ESE 5700: Digital Integrated Circuits and VLSI Fundamentals

Project 1: An SRAM-Based 4-bit MAC Accelerator

Prof. Robert M. Radway

radway@seas.upenn.edu

Released: 9/17/2026 @ 10 AM ET

Due: Part 1 (The 6T Bitcell) — 10/13/2026 @ 11:59 PM ET
Due: Part 2 (The 16 × 4 SRAM) — 10/20/2026 @ 11:59 PM ET
Due: Part 3 (The MAC Unit) — 10/27/2026 @ 11:59 PM ET

Project Teams: Teams of no more than two. Each team turns in **one report per part**, with the contributions of each member clearly delineated. Everyone is responsible for understanding the design and the report in their entirety, and the instructor reserves the right to interview the students to verify this.

Points and Schedule: 15 points, in three parts of 5 points each, with separate deadlines and reports. Each part builds on the one before it. Per the course policies, a part may be turned in up to 3 days late at −1 point per day; nothing is accepted after that. Course compute is shared: start early.

Motivation

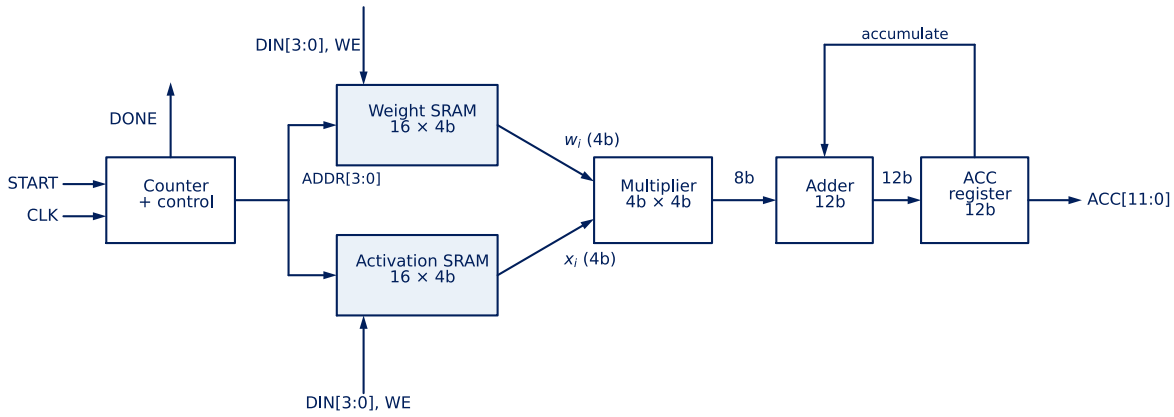
Nearly all of the arithmetic in a neural network is the dot product $y_j = \sum_i w_{ji}\, x_i$. Accelerators are therefore arrays of **multiply-accumulate (MAC)** units fed from **on-chip SRAM** holding weights and activations. Two facts set up this project:

- **Memory dominates.** The table below, from Mark Horowitz's ISSCC 2014 keynote, gives the energy of basic operations in a 45 nm process at 0.9 V. An 8-bit multiply costs 0.2 pJ; reading a 64-bit word from the smallest cache costs 10 pJ, and a DRAM access ten thousand times more. The SRAM arrays also occupy most of the die. This is why the project starts with the bitcell.
- **Small integers are enough.** Quantized inference runs on 8-, 4-, and fewer-bit integers. Integer energy falls roughly quadratically with word width, and every bit removed from the datapath is also removed from the memory.

Integer	Energy	Floating point	Energy	Memory (64-bit word)	Energy
8-bit add	0.03 pJ	16-bit FAdd	0.4 pJ	8 KB cache	10 pJ
32-bit add	0.1 pJ	32-bit FAdd	0.9 pJ	32 KB cache	20 pJ
8-bit multiply	0.2 pJ	16-bit FMult	1.1 pJ	1 MB cache	100 pJ
32-bit multiply	3.1 pJ	32-bit FMult	3.7 pJ	DRAM	1.3–2.6 nJ

Energy per operation in a 45 nm process at 0.9 V. Source: M. Horowitz, "Computing's Energy Problem (and what we can do about it)," *ISSCC* 2014, paper 1.1.

You will build **one MAC unit fed by two SRAMs**, computing a 16-element dot product of 4-bit unsigned integers.



Each SRAM holds 16 words of 4 bits. A counter walks the 16 addresses; each cycle both memories are read, the weight and activation are multiplied into an 8-bit product, and the product is added into a 12-bit accumulator ($16 \times 15 \times 15 = 3600 < 2^{12}$, so it never overflows). After 16 cycles the dot product is in the accumulator. The three parts:

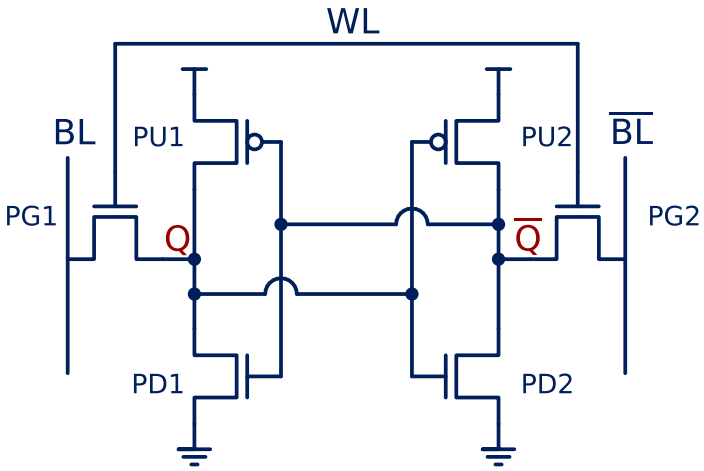
Part	Due	Deliverable
1	10/13	A 6T SRAM bitcell: schematic, custom layout, post-layout stability characterization
2	10/20	The bitcell tiled into a 16 × 4 array with its decoder, precharge, write, sense, and control circuits: a synchronous SRAM macro
3	10/27	Two SRAM macros, a multiplier, an accumulator, and a sequencer wired into a MAC unit that computes a 16-element dot product

Common Rules

- **Technology and tools.** All work is done in the Cadence GPDK 18 nm technology, at the transistor level, in Virtuoso (schematic and layout), with Spectre/ADE for simulation and the tutorial's DRC, LVS and parasitic-extraction flow. Nominal $V_{DD} = 0.8\,\text{V}$. Part 1 is characterized at 0.8 V (plus a supply sweep); in Parts 2 and 3 you may run any $V_{DD} \leq 0.8\,\text{V}$ as an optimization, as long as you state it.
- **Custom layout.** The bitcell (Part 1) and the bitcell array (Part 2) must be laid out by hand, DRC and LVS clean, and simulated from the extracted (post-layout) view. Layout of the periphery and of the MAC logic is **not** required (see the area accounting below).
- **Hierarchy and unit tests.** Every block gets its own schematic, symbol and test bench. Verify the cell before the column, the column before the array, the array before the MAC.
- **Area accounting.** Report the layout area of whatever you laid out exactly. For blocks you did not lay out, estimate area as $A_{est} = D \cdot \sum_i W_i L_i$ over that block's transistors, where the density factor $D = A_{cell} / \sum_{cell} W_i L_i$ is measured from your own bitcell layout. State D in every report that uses it.
- **Reports** are typed, one PDF per team per part, uploaded to Gradescope. Schematics must have legible transistor sizes; every number must be traceable to a simulation or calculation in the report. Report metrics to two significant figures.

Part 1: The 6T Bitcell (5 Points)

Due 10/13/2026 @ 11:59 PM ET.



Design Tasks

1.1 Sizing. Choose (W/L) for the pull-down, pull-up and pass-gate devices; cell ratio $\beta = (W/L)_{PD}/(W/L)_{PG}$, pull-up ratio $\gamma = (W/L)_{PU}/(W/L)_{PG}$. Justify with hand analysis:

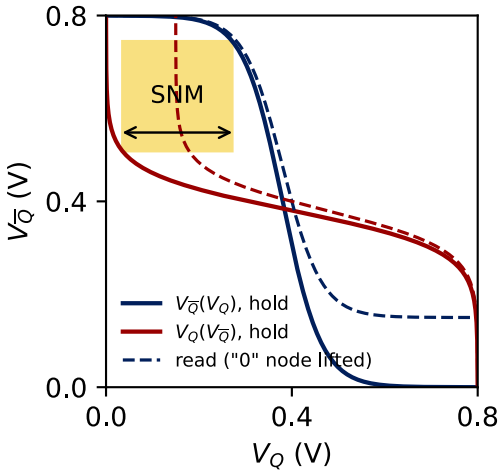
- **Read stability.** During a read the "0" node is a voltage divider between PG and PD with the bitline at V_{DD}. Estimate its voltage as a function of β and keep it safely below the switching point of the other inverter.
- **Writability.** During a write the pass gate must pull the "1" node below the switching point against the pull-up. Estimate the required γ.

Then sweep β over three values at the schematic level (e.g. minimum-width PG with PD at $1\times$, $1.5\times$, $2\times$), measuring read SNM (1.4b) for each, and pick a design point. Keep the cell symmetric.

1.2 Layout. Draw a DRC- and LVS-clean layout of the cell, wordline horizontal and bitline pair vertical, so that it tiles into the Part 2 array (abutting mirrored cells share bitline contacts and supply rails; you need not demonstrate the tiling yet). Report the cell area in μm^2 and F^2 ($F = 10\,\text{nm}$).

1.3 Extraction. Extract the layout parasitics; use the post-layout view for all measurements below. Report the extracted bitline-node and wordline-node capacitance of one cell (your per-cell loads in Part 2).

1.4 Stability characterization (post-layout, $V_{DD} = 0.8\,\text{V}$). The static noise margin (SNM) is the side of the largest square that fits between the two inverter transfer curves of the **butterfly plot**; with any asymmetry, $\text{SNM} = \min(\text{SNM}_L, \text{SNM}_R)$.



a. **Hold SNM.** WL = 0. Sweep V_Q with the feedback broken and record $V_{\overline{Q}}$; repeat for the other inverter; overlay to obtain the butterfly plot and extract SNM_{hold}.
b. **Read SNM.** WL = V_{DD}, BL = $\overline{\text{BL}} = V_{DD}$ (ideal sources). Repeat the butterfly plot and extract SNM_{read}. Report the DC voltage of the "0" node and compare to your estimate from 1.1.
c. **Write margin.** Store $Q = 1$, set WL = V_{DD}, hold BL = V_{DD}, and sweep $\overline{\text{BL}}$ from V_{DD} down to 0. The cell flips at $V_{\overline{BL}} = V_{trip}$; report $\text{WM}_{BL} = V_{trip}$ (larger is an easier write).
d. **Cell current.** Report the read current I_{cell} drawn from the bitline on the "0" side (WL = V_{DD}, bitlines at V_{DD}).
e. **Supply dependence.** Repeat (b) and (c) at $V_{DD} = 0.7$ V and 0.5 V; tabulate SNM_{read} and WM_{BL} against supply. Does the cell still read and write at 0.5 V?

1.5 Functional check. Transient test bench with ideal WL and precharge pulses and a bitline pair loaded by 15 more per-cell bitline loads (a stand-in for the Part 2 column). Write a 0, read it, write a 1, read it back. Show WL, BL, $\overline{\text{BL}}$, Q and $\overline{Q}$; the read must not disturb the cell and the write must complete.

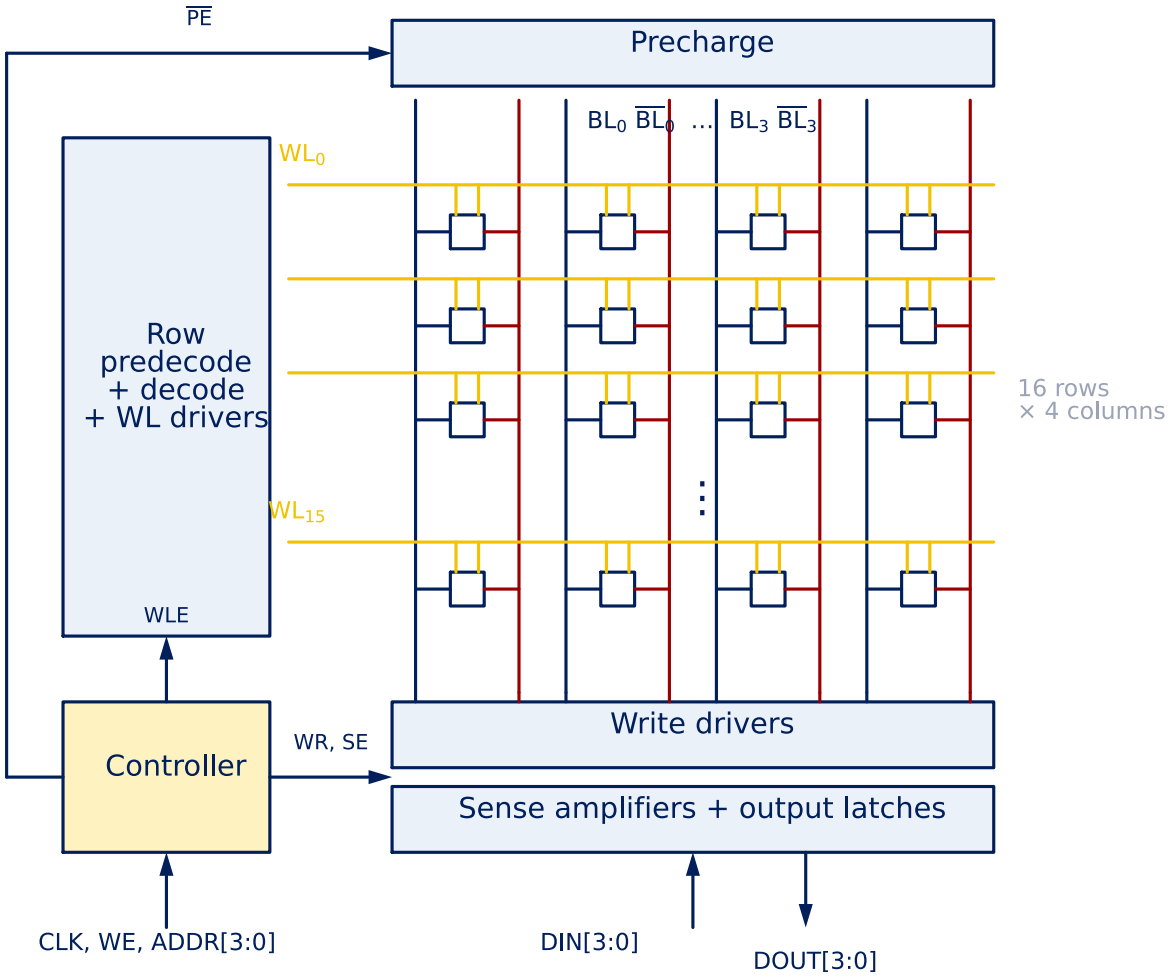
Report (Part 1)

- Team members and the contribution of each.
- Cell schematic with sizes, the hand analysis for β and γ, and the β sweep (a table is enough).
- Cell layout with DRC and LVS summaries; area in μm^2 and F^2.
- Extracted bitline and wordline node capacitances.
- Butterfly plots (hold and read), write-margin sweep, I_{cell}, and the supply table.
- Functional-check waveforms.
- A short paragraph: what limits the cell, and what you would change for Part 2.

Grading (5 points): sizing rationale and β sweep 1; layout, DRC/LVS and area 1.5; stability characterization (1.4) 2; functional check and report quality 0.5.

Part 2: The 16 × 4 SRAM Macro (5 Points)

Due 10/20/2026 @ 11:59 PM ET. A synchronous SRAM of 16 words × 4 bits with random read and write access, instantiated twice in Part 3.



Specification

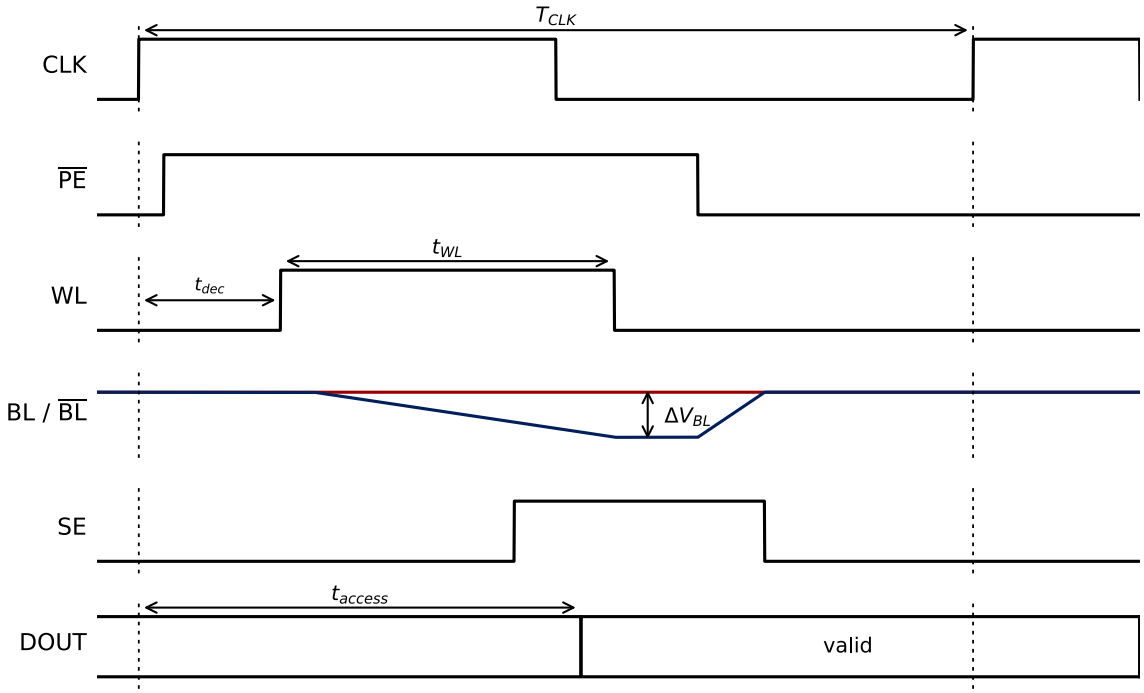
- **Interface:** `CLK`, `WE` (1 = write, 0 = read), `ADDR[3:0]`, `DIN[3:0]`, `DOUT[3:0]`. Inputs are sampled on the rising edge of `CLK`. A write completes within the cycle. Read data is valid on `DOUT` a delay t_{access} after the edge and is latched until the next read completes (the MAC in Part 3 consumes it on the following edge).
- **Capacity:** 16 words × 4 bits, one word per cycle. All four columns are accessed together; no column multiplexer.
- **Cell:** your Part 1 cell. If you change it, re-run the Part 1 characterization.

Design Tasks

2.1 Array layout and extraction. Tile the cell into the 16×4 array in layout (mirroring rows and columns to share contacts and taps), DRC and LVS clean, and extract it. Report the bitline capacitance C_{BL} per column and wordline capacitance C_{WL} per row, and compare each to $16\times$ (or $4\times$) the Part 1 per-cell values plus a wire estimate.

2.2 Row decoder and wordline timing. The decoder turns `ADDR[3:0]` into one of 16 wordlines, gated by the controller's wordline-enable pulse.

a. Compute the path effort from an address input to a wordline: electrical effort H from the address-input capacitance to C_{WL} plus the WL driver, and branching effort B (each address literal fans out to 8 of the 16 row gates). Find the optimal number of stages and the stage effort.
b. Compare at least two implementations, e.g. a flat 4-input NAND per row versus a 2-2 predecode with 2-input NAND rows, each with sized wordline drivers. Predict delays with logical effort, then measure by simulation (schematic decoder driving the extracted array). Pick one.
c. **Timing.** The wordline must rise after precharge releases the bitlines and fall once enough differential has developed (read) or the cell has flipped (write). Derive the minimum wordline pulse width for a read (from I_{cell}, C_{BL} and the sense amplifier's required differential) and for a write (from the cell's write time). Annotate the timing diagram below with your numbers.



2.3 Column circuits. Design the precharge, write driver, and read path for each column. The read path may be full-swing (an inverter on the bitline) or a sense amplifier (differential pair or clocked latch) that resolves a small differential; state the differential it needs and the margin you give it. Include an output latch so `DOUT` holds. Unit-test each block with the extracted C_{BL} as load.

2.4 Controller. From `CLK`, `WE` and `ADDR`, generate the precharge, wordline-enable, write-enable and sense-enable pulses with the ordering and widths from 2.2c, via delay chains or a self-timed replica. Show the margin on each edge and how you would recover it if C_{BL} were 20% larger than extracted.

2.5 Verification. Full-macro test: (i) write all 16 words with a checkerboard (`0101`, `1010`, alternating by row) and read every word back; (ii) overwrite every word with its complement and read back; (iii) for the worst-case row, read immediately after a write to the opposite value. Show waveforms for (iii) and a pass/fail table for (i)–(ii). Drive inputs through buffers of realistic strength, not ideal zero-rise-time sources.

Metrics and Figure of Merit

At the supply you chose:

- **Delay:** worst-case read access time t_{access} (rising `CLK` edge to valid `DOUT`) and worst-case write time (rising edge to both storage nodes past their switching point). $T_{CLK,min}$ is the shortest period at which every test in 2.5 passes, including the next cycle's precharge.
- **Power:** average power while writing the entire array with 0s, then with 1s, the pair repeated four times, at $T_{CLK,min}$. Also the standby power (`CLK` running, no access) and the average power of the checkerboard read sweep.
- **Area:** the array's layout area, plus the periphery estimated with your density factor D. Report both terms and their sum.
- $\mathbf{FOM}_{SRAM} = \text{Area} \cdot \text{Power}_{write} \cdot T_{CLK,min}$, with area in μm^2, power in μ W and period in ns.

Report (Part 2)

- Team members and contributions.
- Array layout with DRC/LVS summaries; extracted C_{BL}, C_{WL} versus the Part 1 estimate.
- Decoder: logical-effort analysis, alternatives compared (predicted vs. measured delay), chosen implementation with sizes.
- Timing: derived wordline pulse widths, annotated timing diagram, controller schematic, margin analysis.
- Column circuits with sizes and unit tests.
- Verification waveforms and pass/fail table.
- Metrics table with FOM_{SRAM}, D and V_{DD}.
- A short paragraph on the trade-offs made.

Grading (5 points): array layout, extraction and C_{BL}/C_{WL} analysis 1; decoder design and logical-effort analysis 1.5; controller timing and column circuits 1.5; verification, metrics and report quality 1.

Part 3: The MAC Unit (5 Points)

Due 10/27/2026 @ 11:59 PM ET. Two copies of the Part 2 macro, as weight memory W and activation memory X, plus the datapath and sequencer.

Specification

The unit computes $\text{ACC} = \sum_{i=0}^{15} W[i] \cdot X[i]$ for 4-bit unsigned $W[i]$ and $X[i]$, producing a 12-bit unsigned result.

- **Interface:** `CLK`, `RST` (synchronous reset of the accumulator, counter and controller), `START`, `DONE`, `ACC[11:0]`, plus a load port shared by both memories: `ADDR[3:0]`, `DIN[3:0]`, `WE_W`, `WE_X`.
- **Load phase:** while `START` is low, the external port writes words into W (`WE_W` $= 1$) or X (`WE_X` $= 1$) at `ADDR`, one word per cycle.
- **Compute phase:** on a rising `CLK` edge with `START` high, the accumulator clears and the counter begins at address 0. On each subsequent cycle the unit reads $W[i]$ and $X[i]$, multiplies them, and adds the 8-bit product into the accumulator. When all 16 products have been accumulated, `DONE` goes high and `ACC` holds the result until the next `START` or `RST`.
- **Pipelining** is allowed (read in cycle i, multiply-accumulate in cycle $i+1$); report cycles per dot product and clock period. A dot product must take no more than 20 cycles from `START` to `DONE`.

Design Tasks

3.1 Sequential elements. Design a register bit at the transistor level and use it for the accumulator, address counter, controller state and `DONE`. Characterize setup time, hold time and clock-to-Q delay (show the simulation that determines each), clock-input capacitance and energy per clock. Justify the style.

3.2 Datapath. Design the 4×4 unsigned multiplier (AND array plus adders, carry-save tree, or another structure) and the 12-bit accumulator adder. Any adder architecture and any logic discipline from the course is allowed if justified. Size the critical path with logical effort before simulating.

3.3 Sequencer. Design the 4-bit address counter and the control FSM (idle, compute, done) with your registers. Show the state diagram and the timing of `START`, counter, memory reads, accumulator update and `DONE` over one dot product.

3.4 Timing closure. Identify the critical path: clock edge, SRAM access, multiplier, adder, accumulator setup (or, if pipelined, the longer of the memory-read and multiply-accumulate paths). Compute the minimum clock period by hand from t_{access}, the logical-effort estimate of the datapath, and the register's clock-to-Q and setup times; then measure it as the shortest period at which the tests below pass. Check hold at the fast paths (e.g. counter to address inputs).

3.5 Verification. Run the three test vectors below. Show `ACC` and `DONE` waveforms for Test A and a pass/fail table for all three.

Test	$W[0..15]$	$X[0..15]$	Expected ACC
A	3, 12, 7, 0, 15, 9, 1, 14, 6, 11, 2, 13, 8, 5, 10, 4	9, 2, 15, 7, 4, 12, 0, 11, 13, 1, 8, 6, 3, 14, 5, 10	845 (`0x34D`)
B	all 15	all 15	3600 (`0xE10`)
C	5, 10, 5, 10, …	10, 5, 10, 5, …	800 (`0x320`)

Test B is the full-scale case; Test C maximizes toggling on the memory outputs and multiplier inputs.

Metrics and Figure of Merit

At the supply you chose, report:

- **Delay:** $T_{CLK,min}$ and the dot-product latency $t_{dot} = N_{cycles} \cdot T_{CLK,min}$ from `START` to `DONE`.
- **Energy:** E_{dot}, the supply energy during the compute phase of Test A at $T_{CLK,min}$ (integrate supply current from `START` to `DONE`). Report Tests B and C as well, and break Test A into memory, datapath and sequencer if you use separate supplies.
- **Area:** two SRAM macro areas from Part 2, plus the datapath and sequencer estimated with your density factor D.
- $\mathbf{FOM}_{MAC} = \text{Area} \cdot E_{dot} \cdot t_{dot}$, with area in μm^2, energy in pJ and latency in ns.

Report (Part 3)

- Team members and contributions.
- Top-level block diagram and schematic; register cell with its characterization.
- Multiplier and adder schematics with sizes; logical-effort sizing of the critical path.
- State diagram, sequencer schematic, one-dot-product timing diagram.
- Timing closure: hand-computed critical path versus measured $T_{CLK,min}$; hold check.
- Verification waveforms (Test A) and pass/fail table.
- Metrics table with FOM_{MAC}, D and V_{DD}; energy per MAC with and without the memory accesses, and a paragraph on where the energy goes.
- A short paragraph: what limited the design.

Grading (5 points): register design and characterization 1; datapath design and sizing 1.5; sequencer and timing closure 1.5; verification, metrics and report quality 1.

Where Do I Start?

Part 1. The cell is two tutorial inverters plus two pass gates. Sketch the layout on paper first: which layer carries WL and which the bitlines, where the shared contacts fall, where the well tap goes. Build the butterfly-plot test bench at the schematic level and finish the β sweep before drawing polygons.

Part 2. Extract the array first; C_{BL} and C_{WL} are the inputs to everything else. Work one column with ideal control pulses, then add the decoder with ideal enables, then the controller. What must be true for a write to land only in the addressed word?

Part 3. Treat the SRAM as a black box with the interface and t_{access} you measured. Get the register right first, then the multiplier with an exhaustive test (all 256 input pairs), then the accumulator, then the sequencer. Simulate the full unit only when every piece has passed on its own.