High Speed Circuits That Work:

The Road From Ideas to Products

EE 371 Lecture 15

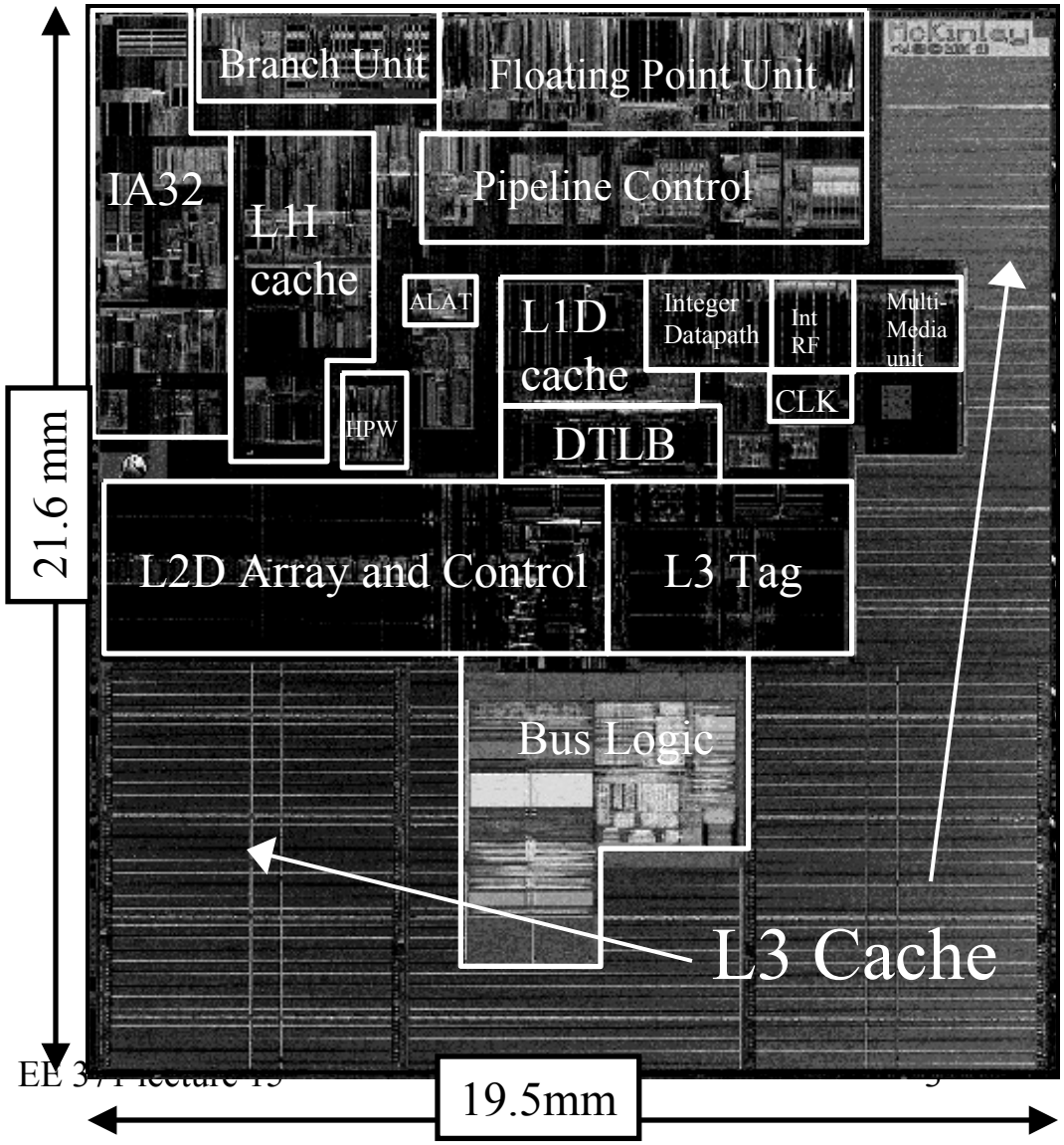
Sam Naffziger
Hewlett Packard Co.

Overview

- Goal: Provide some insight into the design process for high speed circuits taken from concept to customer ship
- I'll use a block on the soon-to-ship McKinley (Itanium® 2) microprocessor as a case study
  - What motivated and supported some of the circuit innovation ($$)
  - What the design process involved (blood, sweat and tears)
  - Some of the issues that arose (weeping and gnashing of teeth)
  - How the silicon actually behaved (very un-SPICE-like)
- The 1st level data cache (L1D) is a great example
  - 16KB, 4 way set associative, physically tagged, 1 cycle load to use (0 cycle penalty)
- But first, some background on the microprocessor

McKinley Overview

- .18µm bulk, 6 layer Al process
- 8 stage, fully stalled in-order pipeline
- Symmetric six integer-issue design
- IA32 execution engine integrated
- 3 levels of cache on-die totaling 3.3MB
- 221 Million transistors
- ECC / parity protection on all memory arrays
- 130W @1GHz, 1.5V



McKinley vs. other 64b .18µm processors

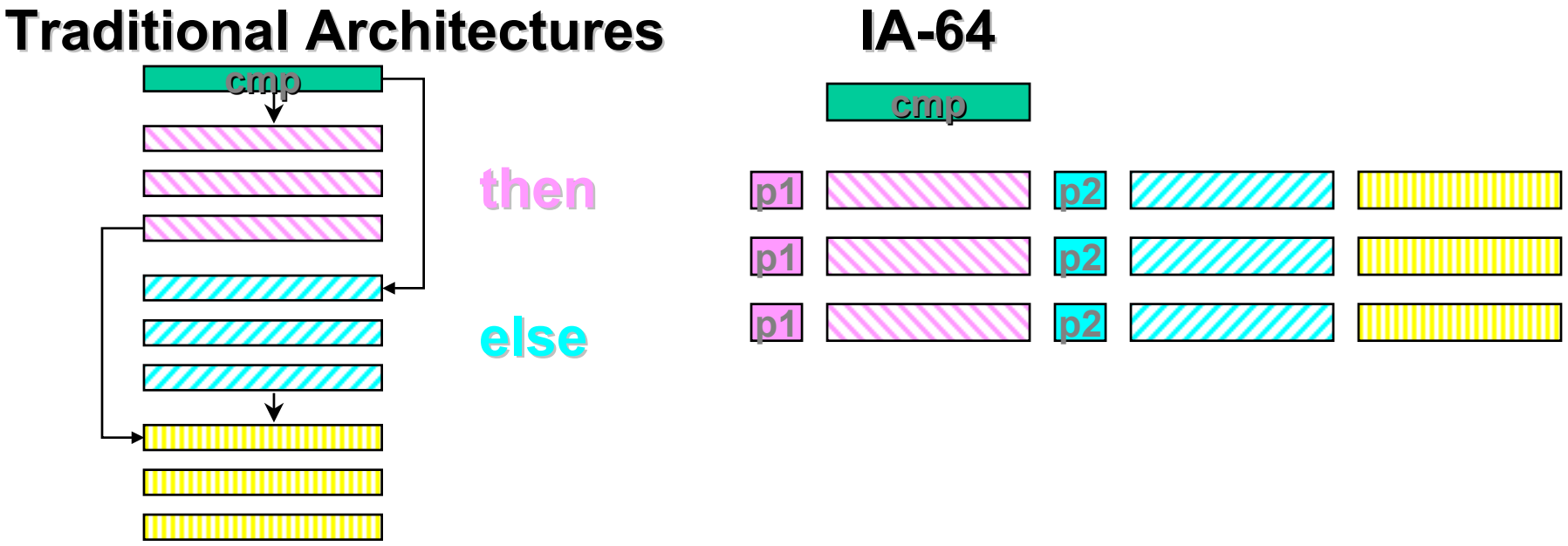
(sources provided in final slide)

	McKinley	USIII	EV7	Power 4 (2 cores)
Frequency (GHz)	1.0	1.05	1.0-1.2	1.3
Pipe stages (mpb)	**8 (6)**	14 (8)	9 (~11)	12 (~11)
Sustainable Int BW	**6 / cycle**	3 / cycle	4/cycle	3/cycle
FP units	2/cycle	2/cycle	2/cycle	2/cycle
On chip cache	**3.3MB 4 arrays**	96KB 2 arrays	1.8MB 3 arrays	1.7MB 2.5 per core
D Cache read BW	**64GB/s**	16.8	19.2	41.6 / core
Die size (mm^2)	**421**	244	397	400 est.
Core size (no IO, only lowest level caches)	142	206	115	100 est.
Power (Watts)	**130**	75	125	125

Background

- Memory access is a very important aspect of processor design
  - Bandwidth must support full instruction streaming
  - Latency must be low enough to minimize pipeline stalls
    - Contemporary processors have a 1-2 cycle load to use penalty for the 1st level cache. I.e. **LD addr→A** followed by **Add A,B** incurs a stall
- Lately, most microprocessors have embraced out of order execution (PA, Power4, Alpha, Pentium)
  - Can do useful work even if the main pipe is stalled
  - Can significantly mitigate cache latency impacts
    - Witness the PA8700 with a 1.5MB *first level* data cache. Latency is 3 cycles, but more than offset by the benefits of the large cache
- BUT, the Itanium architecture is resistant to out of order implementations
  - The whole template based instruction format
  - Worse, ubiquitous predicates – any instruction can be predicated off

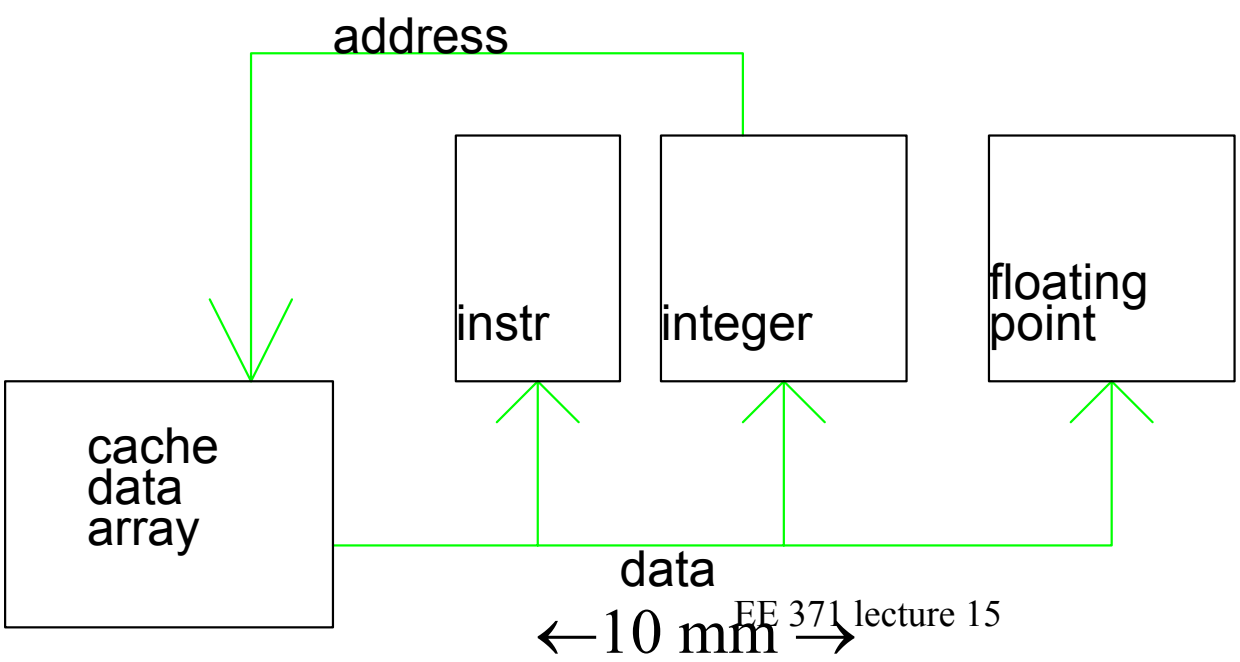
Background: Predication

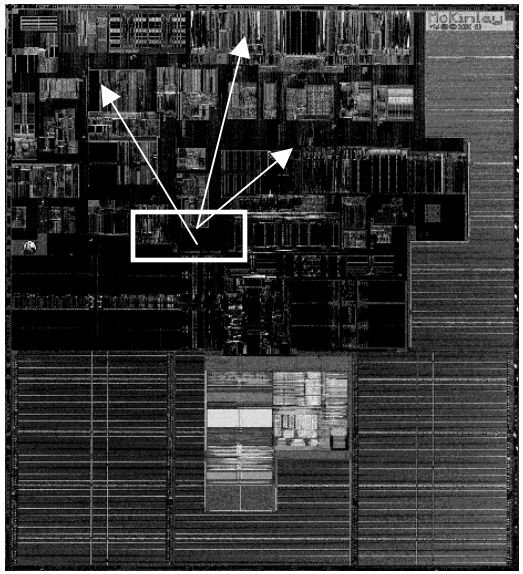


- Removes branches, converts to predicated execution
  - Executes multiple paths simultaneously
- Increases performance by exposing parallelism and reducing critical path
  - Better utilization of wider machines
  - Reduces mispredicted branches

Background

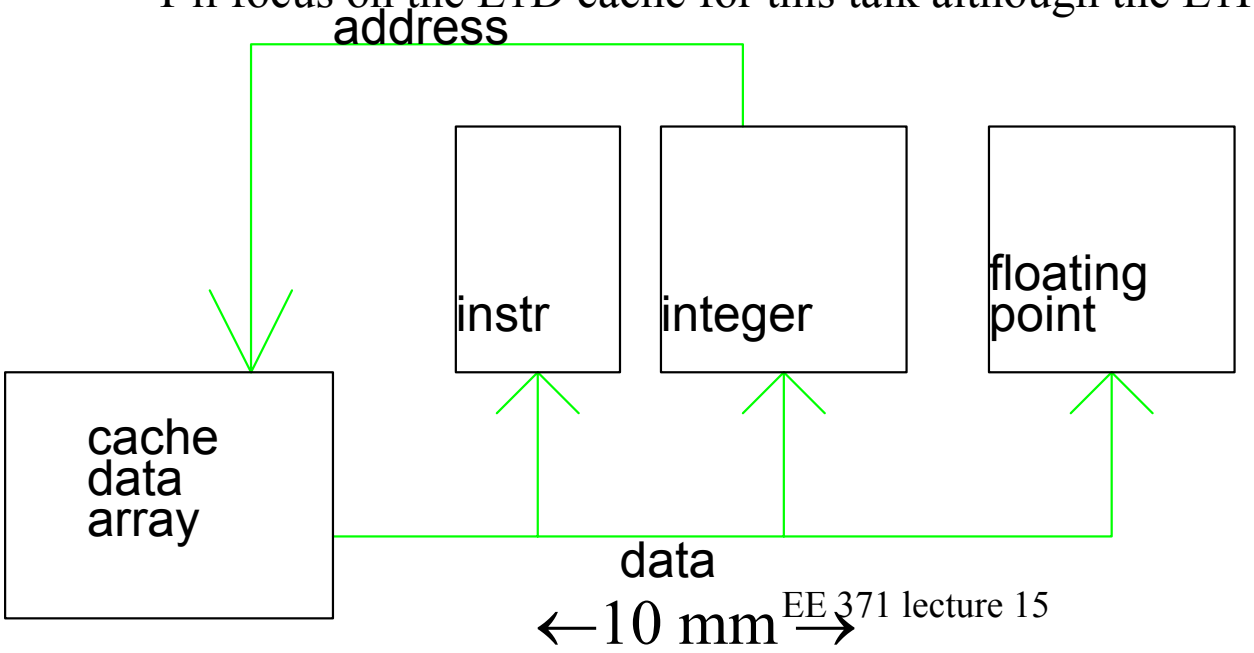
- Customers don't care about my chip's architectural handicaps, they just pay for performance (and cause my company to pay for my lab!)
- So, if we can't use out of order execution to help with the memory latency problem *and* we still need a GHz+ processor design, the only option is to innovate
  - A six-issue machine makes the load-to-use penalty problem even *more* acute⇒ A single cycle cache would be big win
- The first big barrier was the distance issue

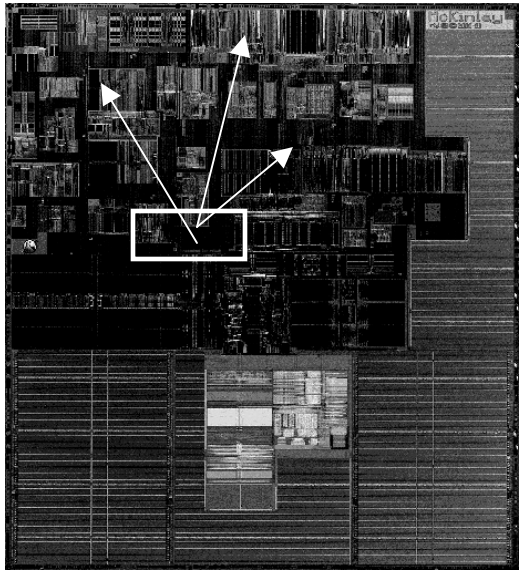




Background

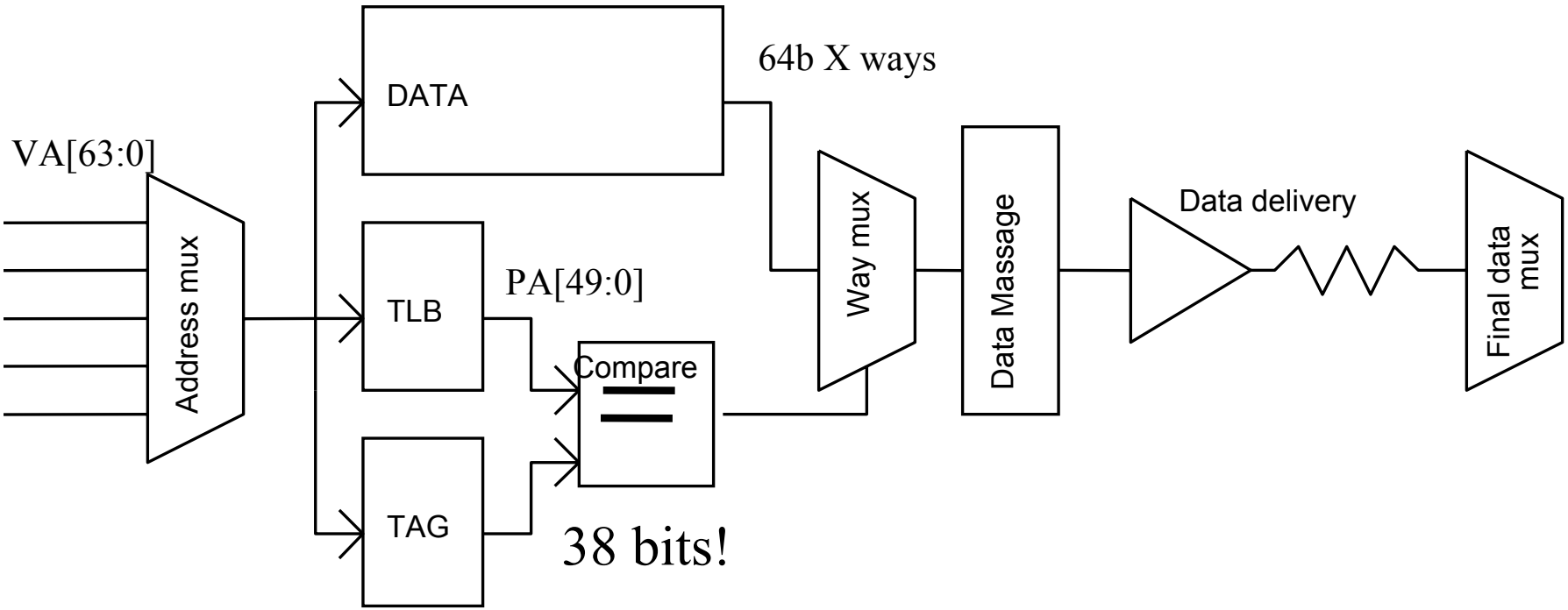
- To be truly single cycle, a cache must *receive a new address* and *deliver the dependent data* in one cycle
  - If our best repeated interconnect (for a bunch of 64b busses) is 50ps/mm, the situation in our floorplan means our *whole cycle* is consumed by wire!
- Solution: Split the caches into multiple independent arrays
  - First level instruction cache, data cache, with the floating point going directly to the larger, second level cache
  - Tight coupling between address source and data delivery
- I'll focus on the L1D cache for this talk although the L1I is similar





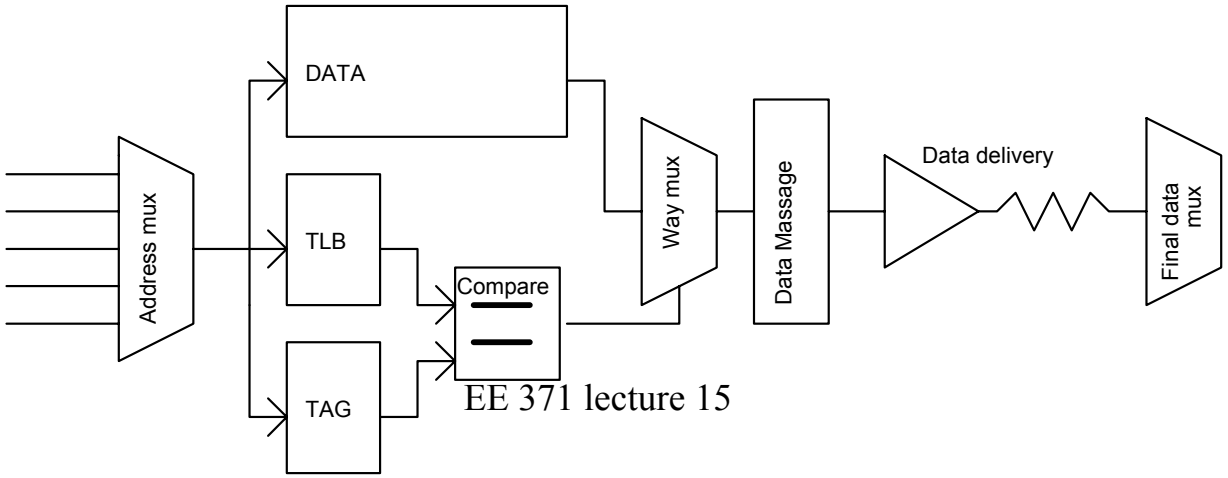
Cache Access Timing

- High speed caches usually take the following form (only for latency insensitive or very power sensitive designs does the data array access wait on the tag)
- If a cycle time is only 20-24 FO4 inverters, this is *a lot* to get done
  - The TLB access is a problem – wide CAM + RAM access
  - 38 bits is a lot to compare
  - Just the way mux, data massage and deliver is generally a full cycle …



Cache Access Timing

- So, I set out to attack the following areas:
  - TLB access + TAG compare
    - 76 bit CAM, followed by a RAM access followed by a 38b compare
    - Then buffer up the compare result to drive the way multiplexors for a 64b datapath
  - Way mux + data massage (this is *way* harder than you might think) + delivery
    - Must deliver either big or little endian data, must zero extend sub-word loads, must align subword loads, must bypass store data in and merge it with load data!
  - Data array access – speed enables a larger cache
    - Timing should match TLB/TAG plus compare
- With only a single cycle to work with, I only get *one* clock edge!
  - If I use the falling edge clock, I'm victimized by phase asymmetries, as well as clock skew which has its impact doubled for phase budgets (bad)
  - Everything must be either self-timed or dataflow => domino

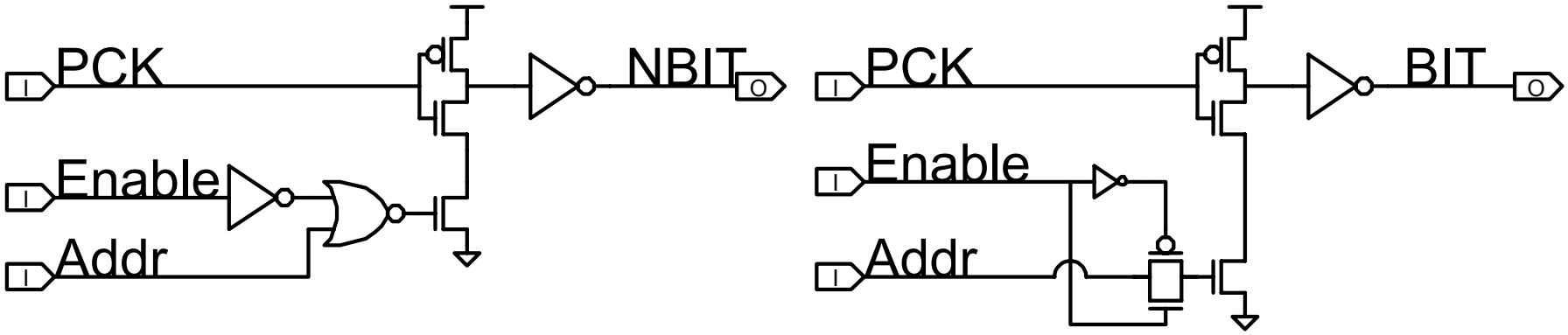


Level 1 TLB Overview

- Takes a 64b Virtual address in, compares to stored translations and generates a physical translation
- 32 Entries, each of which stores 115 bits
  - 52 virtual address bits
  - 24 Region ID bits
  - 38 physical address bits
  - 1 Valid bit
- Fixed 4 kB page size
- 2 independent CAM ports, each with shared CAM lines for both virtual and physical address CAMs
  - This saves wires in the critical dimension
- 2-level dynamic CAM with a self timed path to create a monotonic-rising match signal.

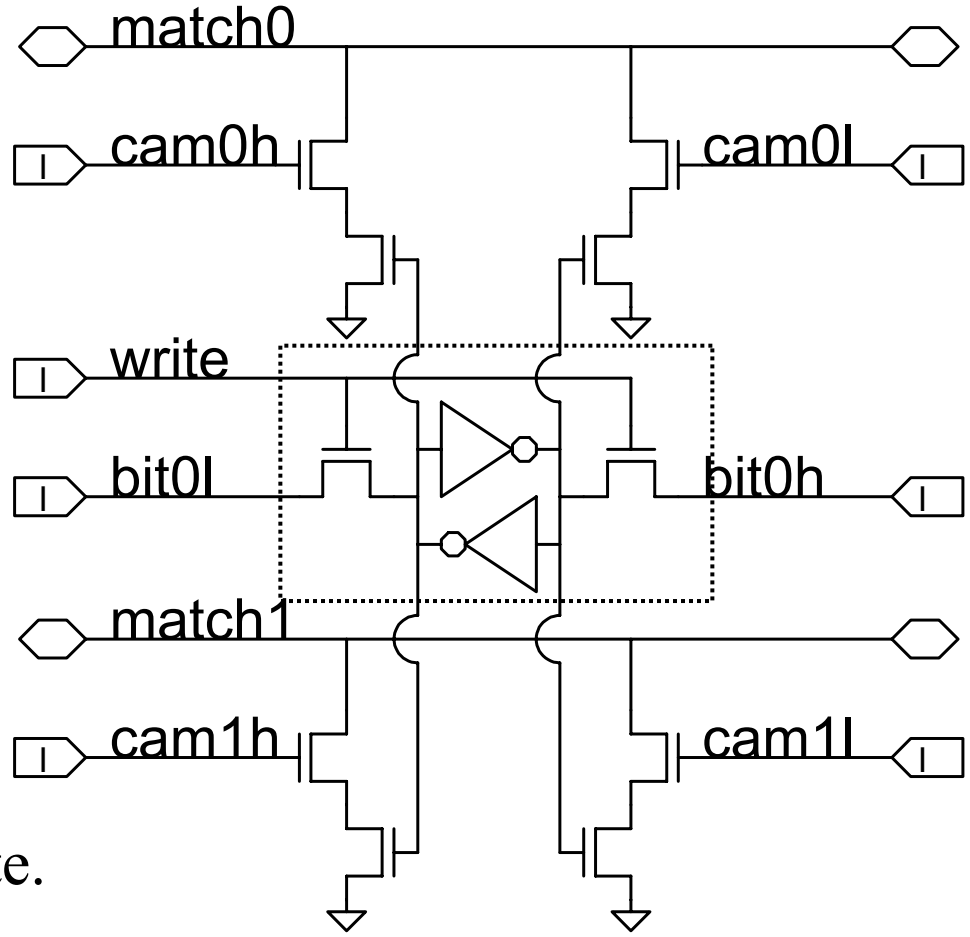
Level 1 TLB Pulsed Bit Lines

- The virtual address is presented to the CAM cell on a pair of dual-rail pulsed bit lines.
- Pulsed bit lines eliminate precharge drive fights and allow for the removal of clocked evaluate FETs on downstream dynamic gates.



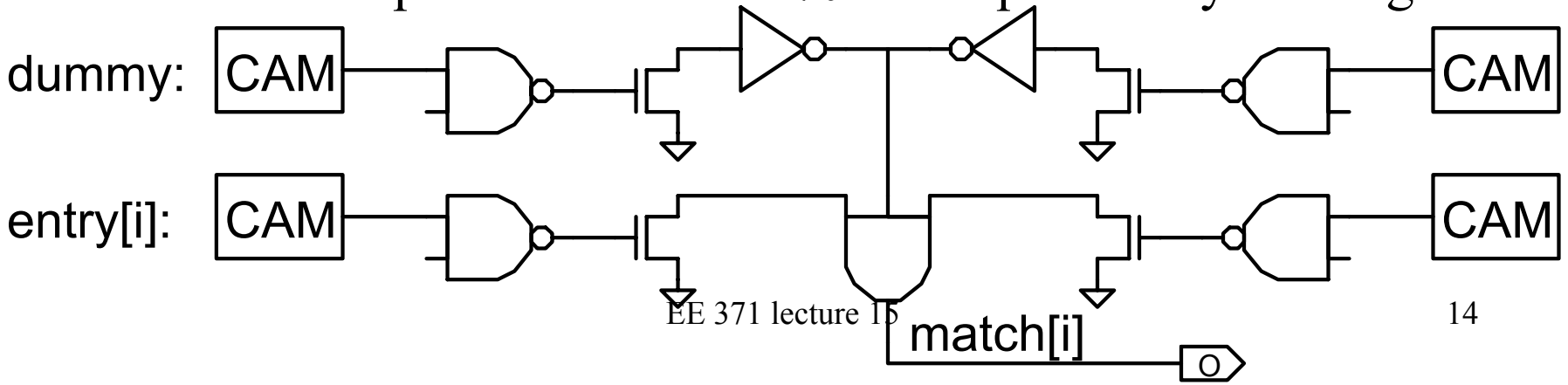
Level 1 TLB CAM Cell

- One write port +2 CAM ports
- CAM bit lines are pulsed
- Cell discharges match[01] when its value does not match the incoming address
- Cell layout is 27.3 μm^2 (4.8 × standard RAM cell)
- Stability is not an issue, since the pass-fets only turn on during a write.
- Charge sharing is not an issue since 3 of every 4 pulldown stacks don't evaluate.

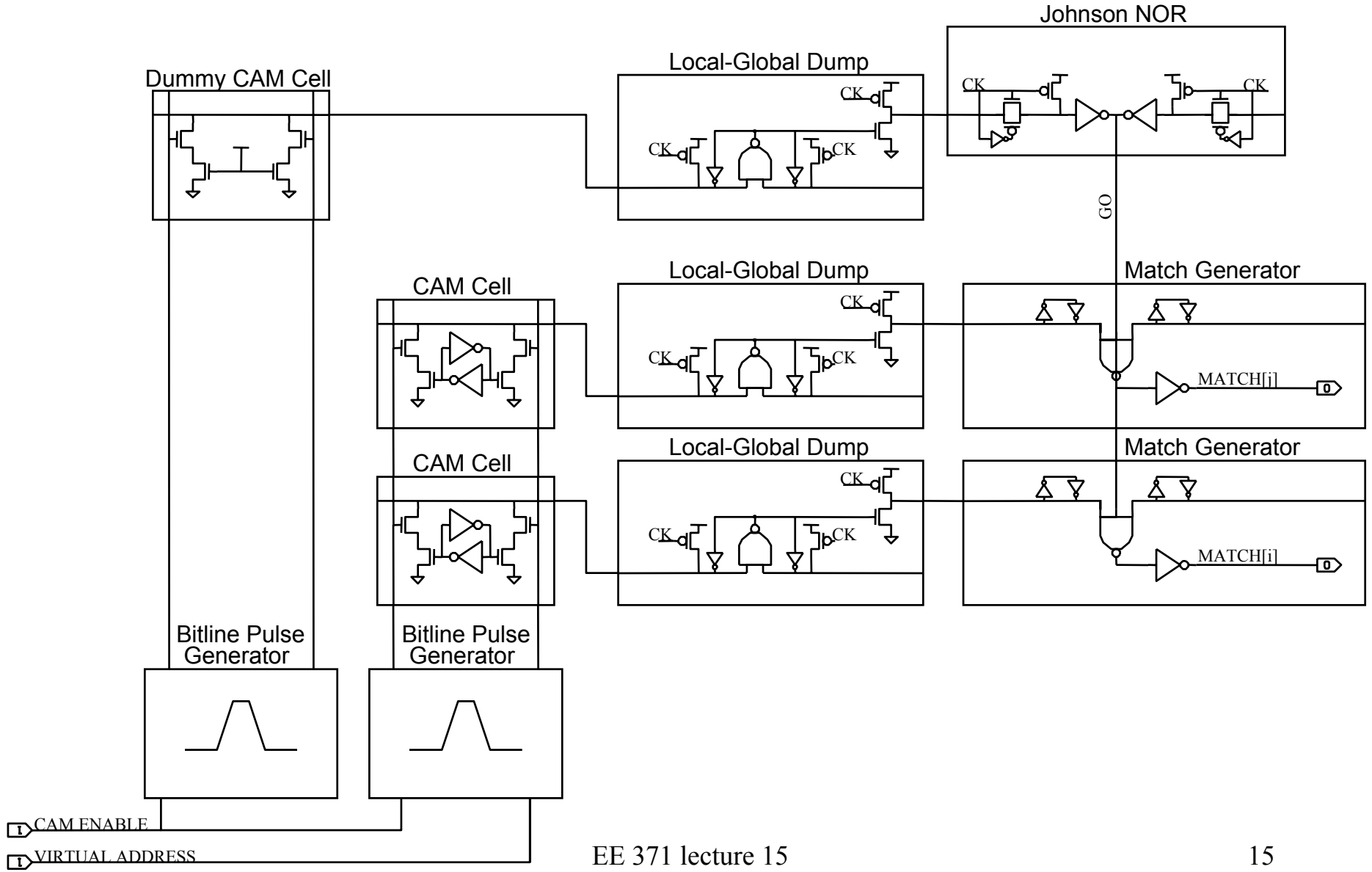


Level 1 TLB Self-Timed Path

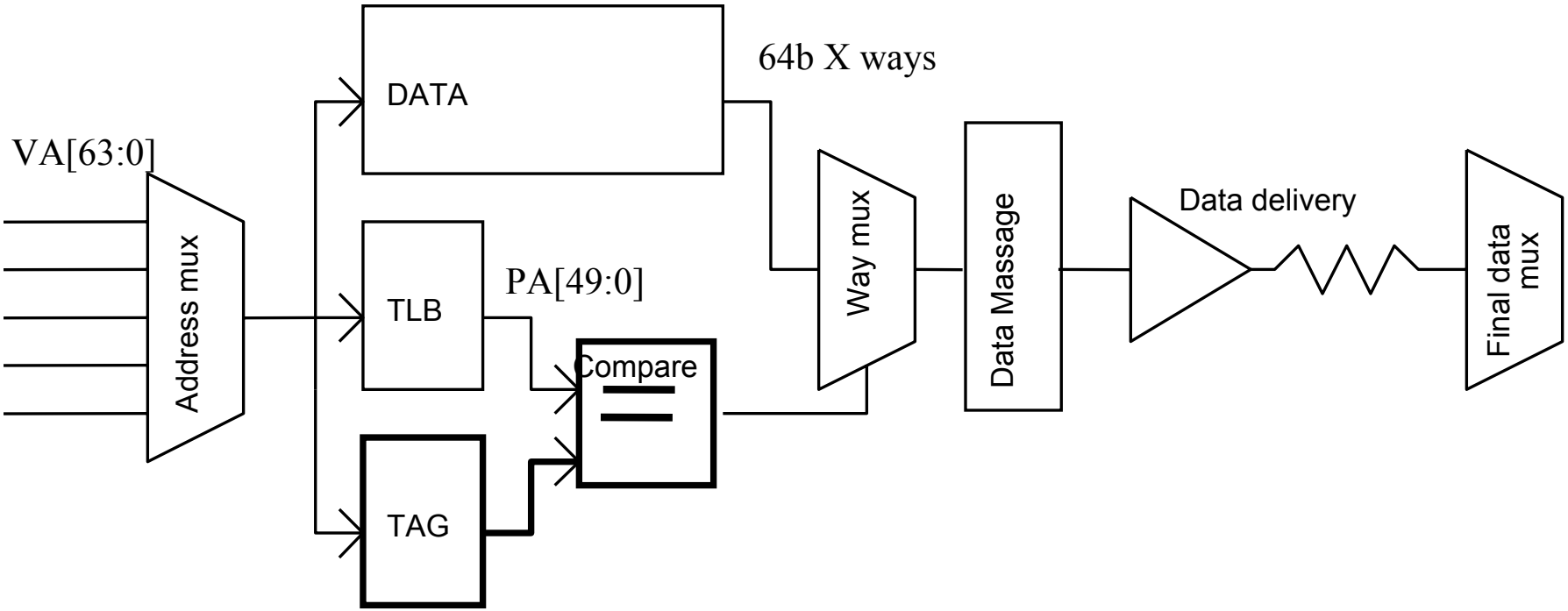
- CAM is wide (90 address bits × 2 pulldown stacks per cell)
- We use a full-rail hierarchical dynamic structure for speed.
  6 CAM cells → 2-input NAND → 4-wide OR → 2-input NAND
- The TLB calculates a monotonic-falling match signal, but the TAG needs a monotonic-rising match signal. Thus, a "dummy" self-timed path is used to create a hard edge.
- The "Johnson NOR" is used to account for PVT and clock route variations between the two halves of the TLB.
- The self-timed path consumes 20% of the path delay in margin.



Level 1 TLB Access



Next: Tag access and Compare



Next: Tag access and compare

- So, I can get the TLB access down to ¼ of a cycle
- For the TAG compare. I'm stuck with either:
  - Doing a true compare: xnor + 38b AND gate, or
  - Doing a precharge pulldown !compare, then either self-time it, or wait for a clock
  - Invent something new -> prevalidated the tags
- The idea is to store a one-hot vector in the tags instead of a physical address
  - This vector points to the TLB entry that contains the translation for the physical address of this tag

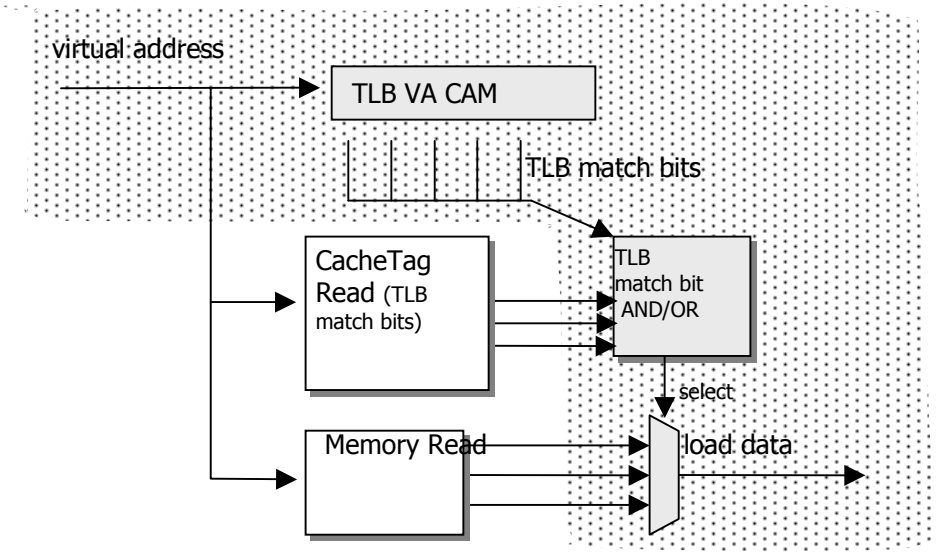


Figure 3: Prevalidated Cache Tag Design Load Critical Path

- The 38 bit compare gets replaced with a 32b (# of TLB entries) domino AND/OR gate.
- *Preserves the monotonicity* for the subsequent way mux and rotate
- Much faster and smaller

Prevalidated Tag Advantages

- Faster TLB outputs
  - No TLB RAM read is required
- Faster TAG compare
  - Dual rail compare + 38-bit dynamic AND

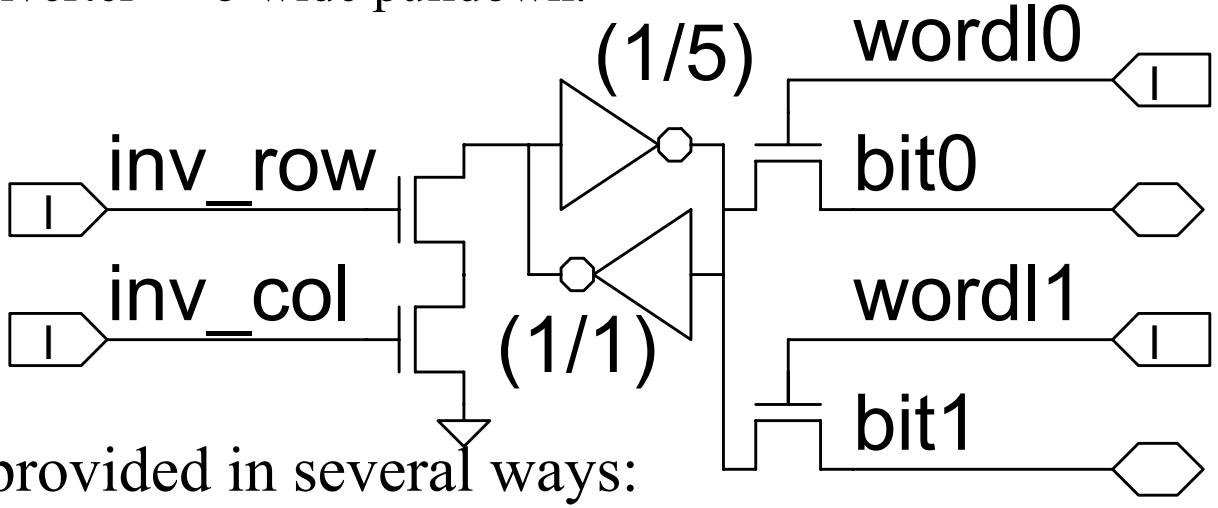
    vs.
  - Dual rail miscompare + 38-bit OR + a hard clock edge

    vs.
  - **Single rail compare + 32-bit dynamic OR**
- Less wiring in both the TLB and TAG arrays
  - 38 address bits × 2 rails per bit

    vs.
  - **32 bits × 1 rail per bit**

Prevalidated Tag Array

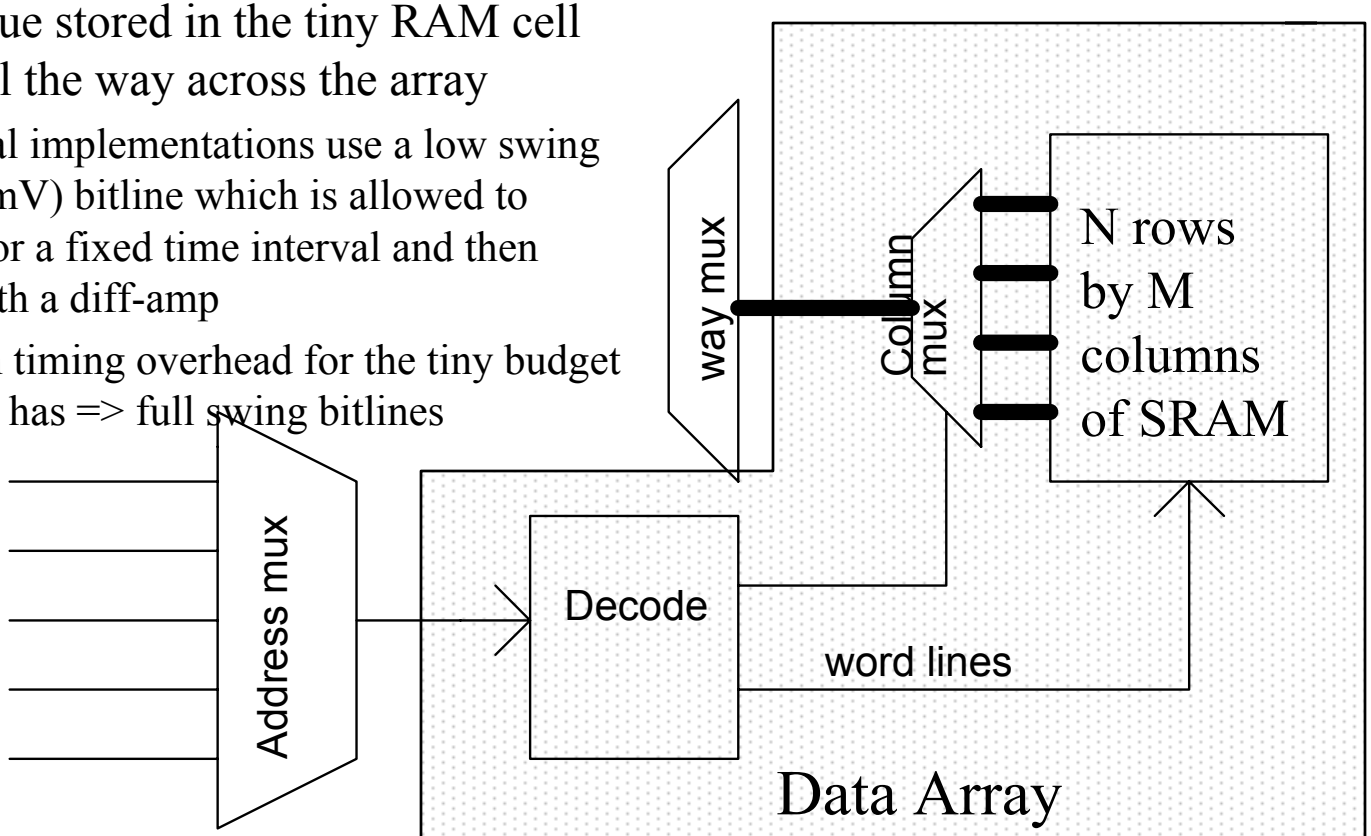
- >1 kB array
  - 64 indices × 4 ways × (32b one-hot vector +1 valid bits)
- 2 independent read ports
- Two-level dynamic full-rail RAM read for speed
  - 8 RAM cells -> inverter -> 8-wide pulldown.
  - No sense-amp.



- SER detection is provided in several ways:
  - Cache is write-through, so spurious invalidations are OK
  - The ways are interleaved so that double-bit errors either affect two different ways or two different indices.
  - Multi-hot detector for detecting errors.

Next: Data Array Access

- For a data array access, we must take the same address bits that go to the TAG, decode them into individual word lines and column selects
  - Pull one 64b word out of the 512 in each way
- Sense the value stored in the tiny RAM cell and send it all the way across the array
  - Traditional implementations use a low swing (100-200mV) bitline which is allowed to develop for a fixed time interval and then sensed with a diff-amp
  - Too much timing overhead for the tiny budget this cache has => full swing bitlines

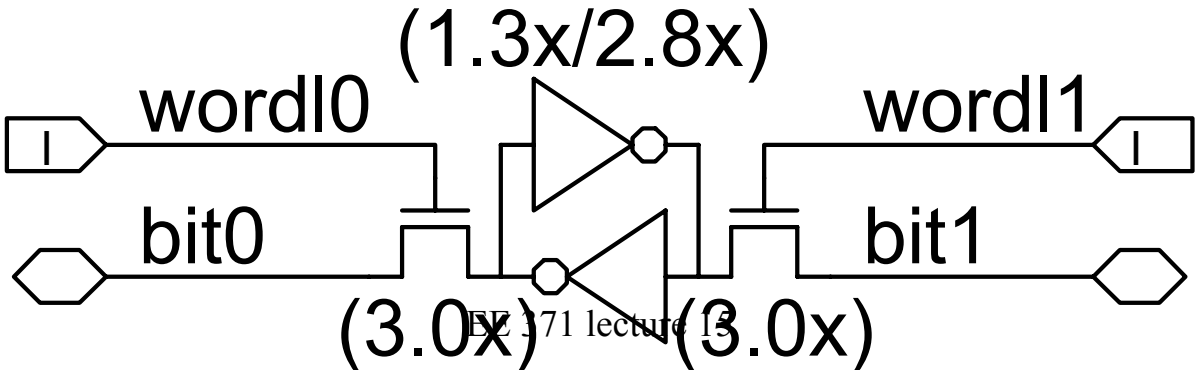


Level 1 Data Array Overview

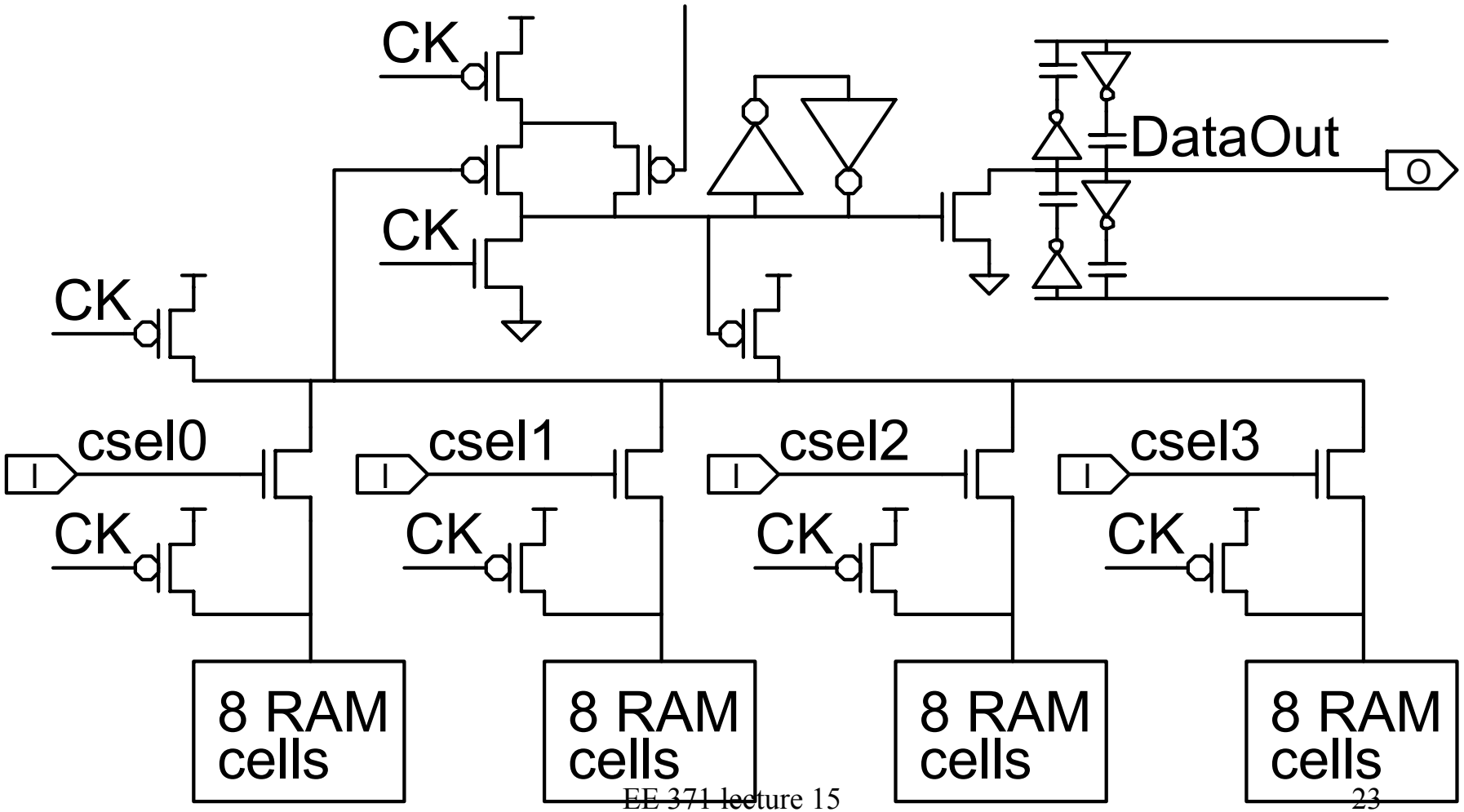
- 16 kB of data + 2kB of parity
  - 4 ways of 4kB each
- Parity protected at the byte level to allow sub-word stores
- 2 independent load ports and 2 store ports.
- Supports 3 operations
  - Loads (8B)
  - Stores (1B - 8B)
  - Fills (64B)
- Array is approximately 50% RAM-area efficient
  - Tradeoff area for speed.
- 1280µm × 1770µm in a 0.18µm process.

Level 1 Data Array RAM Cell

- 6T-RAM cell is made dual-ported by independently controlling the word lines for the pass fets.
- Single-ended full-rail dynamic reads.
- Cell uses large NFETs for speed.
- Stability is afforded by a quick discharge of a lightly loaded bit line during a single ended read.
- Cell layout is 7.01 μm^2 (1.25 × standard RAM cell)

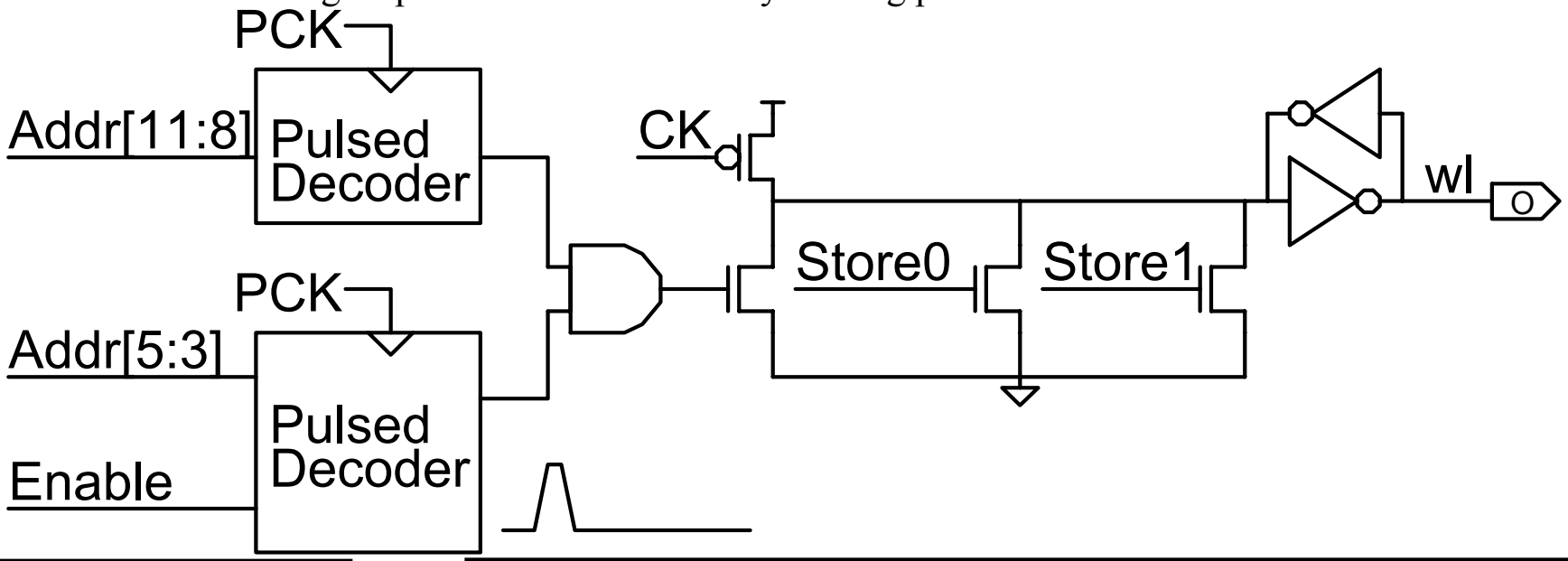


L1D Data Array Local-Global Dump Circuit



Level 1 Data Array Decoder

- Decode is performed hierarchically.
- Pulsed decoders allow for the removal of clocked-evaluate FETs in downstream dynamic logic.
- Static AND gate provides noise immunity on long pulse routes.



A domino decoder was too slow → use a self-timed "annihilation gate" for speed

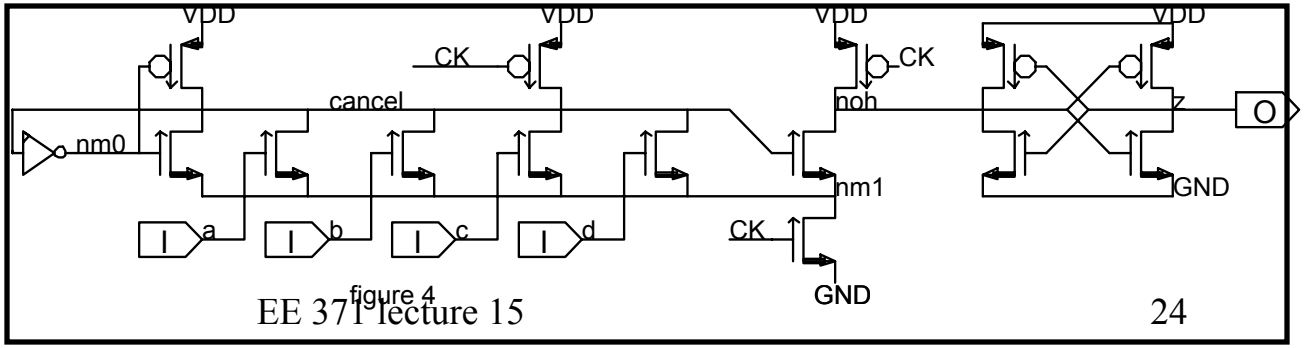


figure 4

Level 1 Data Array Decoder

- An “annihilation gate” is:
  - Fast: 1 domino gate delay for a large fan-in AND
  - Small: A single FET per input
  - Low input load
  - Scary!

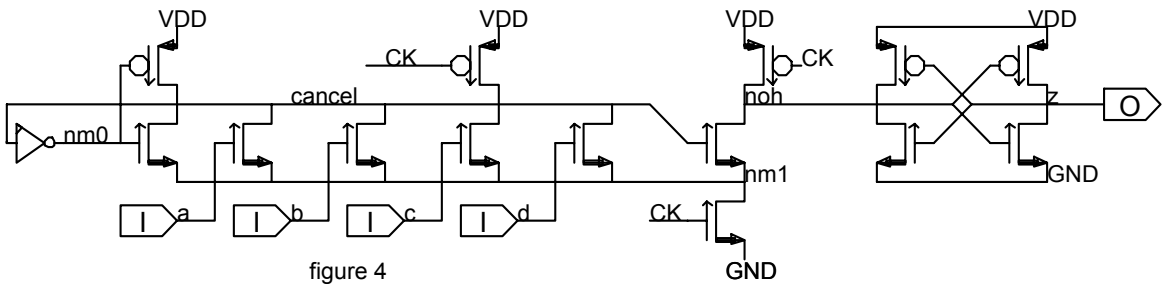
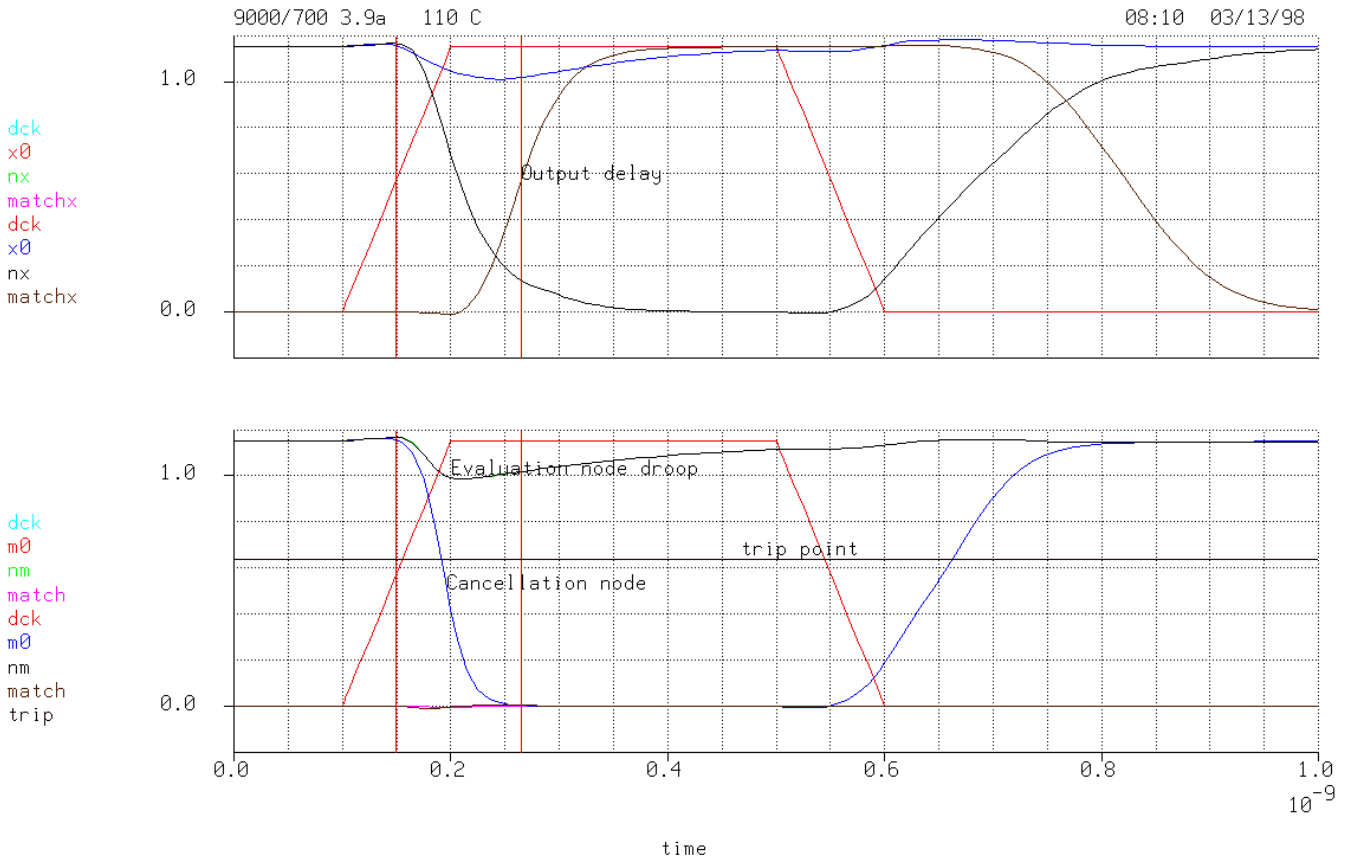
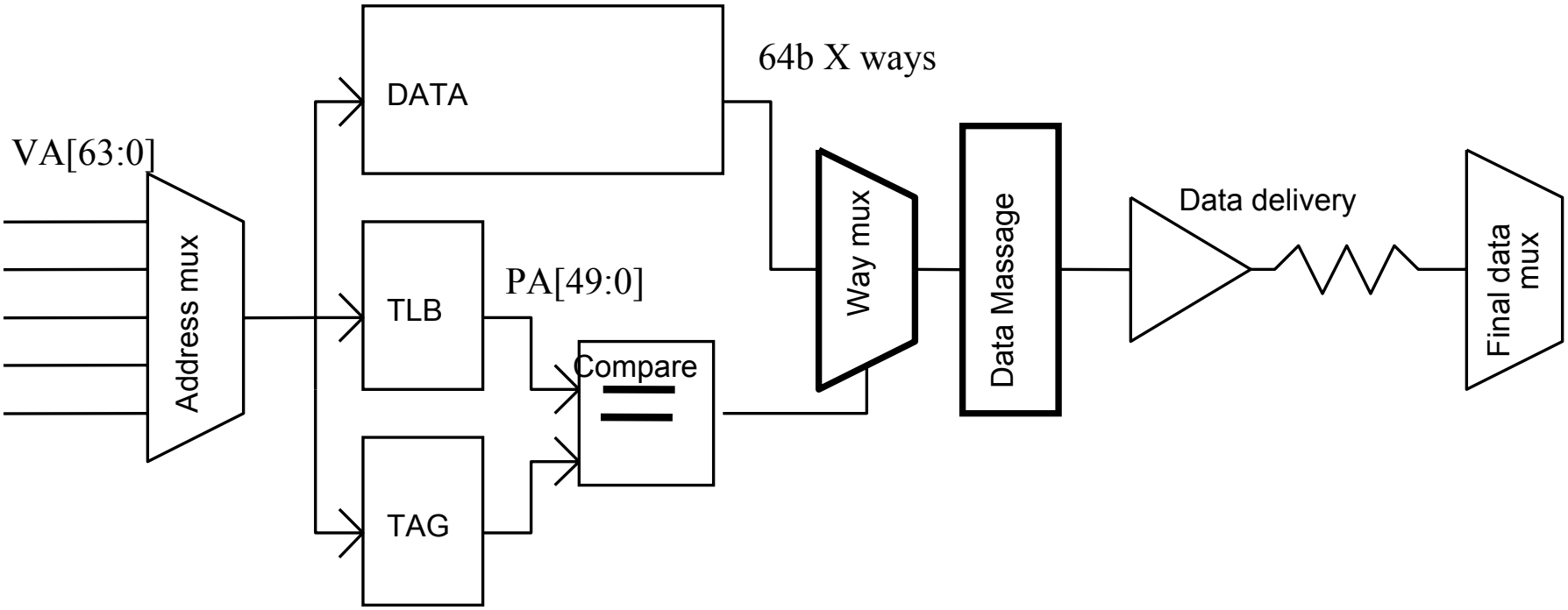


figure 4
AGATE_SIM1 SL <858BSIM3 SCIPINFO>

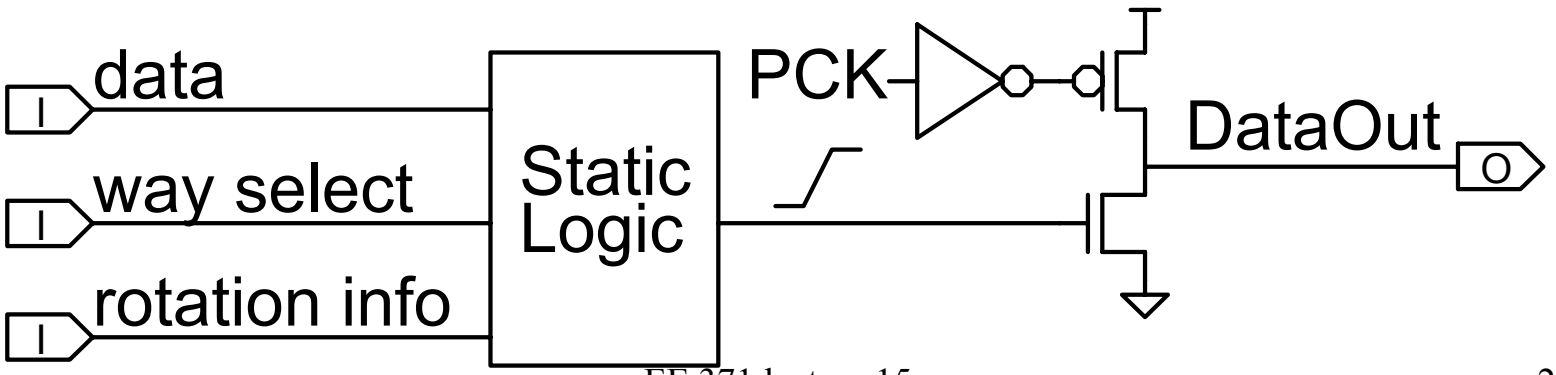


Way Selection and Data Massage

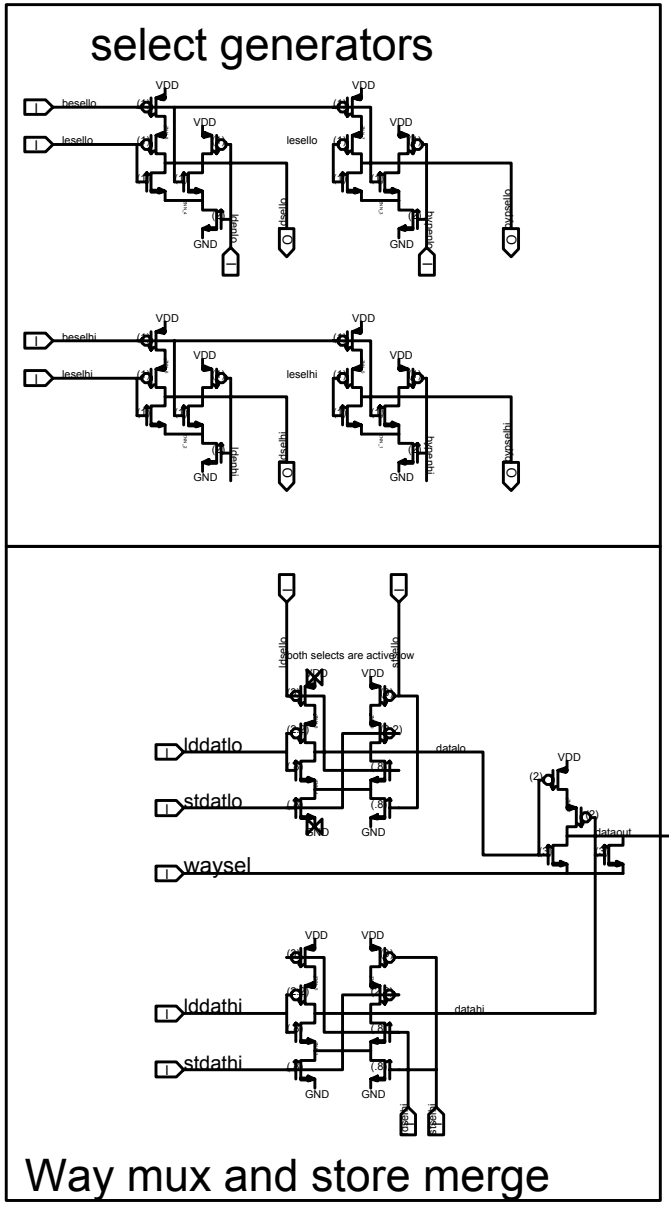


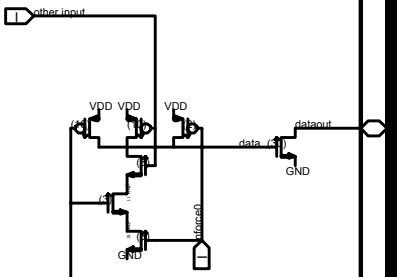
Rotating Way Mux

- This block accomplishes 3 main tasks:
  - Way Selection
  - Byte Rotation (for sub 8B loads)
  - Litte Endian / Big Endian Transformations
- A series of monotonicity-preserving static gates are used to perform the first stages of muxing.
  - Provides noise immunity
  - Allows phase stealing
- A pulse-precharged distributed pulldown network is used for the final stage.

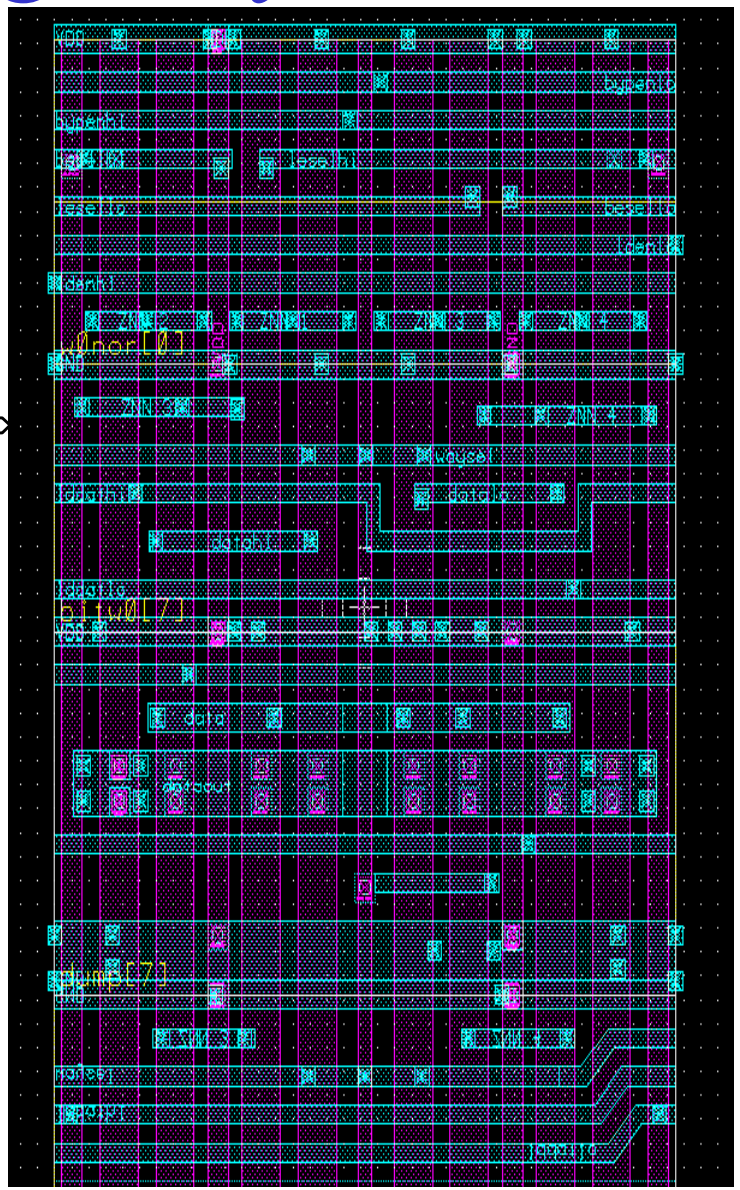


Rotating Way Mux



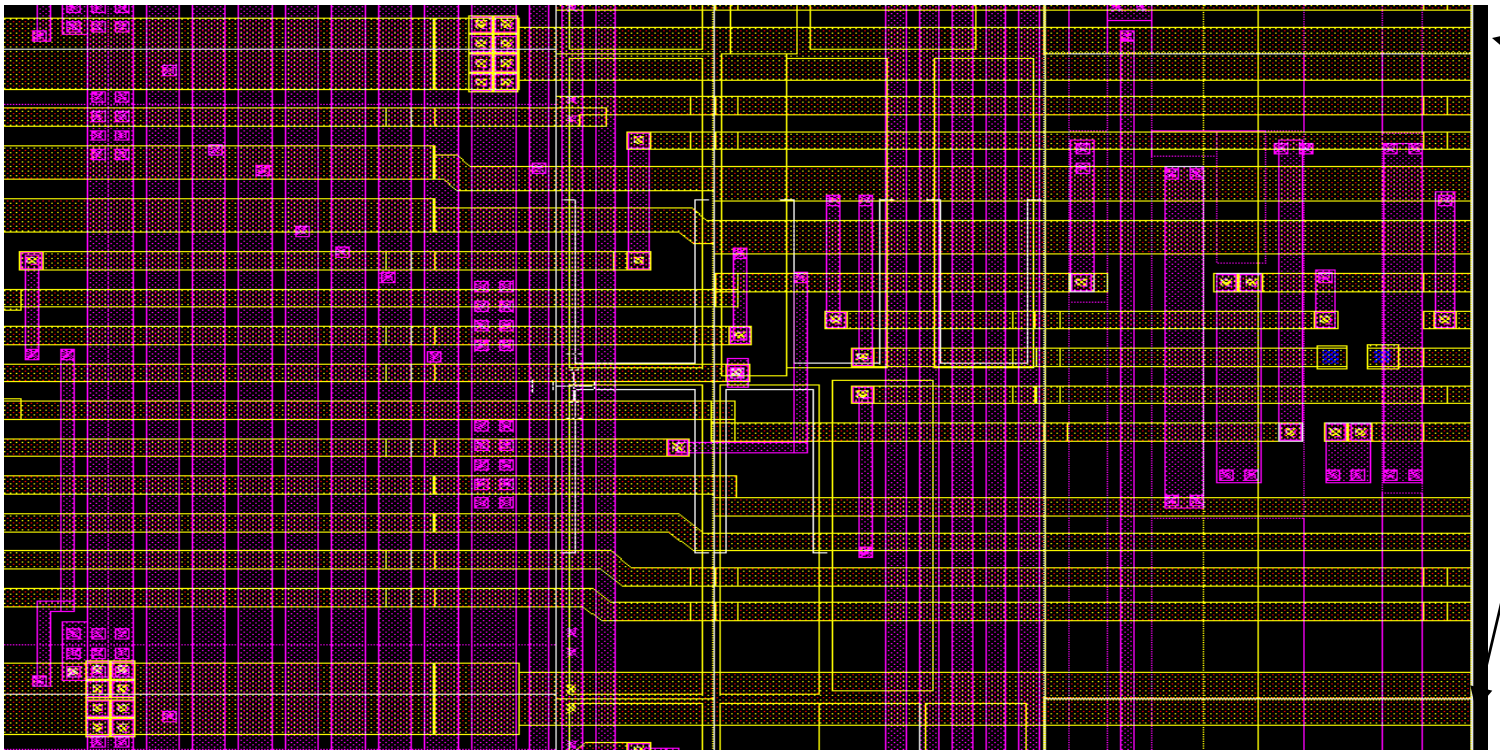


- Painstaking Layout is critical
- RC delay
- Coupling
- Size and features

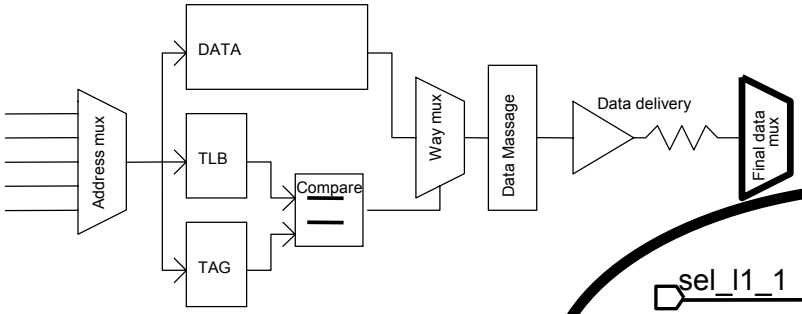


Finally: Delivering the Data

- This requires careful wire engineering
  - With all the parallelism in the execution unit, we had to use every last metal 4 track
  - Since the cache data delivery is the latest arrival, we could demote a couple other tracks down to metal 2 to widen up the cache busses
    - Almost a 50ps (5% of cycle) gainer



Cache->Integer Bypass

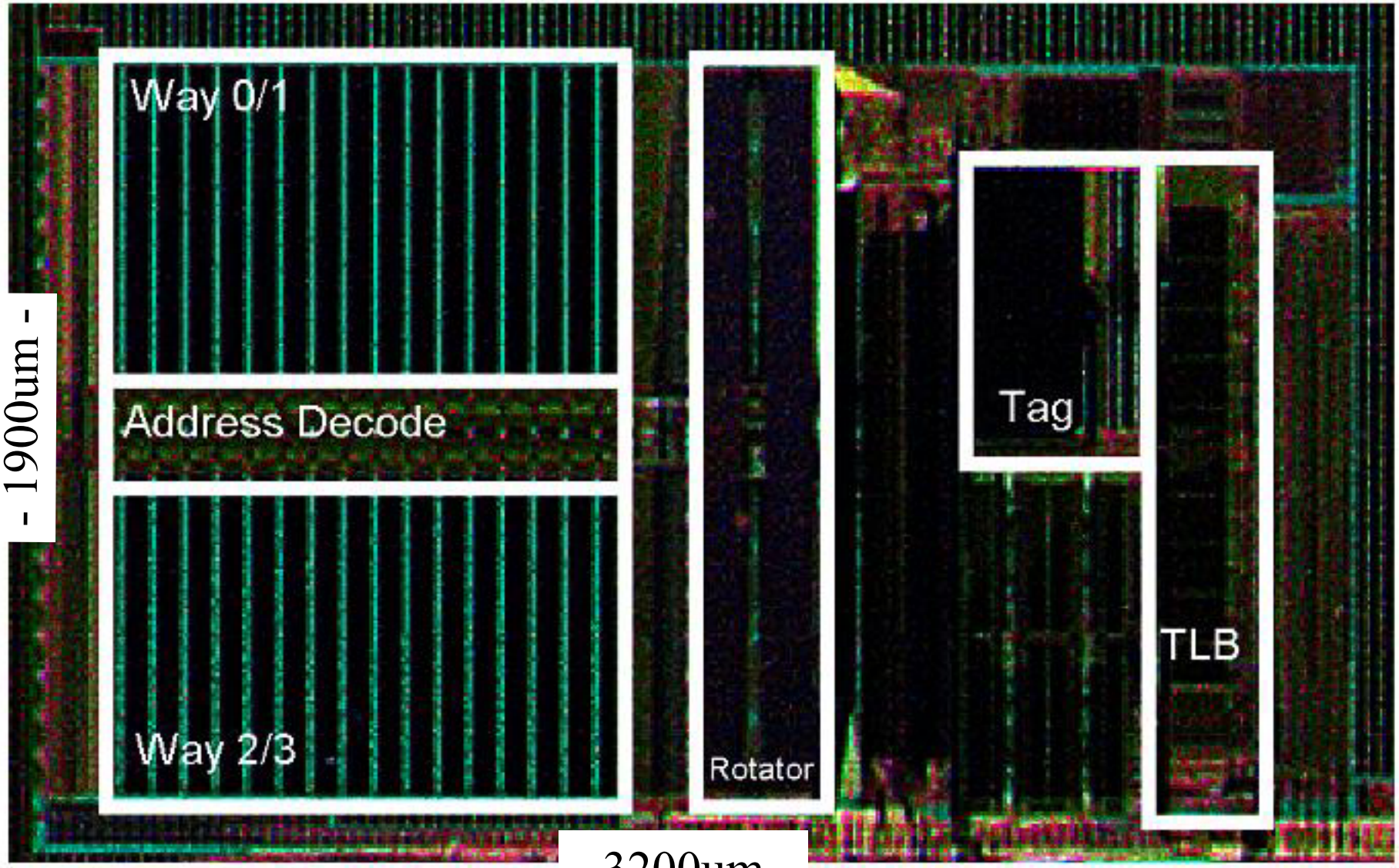


Mux selects enable pseudo-NMOS pullup and latch reducing power and enabling stall recovery

High Strength Feedback for fast rising edge

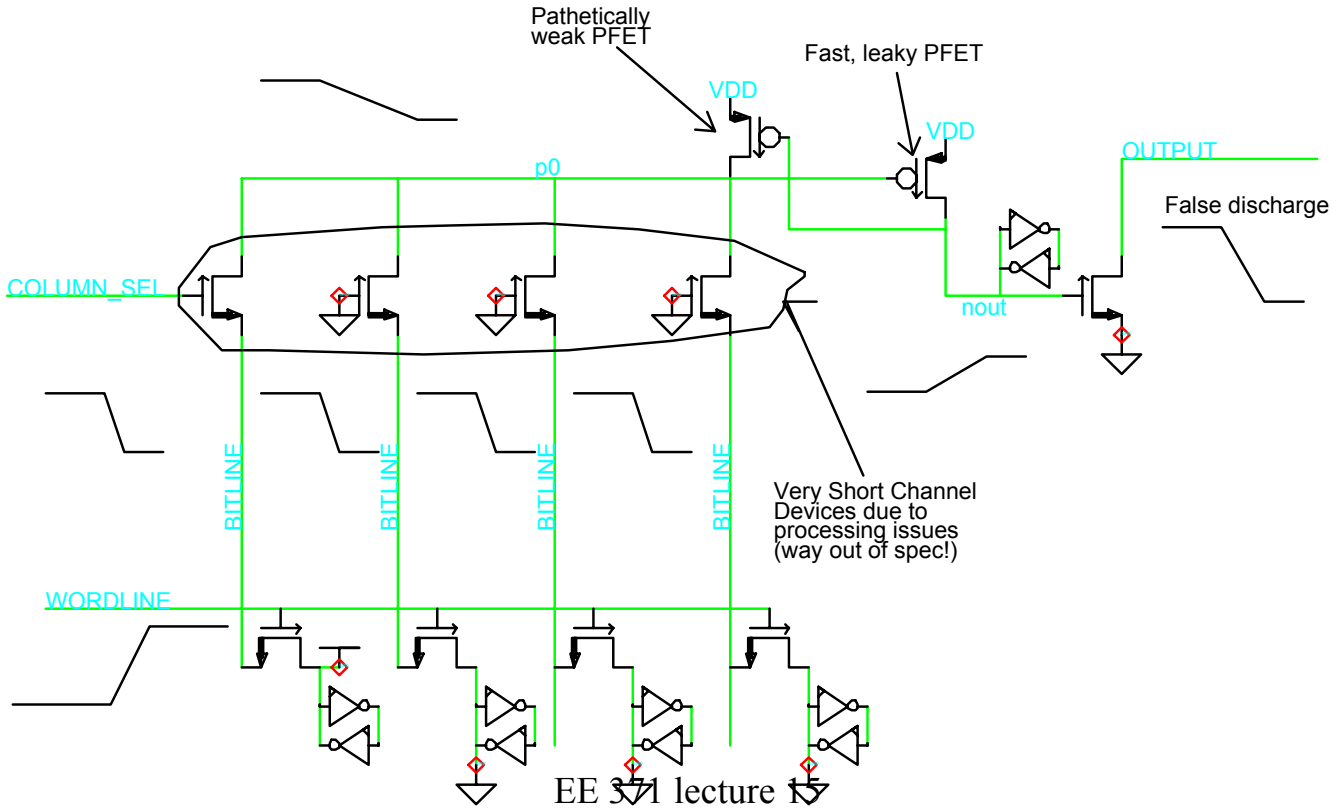
Noise problem?

Level 1 Cache Photo



Silicon Results

- Most everything worked as expected – careful engineering pays off
  - No coupling or supply noise problems
  - No pulse width or self-timing problems
- Only significant issues had to do with unexpected processing conditions for the FETs combined with sensitive, high speed circuits

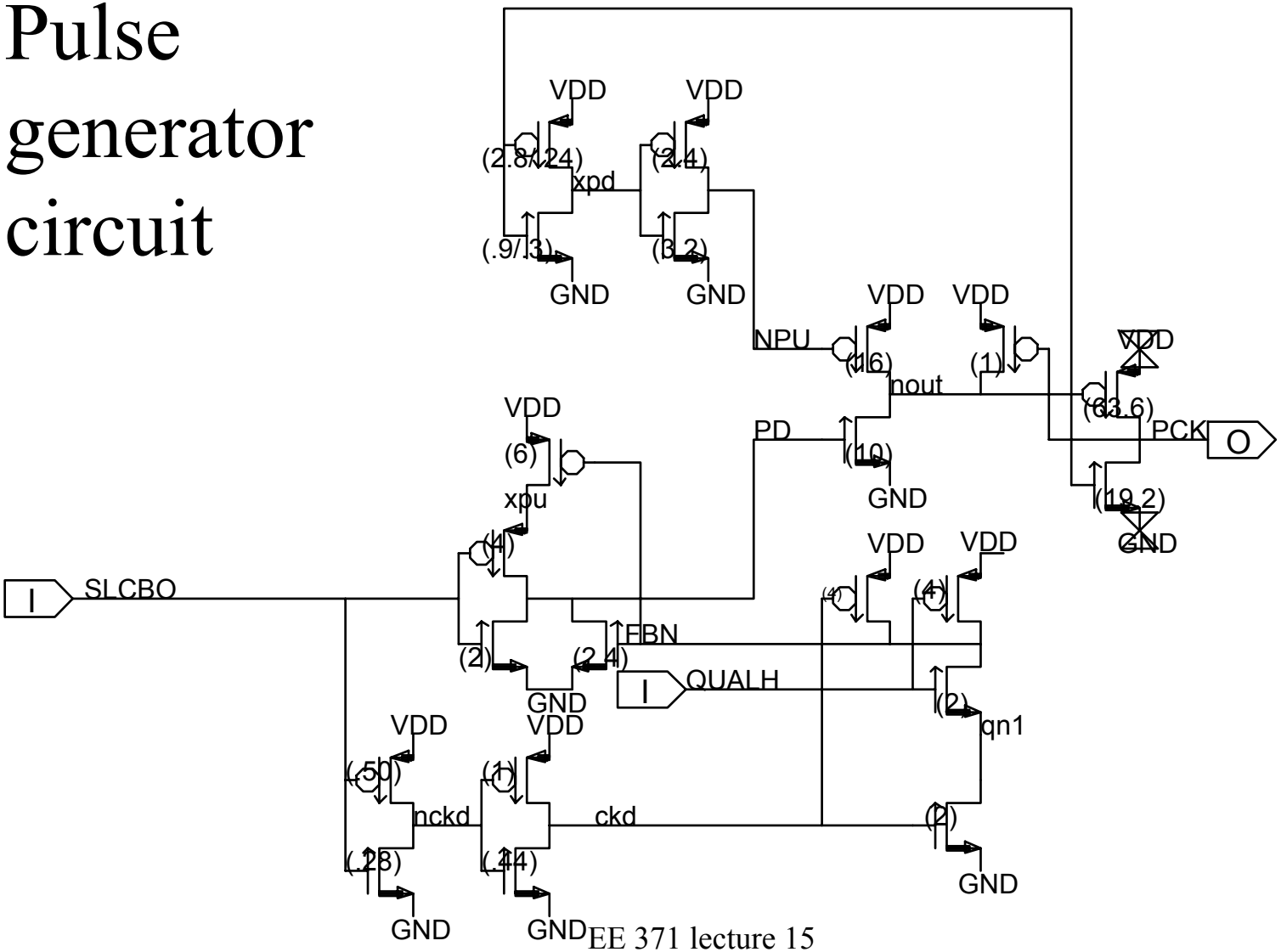


Conclusions

- There are *plenty* of opportunities to innovate in VLSI design (and get paid for doing it!)
  - As long as silicon technology keeps advancing and changing, there are new and interesting ways to exploit it
  - As long as information technology keeps driving our economy, people will pay for performance and features
- A few keys to finding the best way to contribute:
  - Study the landscape; what's been done before, the capabilities of the technology, what customers want
    - Understand the architecture and performance issues, not just circuits
  - Pick one or two strategic areas to do something truly new (it will always be more challenging than you thought)
    - For this example, we had the pre-validated tag, single-ended RAM and annihilation gates, the rest was just tenacious engineering
  - Engineer the heck out of it; leave no stone unturned searching for problems and applying airtight solutions

Backup

- Pulse generator circuit



Backup: Coupling Noise Solutions

- Top level cache read bitlines: 2mm, min space, precharged-pulldown, metal 4
- Anti-miller devices reduce crosstalk induced droops
- Noise margins improved significantly with a small performance penalty