Lecture 7

Gates with Lower LE
RC Model Limitations

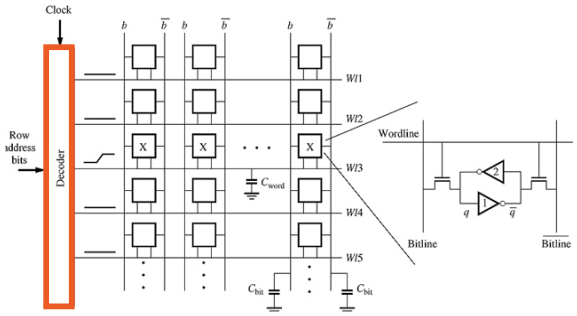
Boris Murmann
Center for Integrated Systems
Stanford University
murmann@stanford.edu

Overview

- **Reading**
  - Amrutur, Horowitz, "Fast Low Power Decoders," (section III)
  - Hodges 7.5 – Domino Logic
- **Introduction**

  We will use the decoder and logical effort to motivate some 'creative' ways to build CMOS logic gates. This gives rise to some advantages of using pulses which will lead to pre-charged gates. Some of these gates will make clear the fact that our resistor model is not perfect. For many things it works very well, but needs calibration. For example, it would be nice to know how resistance and capacitance scale with operating voltage, and what to expect from technology scaling. More serious is the fact that the model does not work in some situations, and we need a more accurate model of the transistor's IV in these situations.

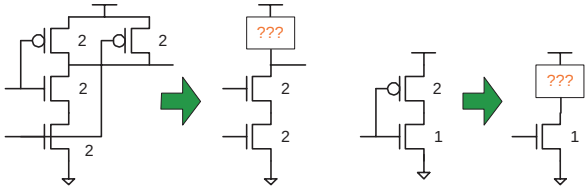
SRAM Example



- Decode speed is critical (up to 50% of RAM access time)
- Decoder fan-out is fixed (C_{cell}/C_{addr})
- For a given technology, the only way to improve speed is to reduce the logical effort of the decoder gates!

Gates with Lower LE

- How can we do better than simple static gates?
- Promising approach: Try to somehow get rid of PMOS
- Remember: Gate speed depends on intrinsic RC product

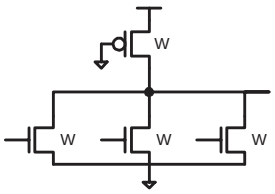


C: 4 2 3 1

- We still need some kind of pull-up device though
- Let's look at a few options...

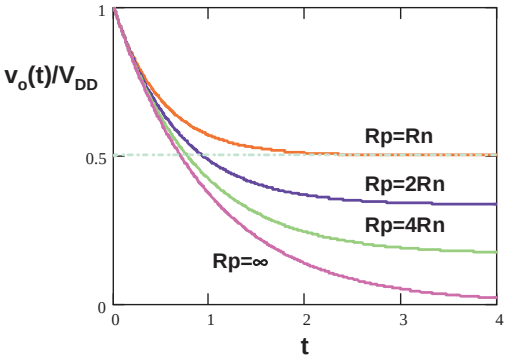
Ratio Logic (1)

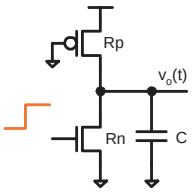
- Spend power for speed
  - Use pseudo nMOS NOR gates, not NAND gates



- What is LE for this gate?
- Using linear model "blindly" gives the **WRONG** answer. It says that the gate gets faster, since Rp and Rn are in parallel
- Why is that wrong?

Ratio Logic (2)





$$\tau = \frac{Rn \cdot Rp}{Rn + Rp} \cdot C$$

$$\frac{v_o(t)}{V_{DD}} = \frac{Rn}{Rn+Rp} + \left(1 - \frac{Rn}{Rn+Rp}\right) e^{-t/\tau}$$

- Time constant is smaller, but it takes more time to complete 50% V_{DD} transient.
- Have to choose Rp fairly large to swing to acceptable "LO" values (more later when we talk about noise margins)

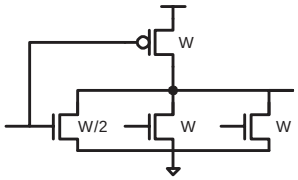
Ratio Logic (3)

- Another way to quickly analyze this situation
  - Think in terms of I_{drive}, current available to drive C_{load}
  - Use resistor to represent that current
- When you have a conflict in the current
  - Available current is the difference between the two
- Assume that pMOS is about ½ current of nMOS
  - Logical Effort for this gate is 2/3 ! (it is less than 1)
  - Where does 2/3 come from? Get ½ the current with 1/3 the input capacitance. (Roughly)
- Severe showstopper for ratio logic: Power dissipation!

+ Lower Power Ratio Logic

- No need to make the gate dissipate power all the time
  - Can have the pMOS gate connected to one of the inputs
    - One input might be available early (make its LE larger)
    - Or you can rotate which input you use in the large number of decoder gates. Then you even out the load among the address inputs. On average each input only drives ¼ of the decoders

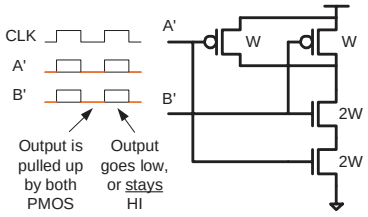
(Notice the nMOS can be smaller, Since it does not need to fight the pMOS)



- While lower power, it is still not low enough power to make it very interesting today.

Pulsed Logic (1)

- Suppose we can make the input pulses
  - A*Clk ; B*Clk, both inputs low when Clk is low, "reset state"
- Does that make it possible to make a faster gate?
  - Know BOTH inputs are low for rising output of NAND gate



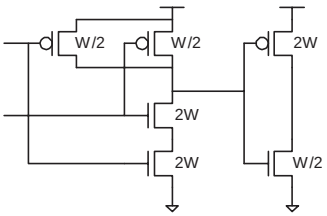
- Logical effort of pulsed NAND2 is 3/3=1!!

Pulsed Logic (2)

- This approach seems great for our decoder
  - Well defined "reset state": All wordlines low
- From a decoder perspective
  - Assert edge is critical
    - This is what causes the new wordline to rise
  - Reset edge is less critical
    - Lowers wordline when access is finished
    - Can't be too slow, but there may be a clever way to do the reset without "rippling" through the logic chain
- Can extend this thought and try to build asymmetric gates with faster assert timing

Skewed Gates (1)

- If we decide that the falling transition of the wordline can be slower than the rising edge (slow de-assert)
  - Can make the pMOS in NAND gates even smaller
  - Follow by an inverter with opposite skewing



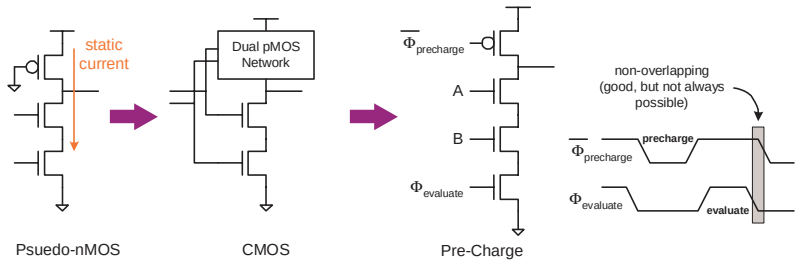
- Logical effort of above NAND2 and INV is 2.5/3=5/6
- De-assert will have twice the delay of the assert edge

Skewed Gates (2)

- Issues
  - Need to calculate rising delay and falling delay separately
  - LE of the gate will be different for the different edges
  - Useable in a situation where there is a default state
    - Critical path is sized for one edge
- The practical use of skewed gates requires some form of reset device/mechanism to prevent slow reset transition from limiting speed
- See Amrutur's paper (section III) for a comprehensive discussion
- Let's look at a few options...

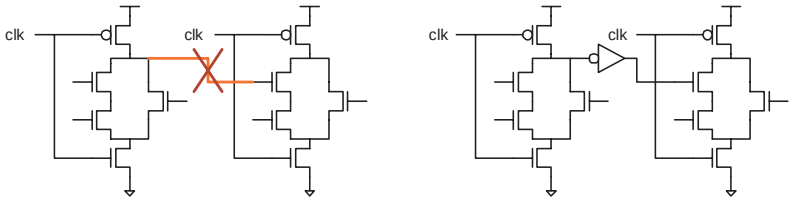
Dynamic Logic (1)

- Just like in pseudo nMOS, we remove the pMOS load from input
- Operation in two phases
  - Pre-charge output high
  - Evaluate. Execution of Boolean expression either discharges output or leaves it high
    - Caveat: Only a single low-to-high transition on the input allowed, but **NOT** a high-to-low transition during evaluation



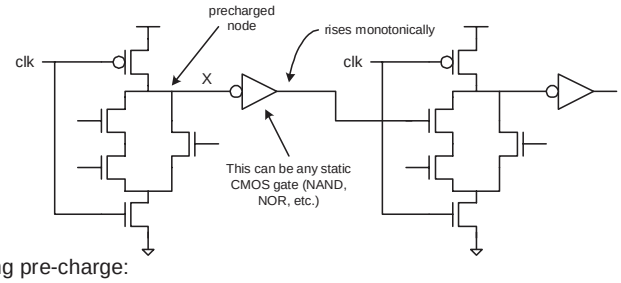
Dynamic Logic (2)

- Implement the logic function with nMOS pull-down stack as in pseudo-nMOS
- Can use a single clock signal clk = $\overline{\Phi}_{pre\text{-}charge}=\Phi_{evaluate}$



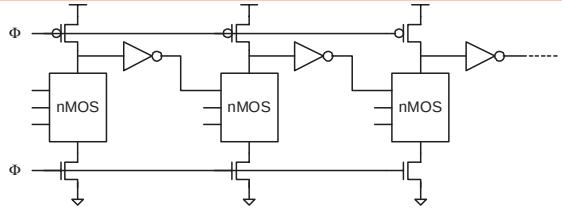
- These gates **cannot** be cascaded, even if complementary clocks are used for alternating stages
  - Constrained by low-to-high transition requirement at the input during evaluation
  - Need to put an inverting stage between them → **"Domino Logic"**

Domino Logic



- During pre-charge:
  - Output of dynamic stage (X) "pre-charged" high when clk is low
  - Domino gate output driving input of another always low during pre-charge
- During evaluate:
  - X is conditionally discharged during evaluation
  - Output of static buffer rises monotonically
  - Inverting gate can be **any** inverting static CMOS gate
  - It is impossible for buffer output to go from H-to-L during evaluation

Domino Chains

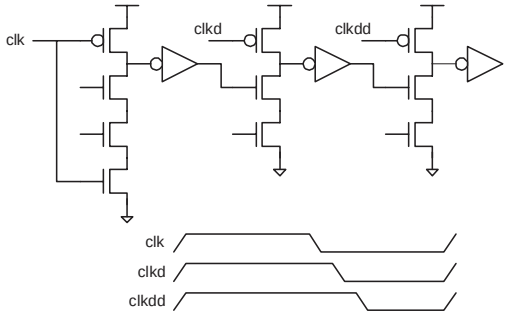


- Cascaded gates can be switched from PRECHARGE to EVAL on the same clock edge
  - Logic decisions propagate through the cascade (or chain) like a row of falling dominos
- Length of domino chains is limited by EVAL time
  - Logic must propagate to the output before Φ falls
- Inputs to domino stage must be held stable during EVAL
- Ratioless gates, i.e. output swing/input threshold don't depend on sizing
- All domino gates are NONINVERTING

Clocked Evaluation Transistor

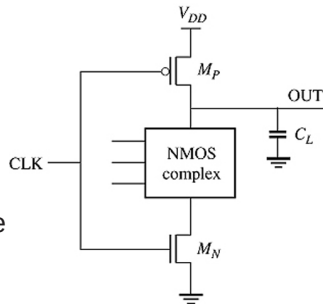
The clocked evaluation transistor is not strictly necessary.

- Can remove if all the inputs are probably low during pre-charge
  - Other domino gate outputs satisfy this condition
- Also okay if high inputs are in series with probably low input
- Delay pre-charge edge to reduce power burned at start of pre-charge



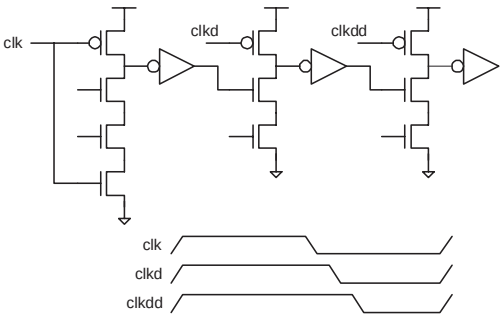
Pre-charge Problem (1)

- Many domino gates in series can evaluate in one half-cycle, so it should be easy to pre-charge a single domino gate in the other half-cycle. But…
  - The domino gate must pre-charge enough to flip the high skew gate, then the high skew gate must fall below V_{TH} by sufficient margin before evaluation starts again
  - To speed up domino evaluation, we want a small pre-charge transistor (small diffusion parasitic capacitances)
    - Makes pre-charge slow
    - High skew gate falls very slowly



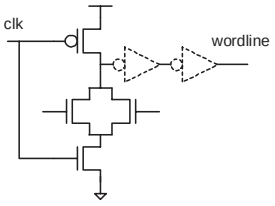
Pre-charge Problem

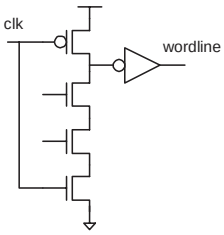
- Delaying the clock to avoid pre-charge contention in un-clocked pull-down stacks reduces pre-charge time for clkdd domino gate
- Cycles are getting shorter
- Advanced domino circuits are stretching the length of evaluation phase at the expense of pre-charge time
- Bottom line: pre-charge time is becoming an important issue. Size for roughly equal pre-charge and evaluate times



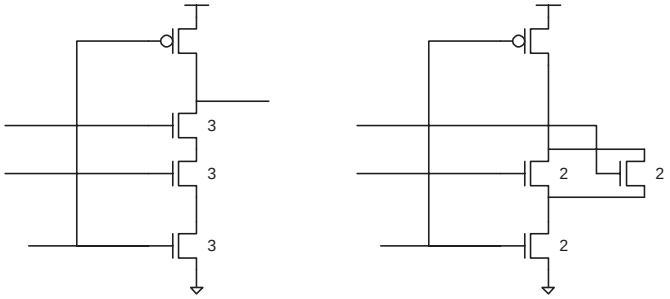
Dynamic Decoders

- Can build dynamic decoders
  - Use skewed inverters to improve the assert edge
- Typically still use NAND gates
  - Might think NOR gate would be better, since parallel pulldown should be faster
  - THE PROBLEM:
    - Think about a precharged NOR decoder
      - All the outputs start high
      - After evaluation all but one of the lines are low
      - That means N-1 outputs change value each cycle
    - In a precharged NAND decoder
      - All outputs start high
      - After evaluation only the selected output changes
      - If we invert the output, we get the wordline signal that we want





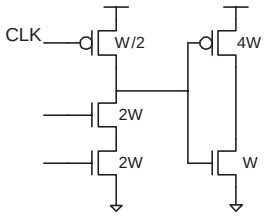
Logical Effort of Dynamic Gates



- What about the foot transistor?
  - Does it need to be sized the same?

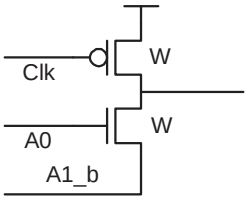
Precharged Final Decode Stage

- Generally Built with NAND gates
  - If you don't use clocked transistors
  - Can get lower logical effort
- If we used NAND gates with skewed inverters afterward
  - Assume inputs are pulses
  - Logical Effort of decode stage is Sqrt(2/3 * 5/6) = 0.75



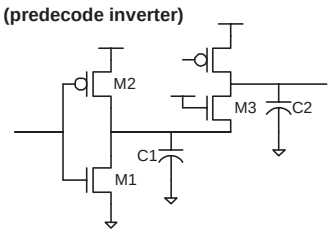
Even Faster Decode Gate (1)

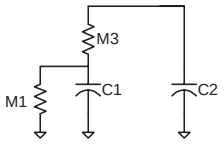
- Can use a "strange" pass transistor NAND gate ("source coupled NAND)
- The speed from A1_b is also fast
  - Buffer driving A1_b is very large
- The logical effort from A0 is small
  - 1/3 (drive strength 1 with only 1/3 of the input capacitance of an inverter)
- For a more general discussion on LE of pass transistor circuits see handout "Logical Effort Revisited"



Even Faster Decode Gate (2)

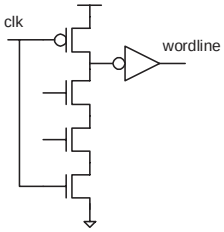
- Need to solve a more complex RC network
- "Stage" includes output inverter of predecode, wire capacitance, and the source coupled NAND gate
- Size the predecode inverter and final gate separately. Problem is much simpler if the load on predecode is much larger than on the NAND gate, which is usually the case in decoders.





Precharge Power Problem

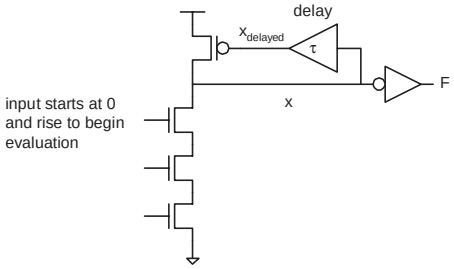
- Good News
  - With a NAND decoder
    - Most of the predecode outputs don't change each cycle
    - Most wordlines don't change each cycle
- Bad News
  - The gates don't know that
  - The precharge transistor still gets clock each cycle
  - The evaluation transistor also gets clocks each cycle
  - Clock power will be huge
- Can we only precharge the gates that change?
  - This is called self-resetting logic or
  - Post-charge logic
  - Cool, but "dangerous"

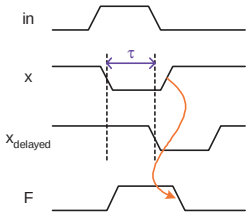


Post-Charge Logic

The dual of pre-charge logic is post-charge logic. A circuit starts in a pre-charged state. Once it evaluates, it generates the DONE signal allowing it to be pre-charged again. The DONE signal de-asserts after the circuit is finished pre-charging.

- Useful when only a small percentage of gates trip – don't need to distribute the pre-charge signals everywhere (Ex: memory decoders)
- A variant of this is self-timed circuits
- Contains a couple built in races (output is a short pulse), inputs better be pulses too





Resistor Model Re-Cap

- Is very simple
  - Allows one to quickly estimate timing and sizing
- Works for a large class of gates
  - When calibrated can work very well
- Has a number of limitations
  - Need to calibrate for each voltage used
  - Can't quantify benefits of technology scaling
  - It's gate delay estimate does not depend on input waveforms
  - Can't really analyze some circuits, e.g. pseudo-NMOS
  - **Can't answer questions about voltage levels**
- Why would we ever care about voltage levels?

Outlook: SRAM Read

- A tiny memory cell drives a huge capacitive load
- It would take a very long time to settle to V_{DD}/GND
- Need to work with small signal swings and "amplification"
- Something like this cannot be analyzed with our simple RC model...
- Therefore, we need to derive a "better" transistor model
- More in the next two lectures