Digital Integrated Circuits (83-313)

Lecture 9: Memory Peripherals

Emerging Nanoscaled Integrated Circuits and Systems Labs Prof. Adam Teman 25 May 2021

The Alexander Kofkin Faculty of Engineering



Bar-Ilan University

Disclaimer: This course was prepared, in its entirety, by Adam Teman. Many materials were copied from sources freely available on the internet. When possible, these sources have been cited; however, some references may have been cited incorrectly or overlooked. If you feel that a picture, graph, or code example has been copied from you and either needs to be cited or removed, please feel free to email <u>adam.teman@biu.ac.il</u> and I will address this as soon as possible.

Lecture Content



© Adam Teman, 2021

Memory Peripherals Overview





Memory Architecture



<u>Memory Size</u>: W Words of C bits = $W \ge C$ bits <u>Address bus</u>: A bits $\rightarrow W=2^A$

Number of Words in a Row: 2^M Multiplexing Factor: M

Number of Rows: 2^{A-M} Number of Columns: $C \ge 2^{M}$

<u>Row Decoder</u>: $A \rightarrow 2^{A-M}$ <u>Column Decoder</u>: $M \rightarrow 2^{M}$

© Adam Teman, 2021

Synchronous SRAM Interface

- A typical on-chip synchronous SRAM features:
 - Single-cycle write/read latency
 - Byte write mask

5

- Active low Write Enable (i.e., WEN=1 → Read Enable)
- The timing diagram can be viewed, as follows:



(2 ^m xn SRAM		
	A[m-1:0]		
	D[n-1:0]	Q[n-1:0]	
	WEN[p-1	:0]	
	CEN		
	CLK		

(1) Rising edge of the clock results in WRITE, when WE is low.

(2) Rising edge of the clock results in READ, when WE is high. Valid data appears on the output after a delay.





Row Decoder Design





Row Decoders

- A Decoder reduces the number of select signals by log₂.
 - Number of Rows: W
 - Number of Row Address Bits: $A = \log_2 W$





Row Decoders

• Standard Decoder Design:

- Each output row is driven by an AND gate with $k = \log_2 N$ inputs.
- Each gate has a unique combination of address inputs (or their inverted values).
- For example, an 8-bit row address has 256 8-input AND gates, such as:

$$WL_0 = \overline{A}_7 \overline{A}_6 \overline{A}_5 \overline{A}_4 \overline{A}_3 \overline{A}_2 \overline{A}_1 \overline{A}_0$$

$$WL_{255} = A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$$

NOR Decoder:

• DeMorgan will provide us with a NOR Decoder.

Precharge

• In the previous example, we'll get 256 8-input NOR gates:

$$WL_{0} = A_{7} + A_{6} + A_{5} + A_{4} + A_{3} + A_{2} + A_{1} + A_{0}$$
$$WL_{255} = \overline{A}_{7} + \overline{A}_{6} + \overline{A}_{5} + \overline{A}_{4} + \overline{A}_{3} + \overline{A}_{2} + \overline{A}_{1} + \overline{A}_{0}$$

Sense Amp

How should we build it?

- Let's build a row decoder for a 256x256 SRAM Array.
 - We need 256 8-input AND Gates.
 - Each gate drives 256 bitcells
- We have various options:



• Which one is best?

Reminder: Logical Effort

$$t_{pd,i} = t_{pINV} \left(p_i \gamma + EF_i \right)$$

$$EF_i \triangleq LE_i \cdot f_i = LE_i \cdot \frac{b_i \cdot C_{in,i+1}}{C_{in,i}} \qquad PE = F \cdot \prod LE \cdot B = \frac{C_L}{C_{in,1}} \cdot \prod LE_i \prod b_i$$

$$EF_{opt} = \sqrt[N]{PE} = \sqrt[N]{F} \cdot \prod LE_i \prod b_i$$

$$N_{opt} = \log_{EF_{opt}} PE = \log_{EF_{opt}} F \cdot LE \cdot B$$

$$t_{pd} = t_{pINV} \sum \left(p_i \gamma + EF_i \right) = t_{pINV} \left(\gamma \sum p_i + N \cdot \sqrt[N]{PE} \right)$$

Problem Setup

• For LE calculation we need to start with:

- Output Load (C_L)
- Input Capacitance (C_{in})
- Branching (B)

Row Decoder

- What is the Load Capacitance?
 - 256 bitcells on each Word Line

$$C_{WL} = 256 \cdot C_{Cell} + C_{Wire}$$

Sense Amp

 $C_{in,addr\ driver} = 4 \cdot C_{Cell}$

- Let's ignore the wire for now...
- What is the Input Capacitance?
 - Let's assume our address drivers can drive a bit more than a bitcell, so:

<u>Precharge</u>



© Adam Teman, 2021

Problem Setup

- What is the Branching Effort?
 - Lets take another look at the Boolean expressions:

$$WL_0 = \overline{A}_7 \overline{A}_6 \overline{A}_5 \overline{A}_4 \overline{A}_3 \overline{A}_2 \overline{A}_1 \overline{A}_0$$
$$WL_{255} = A_7 A_6 A_5 A_4 A_3 A_2 A_1 A_0$$



- We see that half of the signals use A_i and half use A_i!
- So each address driver drives 128 8-input AND gates, but only one is on the selected WL path.

$$C_{on path} = C_{nand}; \quad C_{off path} = 127 \cdot C_{nand}$$
$$B_{add_driver} = \frac{C_{on path} + C_{off path}}{C_{on path}} = \frac{C_{nand} + 127 \cdot C_{nand}}{C_{nand}} = 128$$

Number of Stages

• Altogether the path effort is: $PE = LE \cdot B \cdot F = LE \cdot \Pi b_i \frac{C_{WL}}{C_{address}} = LE \cdot 128 \cdot \frac{256C_{Cell}}{4C_{Cell}}$

The best case logical effort is

$$\Pi LE = 1$$

• So the minimum number of stages for optimal delay is:

$$PE = 2^{13}$$
$$N_{opt} = \log_{3.6} 2^{13} = 7$$

That's a lot of stages!

 $= LE \cdot 8k = 2^{13} \cdot LE$

So which implementation should we use?

• The one with the minimum Logical Effort:



New optimal number of Stages

• So now we can calculate the actual path effort:

$$PE = F \cdot \Pi b_i \cdot \Pi LE_i =$$

= 2.37 \cdot 2^{13} = 19.418k
$$N_{opt} = \log_{3.6} PE = 7.7$$



 We could add another inverter or two to get closer to the optimal number of stages...

Precharge

Sense Amp

Column Mux

Row Decoder

Implementation Problems

• Address Line Capacitance:

- Our assumption was that $C_{\text{in,addr_driver}} = 4C_{\text{cell}}$.
- But each address drives 128 gates
 - That's a really long wire with high capacitance.
- This means that we will need to buffer the address lines

Sense Amp

This will probably ruin our whole analysis...

Bit-cell Pitch:

- Each signal drives one row of bitcells.
- How will we fit 8 address signals into this pitch?

Predecoding - Concept

- Solution:
 - Let's look at two decoder paths: WL₂₅₄, WL₂₅₅



We see that there are many "shared" gates.

Precharge

• So why not share them?

Row Decoder Column Mux

• For instance, we can use the purple output for both gates...

Sense Amp

Predecoding - Method

• How do we do this?

- If we look at the final Boolean expression, it has combinations of groups of inputs.
- By grouping together a few inputs, we actually create a small decoder.
- Then we just AND the outputs of all the "pre" decoders.
- For example: Two 4:16 predecoders

$$D = dec(A_0, A_1, A_2, A_3); E = dec(A_4, A_5, A_6, A_7);$$

$$WL_0 = D_0 \cdot E_0; WL_{255} = D_{15} \cdot E_{15}; WL_{254} = D_{14} \cdot E_{15};$$

Row Decoder

F

 \mathbf{N}

D

4

σ

 A_0

 A_4

 A_5

 A_6

Predecoding - Example

• Let's look at our example:

$$WL_{0} = D_{0} \cdot E_{0}$$
$$D = dec(A_{0}, A_{1}, A_{2}, A_{3}) \qquad WL_{255} = D_{15} \cdot E_{15}$$
$$E = dec(A_{4}, A_{5}, A_{6}, A_{7}) \qquad WL_{254} = D_{15} \cdot E_{14}$$

- What is our new branching effort?
 - As before, each address drives half the lines of the small decoder.
 - Each predecoder output drives 256/16 post-decoder gates.
 - Altogether, the branching effort is:

$$B = b_{addr_driver} \cdot b_{predecoder} = \frac{16}{2} \cdot \frac{256}{16} = 128$$

• Same as before!

Predecoding - Solution

- Why is this a better solution?
 - Each Address driver is only driving eight gates
 - less capacitance.
 - We saved a ton of area by "sharing" gates.
 - We can "Pitch Fit" 2-input NAND gates.

Another Predecoding Example

- We can try using four 2-input predecoders:
 - This will require us to use 256 4-input NAND gates.



How do we choose a configuration?

- <u>Pitch Fitting</u>: 2-input NANDs vs. 4-input NAND.
- <u>Switching Capacitance</u>: How many wires switch at each transition?
- Stages Before the large cap: Distribution of the load along the delay.
- <u>Conclusion</u>: Usually do as much predecoding as possible!



Alternative Solution: Dynamic Decoders

Sense Amp



Precharge



© Adam Teman, 2021

25

Row Decoder

Column Multiplexer





Column Multiplexer

• First option – PTL Mux with decoder

- Fast only 1 transistor in signal path.
- Large transistor Count

27







© Adam Teman, 2021

4 to 1 tree decoder

Second option – Tree Decoder

28

- For 2^k:1 Mux, it uses k series transistors.
- Delay increases quadratically
- No external decode logic → big area reduction.







Column Mux Precharge Row Decoder > Sense Amp

© Adam Teman, 2021

Precharge and Sense Amp





Precharge Circuitry

• Precharge bitlines high before reads





Equalize bitlines to minimize voltage difference when using sense amplifiers



Sense Amplifiers



Idea: Use Sense Amplifier

large

 $C \cdot \Delta V \blacktriangleleft$

av



make ΔV as small

as possible

small

32 Row Decoder Column Mux Precharge Sense Amp

© Adam Teman, 2021

Differential Sense Amplifier

- Non-clocked Sense Amp has high static power.
- Clocked sense amp saves power
- Requires <u>sense_clk</u> after enough bitline swing
- Isolation transistors cut off large bitline capacitance





The Computer Hall of Fame

The machine that many of us got to know during our military service:

The DEC VAX



- 32-bit, CISC architecture, introduced in 1977
- The VAX-11/780 was TTL-based, 5MHz, 2kB cache, reaching 1 MIPS
- Known as a "minicomputer", even though it took up a whole room.
- VAX means "Virtual Address Extension", since the VAX was one of the first minicomputers to use virtual memory.
- Ran the VMS operating system.
- Many systems that were developed during the cold war (e.g., F-15, F-18, Hawk missiles, nuclear programs) still use VAX today!

Further Reading

- Rabaey, et al. "Digital Integrated Circuits" (2nd Edition)
- Elad Alon, Berkeley ee141 (online)
- Weste, Harris, "CMOS VLSI Design (4th Edition)"