must close first. In the macro RD falls 22–25 ps after SE has crossed $V_{DD}/2$, and the latch node X holding a 1 dips only to 0.70 V, well above the latch's switching point. At 0.6 ns this edge and the precharge start both fail (Section 5).

Third, at 1 ns WEN pulls the bitline below the cell's trip point within a few picoseconds of the wordline opening (2 ps before to 4 ps after), as planned. At shorter periods the end tap $n\langle 17\rangle$

(about 395 ps) is still low from the previous cycle when CLK rises, so WEN starts late: at 0.7 ns the bitline reaches the trip point 15–34 ps after the wordline opens. The write still completes 65 ps before the wordline closes.

C_{BL} 20% larger. With $C_{BL} = 1.2 \times 4.72\,\text{fF} = 5.7\,\text{fF}$ the read discharge, the write-driver pull-down and the precharge all slow down. The read is the problem. The cell current depends only on the bitline voltage, so a 20% larger capacitance stretches the bitline waveform by 20% in time. Reading our 0.7 ns waveforms at $t_{WL}/1.2$ after the wordline rises, the bitline would be at 0.30–0.32 V when the wordline closes, instead of 0.23–0.27 V: right at the read inverter's trip point, so the read would have no margin left. DOUT would also come about 35 ps later, around 477 ps, after SE closes at 459–465 ps. The other edges survive: the write driver needs about 20% longer to pull the bitline down, against 65 ps of margin; the precharge, 16% slower in `tb_pre`, would reach 98% at about 580 ps, before the next edge at 700 ps; and a slower precharge helps the latch edge, because $\overline{\text{BL}}$ recovers later. To recover the read, we would move the WLE tap from $n\langle 9\rangle$ to $n\langle 11\rangle$, about 45 ps more wordline time. The wordline would then close at about 420 ps, when the precharge starts at 0.7 ns, so the period would have to grow to about 0.8 ns. A more robust option would be a replica bitline: a dummy column with the same C_{BL} could end the wordline and open the latch after it discharges. This would make the timing track changes in C_{BL}.

5 Verification (2.5)

Test bench. `tb_macro` instantiates `sram_macro` (Figure 17) with the extracted array, through a config view that binds `sram_array` to `av_extracted`. Every input comes from a source through two inverters: `inv_u` pairs for WE, ADDR and DIN, and `inv_u` followed by two parallel `inv_drv_f` for CLK. Each DOUT bit drives 1 fF. The address, data and WE waveforms are piecewise-linear files generated by `scripts/macro_stim.py`, with 20 ps edges. They change at the start of each cycle, while CLK is low and the input latches are open; CLK rises half a cycle later. One run covers 68 cycles:

- cycles 0–15: (i) write the checkerboard, 0101 on even rows and 1010 on odd rows; cycles 16–31: read all rows;
- cycles 32–47: (ii) write the complement of every word; cycles 48–63: read all rows;
- cycles 64–67: (iii) on row 15, the row with the lowest read current (Section 1), write the opposite value and read it in the next cycle, twice.

`scripts/macro_check.py` compares DOUT, sampled 5 ps before the next CLK edge, with the expected word for every read, measures t_{access} for every read that changes DOUT, and makes Table 18 and Figure 19. As further checks, the wordline of row 15 pulses in exactly the cycles that address it, and DOUT never changes during a write cycle or a precharge. The patterns also rule out false passes: during the reads of (i) DIN is fixed while the expected DOUT alternates, and (ii) reverses the relation between row parity and data.

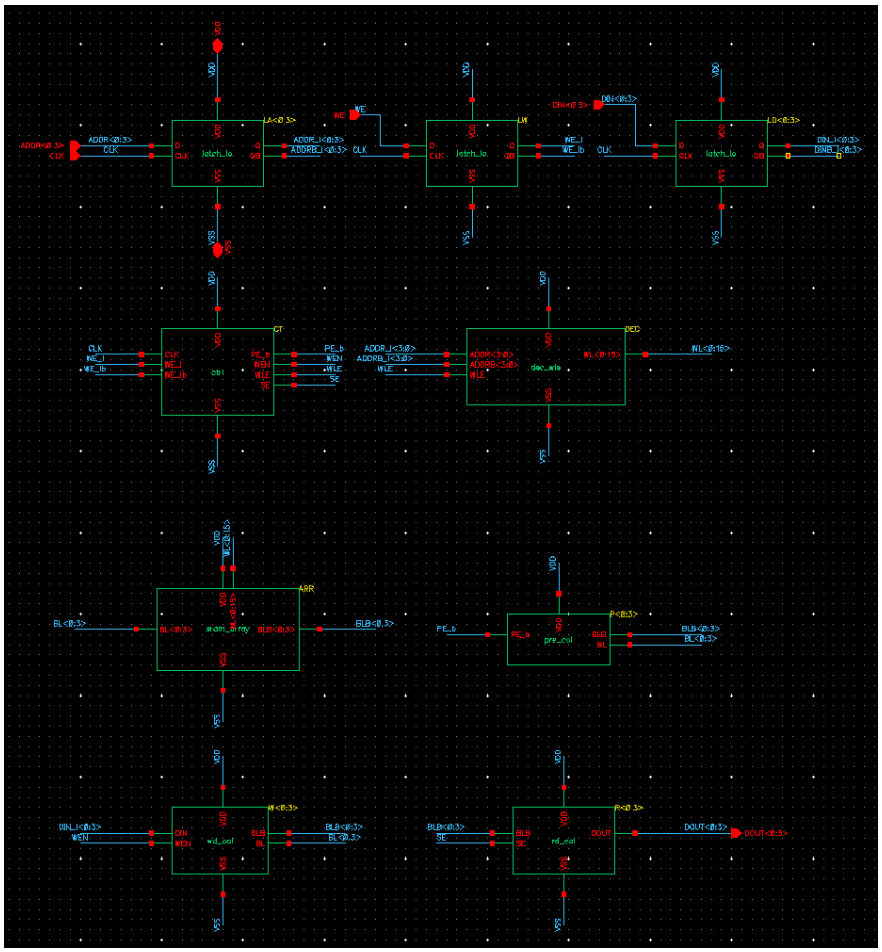


Figure 17: `sram_macro`: the input latches (ADDR, WE, DIN), the controller, the gated decoder, the array (bound to `av_extracted` by the test bench's config) and the four columns of precharge, write driver and read path.

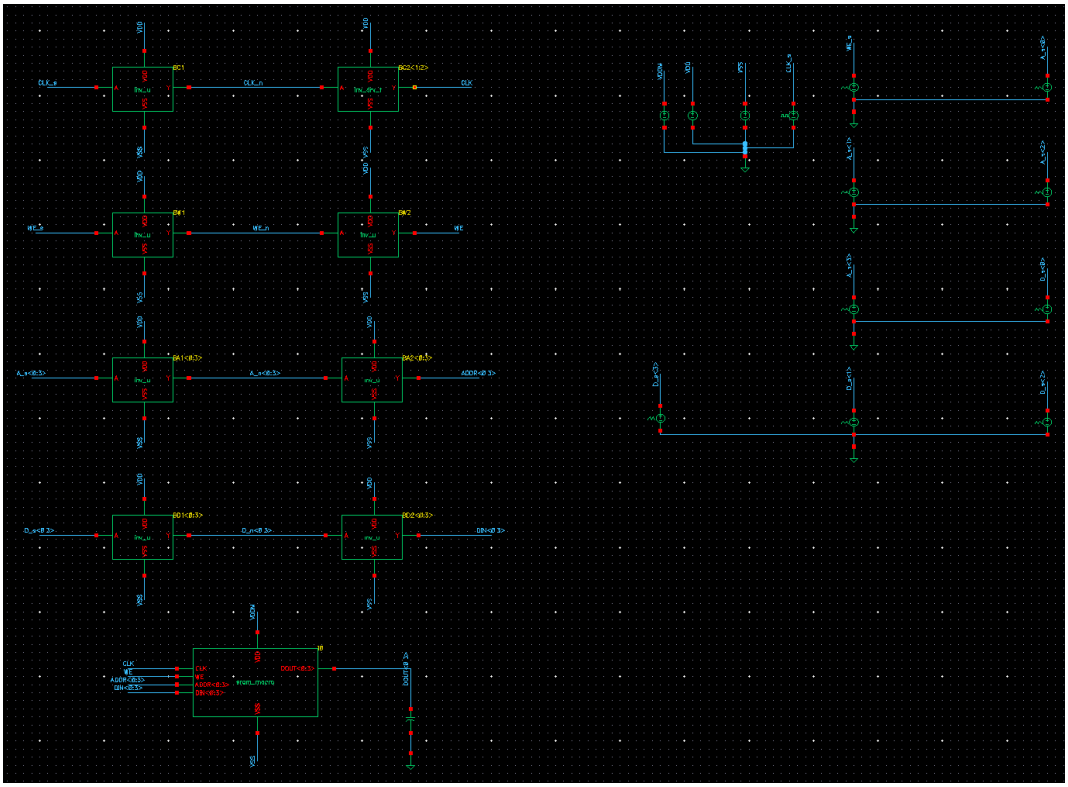


Figure 18: `tb_macro`: piecewise-linear sources through inverter buffers into the macro, which runs from its own supply `V_DDM`; each DOUT bit drives 1 fF.

Row	(i) written	(i) read	(ii) written	(ii) read	Pass
0	0101	0101	1010	1010	pass
1	1010	1010	0101	0101	pass
2	0101	0101	1010	1010	pass
3	1010	1010	0101	0101	pass
4	0101	0101	1010	1010	pass
5	1010	1010	0101	0101	pass
6	0101	0101	1010	1010	pass
7	1010	1010	0101	0101	pass
8	0101	0101	1010	1010	pass
9	1010	1010	0101	0101	pass
10	0101	0101	1010	1010	pass
11	1010	1010	0101	0101	pass
12	0101	0101	1010	1010	pass
13	1010	1010	0101	0101	pass
14	0101	0101	1010	1010	pass
15	1010	1010	0101	0101	pass

Table 18: (i) Checkerboard write and read; (ii) complement write and read, in the full macro with the extracted array at $T = 0.7$ ns. All 32 reads, and both reads of (iii), are correct.

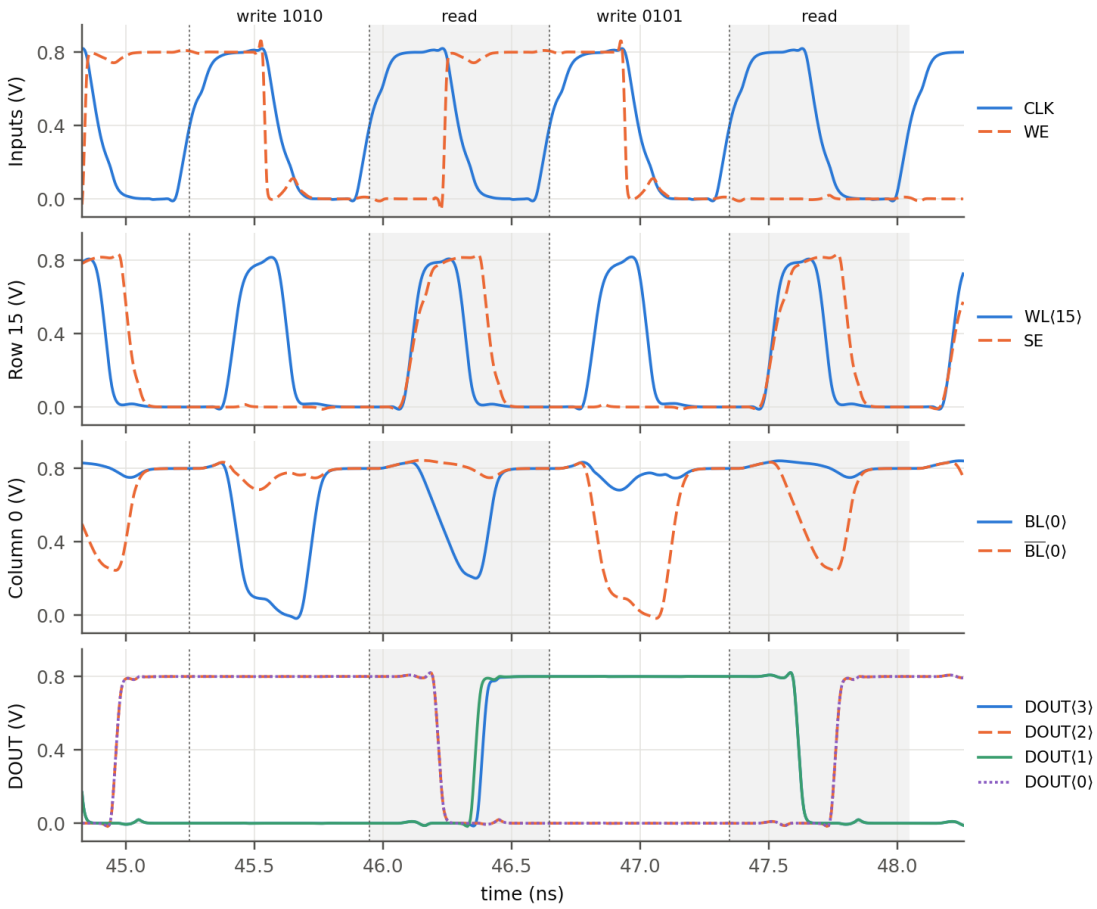


Figure 19: (iii) on row 15 in the full macro (extracted array) at $T = 0.7$ ns: write 1010, read, write 0101, read (dashed lines: CLK edges at the macro; shaded: read cycles). The driven bitline goes to 0 on writes. On reads, a DOUT bit that becomes 0 falls when SE opens the latch, about 268 ps after CLK; a bit that becomes 1 rises when its $\overline{\text{BL}}$ has discharged, 410–412 ps after CLK (439 ps for column 3). DOUT holds through the following write.

Clock period. We ran the full sequence at three periods (Table 19). Everything is timed from the rising edge, so t_{access} does not change with the period; what changes is how close the falling edge, which starts the precharge and closes SE, comes to the end of the access.

T (ns)	Reads correct	t_{access} (ps)	Latch margin (ps)	X holding a 1 (V)	Write driver vs WL
1.0	34/34	411–442	+151	0.77	2 ps before to 4 ps after
0.7	**34/34**	**411–442**	**+22**	**0.70**	15–34 ps after
0.6	27/34 (fail)	–	−3	0.50	52–73 ps after

Table 19: Full-macro runs with the extracted array at three clock periods. Latch margin: time from SE falling through $V_{DD}/2$ to RD falling through $V_{DD}/2$, column 0, worst of the 17 reads of a 1. Write driver vs WL: when the driven bitline falls below the cell's trip point, relative to the wordline rising.

We take $T_{CLK,min} = 0.7$ ns, the shortest period we ran at which every test passes, with a

positive margin on every edge (Table 17). At 0.6 ns seven of the 34 reads fail, all in the same way: a read of 1010 returns 0010, so DOUT⟨3⟩ loses its 1 (rows 7, 11, 13 and 15 in (i), rows 10, 12 and 14 in (ii)). Two edges run out together at 0.6 ns. The precharge starts at 366–376 ps, while the wordline is still closing (369–377 ps), and SE closes at 425–435 ps, before column 3 has resolved. Column 3 is the slowest column, because its sensed bitline $\overline{\text{BL}}\langle 3\rangle$ runs beside the VDD strap (Section 1): at 0.7 ns its DOUT rises at 439 ps, against 410–412 ps for the other columns. Column 0 still reads correctly at 0.6 ns, but with no margin: its latch race is −3 to +1 ps and X dips to 0.50 V.

Our first sweep was run, by mistake, with the schematic array (the test bench had no config view yet). That version passed every test down to 0.5 ns with $t_{access} = 341$–352 ps. The extracted array's bitline parasitics add about 100 ps to t_{access} and move $T_{CLK,min}$ from 0.5–0.6 ns to 0.7 ns, which is why every number in this report comes from the extracted runs.

One effect of the output latch shows up when a bit reads 1 twice in a row (cycle 48, bits 1 and 3, after cycle 31 also returned 1010). SE opens the latch while RD is still low, so DOUT drops to 0 at 268 ps and returns to 1 at 407–436 ps, within t_{access}. DOUT is valid from t_{access} until the next read, as the specification requires; the 1 is not held through the start of the next read.

Metrics and figure of merit

All metrics are measured in the full macro with the extracted array at $T_{CLK,min} = 0.7$ ns and $V_{DD} = 0.8$ V.

Metric	Value
V_{DD}	0.8 V
t_{access} (CLK to valid DOUT, worst case, column 3)	442 ps
Write time (CLK to both nodes past $V_{DD}/2$, worst case)	304 ps
$T_{CLK,min}$ (0.6 ns fails on column 3)	0.7 ns
P_{write} (all 0s then all 1s, ×4)	138.5 µW
P_{read} (checkerboard read sweep)	117.4 µW
$P_{standby}$ (CLK running, WE = 0, no enable)	119 µW
Array area (layout)	33.74 µm^2
Periphery area, $D \sum W_i L_i$	77.9 µm^2
Total area	111.6 µm^2
Density factor D	79
$\text{FOM}_{SRAM} = \text{Area} \cdot P_{write} \cdot T_{CLK,min}$	1.08×10^4 µm^2 · µW · ns

Table 20: Part 2 metrics.

Timing. t_{access} is the worst of the 34 reads in the 0.7 ns verification run (Table 19), measured from the rising CLK edge at the macro pin to the last DOUT bit crossing $V_{DD}/2$; it is set by column 3. The write time is measured on the storage nodes Q and $\overline{Q}$ of one extracted cell, from the rising CLK edge at the macro pin until both have crossed $V_{DD}/2$. Quantus names the extracted cells after the layout instances, and the probed cell, `R(15)|I(0)` in the layout, sits on WL⟨9⟩, column 0. Over its seven flips in the power run, in both directions, the write time is 299–304 ps. Every row has the same decoder path, wordline load and bitline, so we expect other rows to differ by only a few picoseconds.

Power. The macro runs from its own source, `V_DDM`, so the test-bench input buffers are not counted, and each power is $-0.8\,\mathrm{V}$ times the average current of `V_DDM` over a window of one 180-cycle run at 0.7 ns (`scripts/macro_stim.py`, `power` mode). After writing the checkerboard, the run reads all 16 rows (P_{read}, 11.2–22.4 ns), then writes 0000 to every row and 1111 to every row, four times (P_{write}, 22.4–112 ns), then idles for 20 cycles with WE = 0 and the address at 0 ($\mathrm{P}_{standby}$, 113.4–126 ns). The energy per cycle is 97 fJ for a write and 82 fJ for a read, the same as switching about 150 fF and 130 fF rail to rail. The macro has no enable input, so with the clock running "standby" still precharges and reads row 0 every cycle. Row 0 holds 1111 at that point, which discharges $\overline{\mathrm{BL}}$ and switches the read inverter in all four columns, against two in a checkerboard read, so $P_{standby}$ is about equal to P_{read}. A true standby, with the clock gated, would leave only leakage.

Periphery area. The periphery is not laid out, so its area is estimated as $A_{est} = D \sum_i W_i L_i$ over its transistors, with $D = 79$ from the Part 1 cell ($A_{cell} = 0.5271\,\mu\mathrm{m}^2$, $\sum W_i L_i = 0.0067\,\mu\mathrm{m}^2$). We use the same width convention as Part 1: W is the active height, 0.062 µm for 2 fins plus 0.048 µm for each further fin, and $L = 18\,\mathrm{nm}$. `scripts/periph_area.py` walks the netlist of `sram_macro`, skips the array, and adds up every transistor (Table 21).

Block	Devices	Fins	$\sum W_i L_i$ (µm²)	$D \sum W_i L_i$ (µm²)
Decoder with WLE gating (`dec_wle`)	120	616	0.459	36.2
Controller (`ctrl`)	92	244	0.155	12.2
Write drivers (4× `wd_col`)	48	200	0.143	11.3
Input latches (9× `latch_lo`)	90	207	0.124	9.8
Read paths (4× `rd_col`)	48	124	0.078	6.1
Precharge (4× `pre_col`)	12	40	0.027	2.2
Periphery total	410	1431	0.985	77.9

Table 21: Periphery area by block, $D = 79$.

The total area is $33.74 + 77.9 = 111.6\,\mu\mathrm{m}^2$, and the periphery is 70% of it. The figure of merit is

$$\mathrm{FOM}_{SRAM} = 111.6\ \mu\mathrm{m}^2 \times 138.5\ \mu\mathrm{W} \times 0.7\ \mathrm{ns} = 1.08 \times 10^4\ \mu\mathrm{m}^2 \cdot \mu\mathrm{W} \cdot \mathrm{ns}. \tag{6}$$

Trade-offs

Read path. We read single-ended with a skewed inverter instead of a sense amplifier. It needs no sense-enable timing against an offset and no matched pair, but it needs a 0.50 V swing on $\overline{\mathrm{BL}}$. That swing sets the wordline pulse (about 180 of the 200 ps), most of t_{access}, and the read energy, because each read nearly empties one bitline per column. A latch-type sense amplifier that resolves 100 mV would shorten the wordline pulse several times and save most of that energy, at the cost of a carefully timed SE.

Timing generation. A fixed inverter chain with taps is simple, and retiming a pulse only means relabelling a tap. It does not track the array or the period, though. The extracted bitlines left the read with 22 ps of wordline margin, a 20% larger C_{BL} would use it up, and below 0.8 ns the WEN end tap aliases onto the previous cycle so the write driver turns on after the wordline. A replica bitline would make the wordline and SE track C_{BL}.

Decoder. The 2-2 predecoder with NOR2 rows gives a 37% faster wordline rise than the flat NAND4 decoder for 31% more fins. It is still the largest periphery block, 36 µm^2 of 78 µm^2.

Array straps. Placing the supply straps inside the first and last columns saved 8% of the array area. The cost is extra capacitance on the bitline next to each strap. In column 3 that bitline is $\overline{\mathrm{BL}}\langle 3\rangle$, the one the read path senses, so column 3 is the slowest column and the first to fail at 0.6 ns. A strap beside a bitline that is not sensed would avoid this.

Standby. Without an enable input, the idle macro keeps reading row 0 and draws about as much power as a read sweep. Clock gating would remove most of this dynamic activity, leaving leakage as the main remaining power.