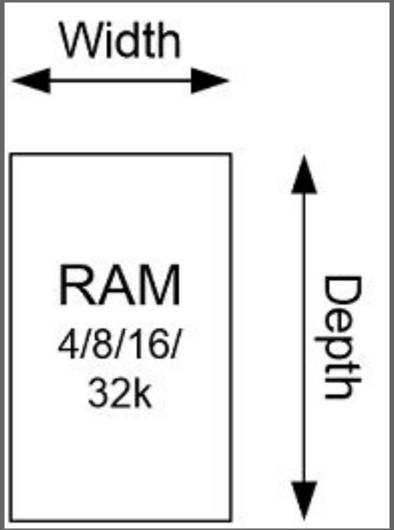
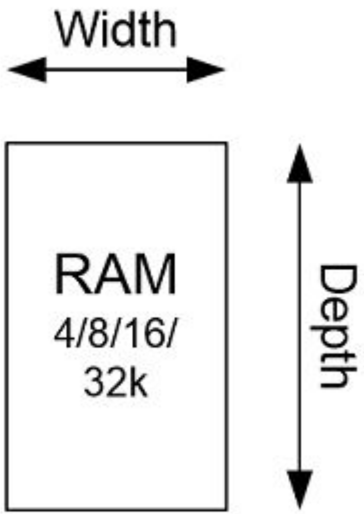
CPE 470 - RAM Configurations



RAM Metrics

- Width: How many bits are read at a time?
- Depth: How many rows are there to read from?
  - Nomenclature: depth x width
  - 128x32 means 128 rows of 32 bits

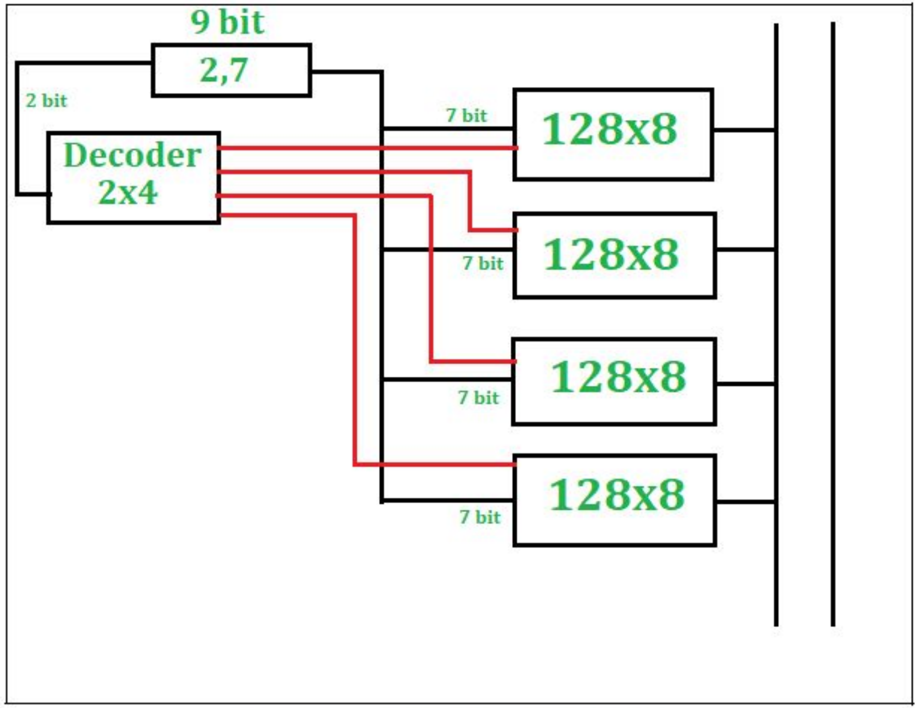


- Ram Size: Width * Depth = Ram Bits
  - Divide by 8 for Bytes

- Addressability:
  - Byte Addressable ⟶ Addresses represent byte locations
  - Word Addressable ⟶ Addresses will be multiple of 4

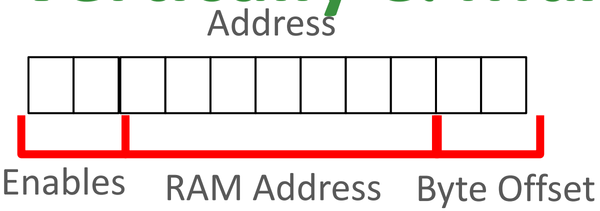
Combining RAMs

- How many bits are needed to address our RAMS?
  - Log Base 2 of total size in bytes gives address size
  - `$clog2` in system verilog can compute this
- This address can be split up into 3 parts:
  - Bottom 0 to 2 bits: byte offset in word
    - Log2 of width in bytes
    - 32 Bit Width → 2 bit byte offset
    - 8 Bit Width → No bye Offset
  - Middle N bits - fed to internal RAMs
    - Log2 of depth
  - Remaining Top Bits: used to select between RAMs
    - Log2 of # of ram modules

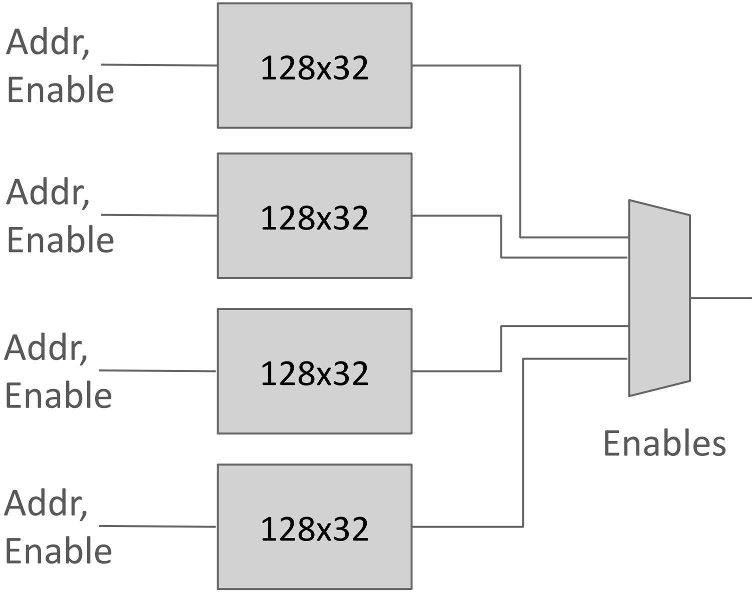


Design of 512 x 8 RAM

Combining Rams: Vertically & Muxed

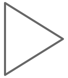


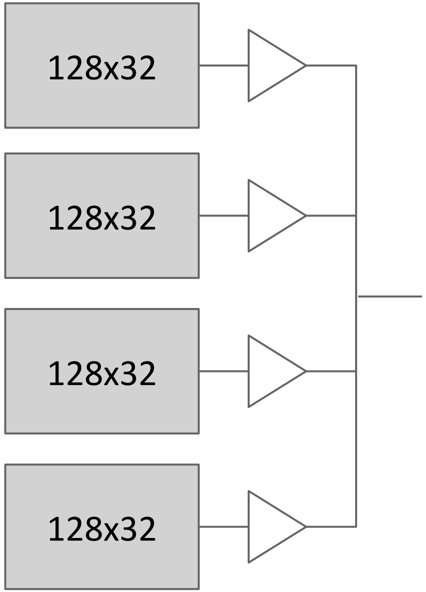
- Combine RAMs vertically to increase their depth
  - Same read size, more locations
- All RAMs get address
  - Each RAM gets enabled only when address is within its range
- RAM output is fed through a mux
  - Mux also driven by enable signals from address
  - Mux enables are one cycle delayed from inputs



Combining Rams: Vertically & High Z

- Similar to muxed technique
- Saves area/speed using tri-state buffers to drive a common output bus
  - Scales better with higher number of internal rams
  - Muxes get deeper/slower as they scale, tri state buffers less so
- Higher risk of errors
  - If any two rams output at the same time, short circuit

tri-state buffer: 



Non-Aligned Reads

- So far, we have only supported word-aligned reads
  - Each word comes from the output of a single aligned ram
  - 32 bit system: can only read at address 0, 4, 8, etc.
- What if we want to read full word, but at non-aligned addresses?
  - Notice how each column is a single byte
  - Each byte is either in the same row as the first byte or the row after

Bytes 0, 1, 2, 3

Reading Address 0

	+0	+1	+2	+3
Addr: 0	0x01	0x23	0x45	0x67
Addr: 4	0x89	0xAB	0xCD	0xEF

Reading Address 1

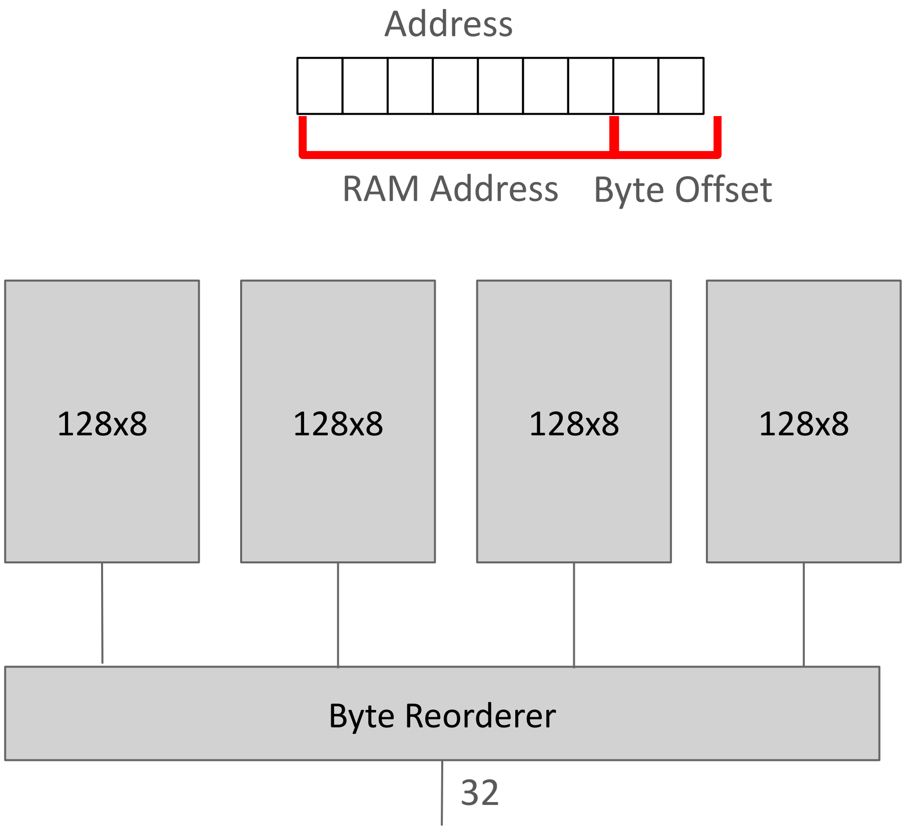
	+0	+1	+2	+3
Addr: 0	0x01	0x23	0x45	0x67
Addr: 4	0x89	0xAB	0xCD	0xEF

Reading Address 3

	+0	+1	+2	+3
Addr: 0	0x01	0x23	0x45	0x67
Addr: 4	0x89	0xAB	0xCD	0xEF

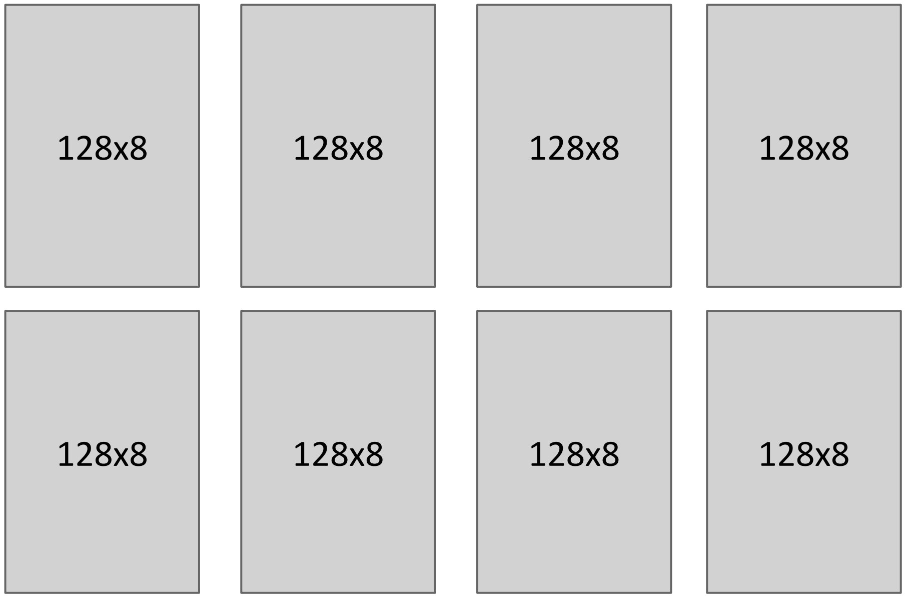
Combining Rams: Horizontally

- Each column corresponds to a specific byte
  - Based on the byte offset, columns are reordered to match previous slide pattern
  - Address might get incremented depending on offset
- No vertical rows → Top part of address fed directly into each module
  - No enable signals, all RAMs are always enabled



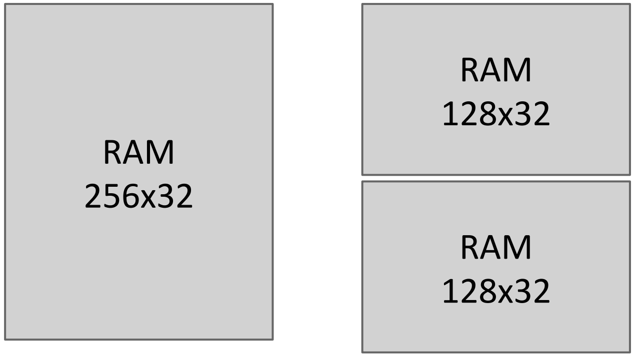
Combining Rams: Multidimensional

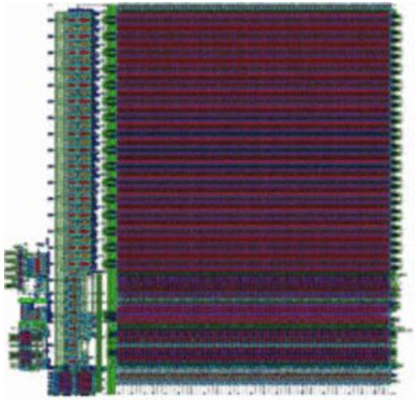
- Techniques of vertical and horizontal combination can be combined
  - Can get byte addressability and bigger RAMs
- Requires a lot of control logic to do both muxing of rows and reordering of columns

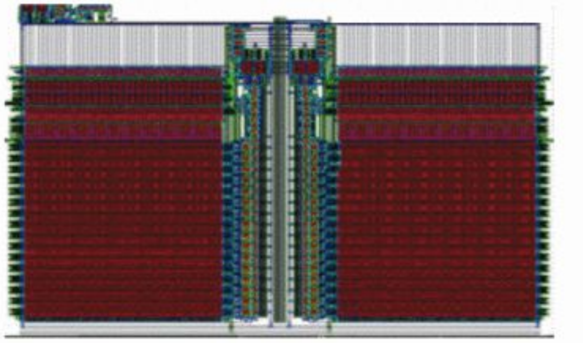


Scaling Tradeoffs

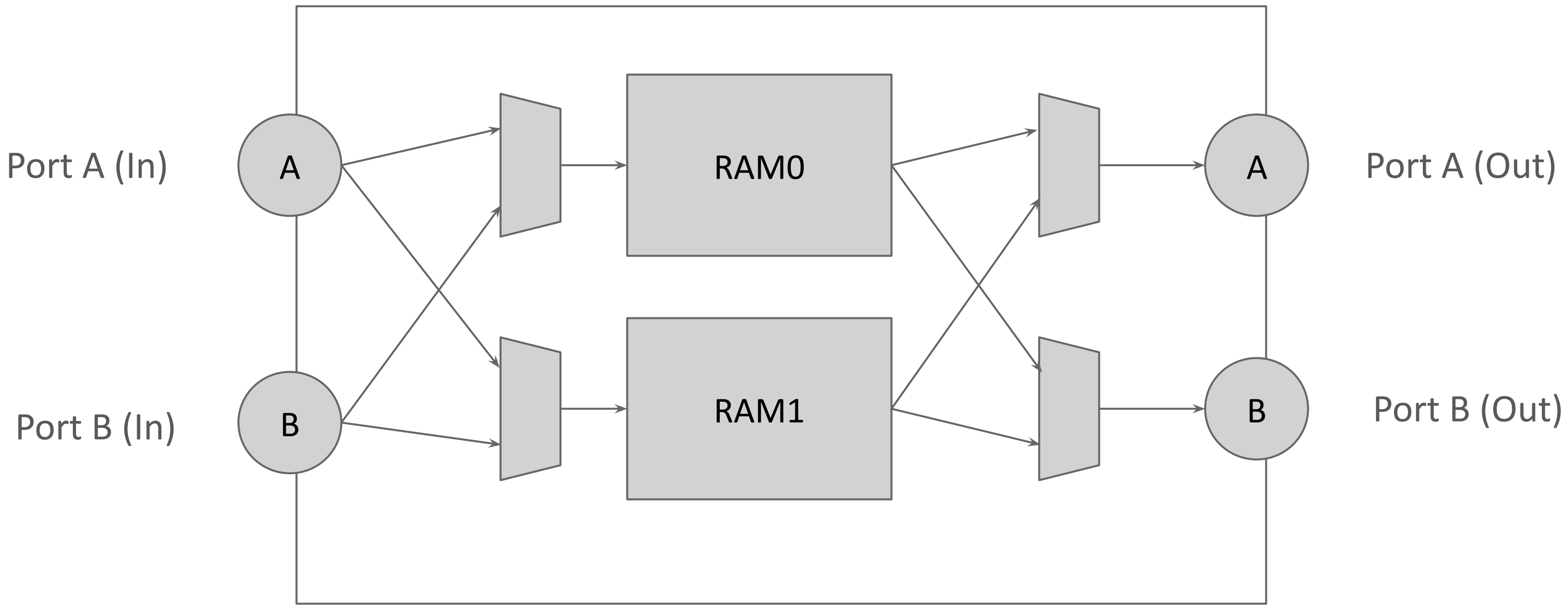
- What is the difference between a ram composed of one module versus multiple modules?
  - Example: 256x32 vs dual 128x32
- Answer: Decode Logic
  - Each unit has its own access control logic for reading/writing
- Single RAM is higher density ⟶ same size but only one set of control logic
  - How can we use repeated control logic to our advantage?







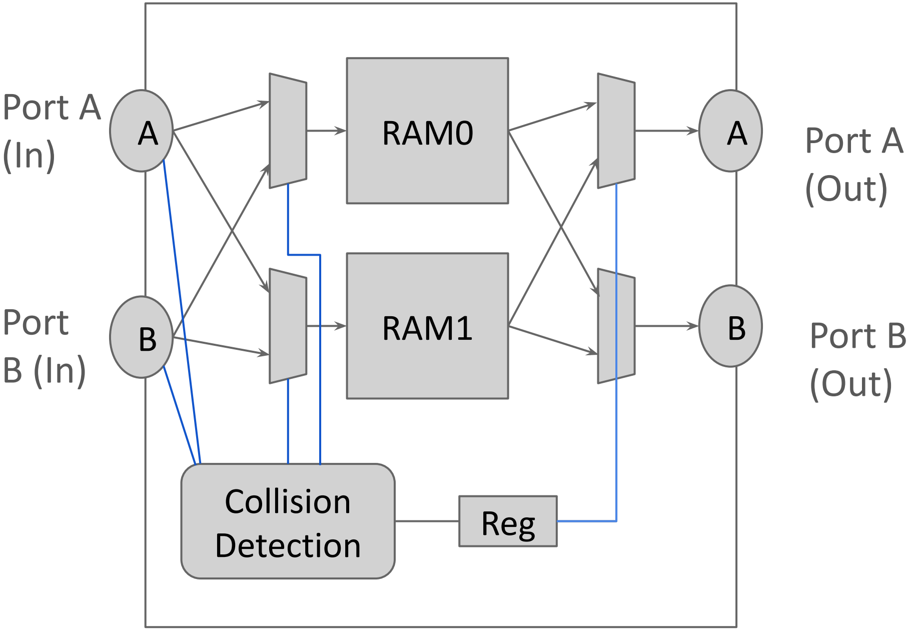
Adding “Ports”



Since each inner RAM has its own access channel, can use this to make a 2 port RAM!
Has the same size as the total of RAM0 and RAM1

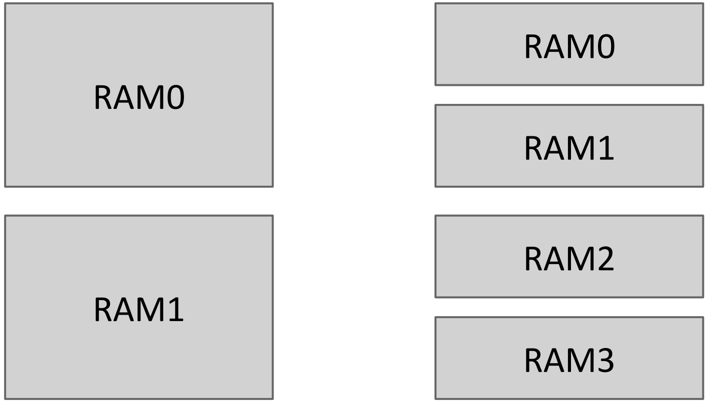
Adding “Ports”

- Relies on each port accessing a different RAM
  - Either A or B can access RAM0 or RAM1, as long as they are not accessing the same one
- If ports access a piece of data on the same RAM, one port has to stall
  - Collision detection and delegation required to manage
- Which port is using which RAM → has to be remembered for one cycle
  - Use register to track which output port owns which output
  - Mux output based on that



Collision Cost

- Assuming Random Access using 2 ports:
  - 2 Inner RAMs → 50% collision chance
  - 4 Inner RAMs → 25% collision chance
- Best in use cases where each port is accessing different parts of memory
- Using more RAM modules lowers chance of collision
  - Tradeoff: hurts density, more logic/wiring



References

- https://www.geeksforgeeks.org/design-of-512x8-ram-using-128x8-ram/