ECE425: VLSI Design

Because I didn't pay attention in lecture - again

Pradyun Narkadamilli

Contents

1 Trivia

- Modern EDA: design -> Arch -> RTL -> Gate-Level (netlist) -> physical implementation (floorplan, layout, CTS, PnR) -> Post-Silicon Validation
- Vacuum tubes ruled in 20th century - large, expensive, power-hungry, unreliable.
- First point contact transistor in 1947 - came out of Bell Labs from John Bardeen and Walter Brattain
- First IC was a two-transistor flip-flop. Built in 1958 by Jack Kilby
- Transistor miniaturization drives growth rate of ICs - smaller is cheaper, faster, lower power
- 1970s mainly used nMOS topologies - inexpensive, but high idle power consumption
  - From 1980s on CMOS was predominant for lower idle power
- Moore's Law - transistor counts double every 18 months
  - frequency, processor performance increase exponentially with this
  - Wafer fabrication cost growing slower
  - Cost per function is dropping exponentially - this cost compounds per generation
  - Gate cost halved in 1.5 years
- Designer productivity not scaling with chip capacity - we need better engineers

1.1 Design Abstractions

- *Abstraction* - start with high-level design before per-transisto
- Design reuse - build on existing IPs
- Product pipelining (current project overlaps with previous)
- Timeline usually goes concept, refinement, realization,
- Dennard's Scaling Law - as transistors shrink, they become faster, consume less power, and are cheaper to make

2 Transistor Mechanics

- n-type and p-type silicon comprise nMOS and pMOS transistors
- Early processes used only pMOS (poor perf, yield, and reliability). nMOS only caused singificant idle power consumption. CMOS eventually took over due to much lower idle power consumption
- Pure silicon crystals consist of 3D lattice of attoms
  - Has 4 valence electrons, so lattice normally is of 4 neighbors

  - Can be *doped* (adding small amounts of impurities into lattice) to increase conductivity
  - Arsenic dopant will add a loosely coupled electron to the arsenic atom outside of the lattice bonds - extra free electrons make this "n-type"
  - Boron dopant has only 3 valence electrons, so you create "holes" that allow electrons to flow through the lattice by stealing electrons from nearby neighbors. p-type silicon
- Junction between p-type and n-type MOS called *diode*
  - p-type *anode*, n-type *cathode*
- ***Transistor Structure***: Silicon *substrate* will have opposite-type regions called *source* and *drain*. In between is a small separation of substrate. On top of thise separation is an insulating SiO2 layer, and a conducting polysilicon layer on top of that called the *gate* terminal.
  - Thin substrate region under gate is called *channel*
  - Apart from gate oxide, there is also a thick *field oxide* layer between M1 and substrate/source/drain/taps to prevent unintended shorts. Contacts must be explicitly etched.
  - Early transistors used to have metal gates. That's why we call them MOSFET
- Given a substrate, the "opposite type" of transistor is built within a *well*. So for p-type transistors, you would have an *n-well*.
- Substrate/Body is usually held to a strong voltage (ground or VDD) to reverse bias the p-n junction between source and drain to body
- If gate voltage is raised, then there is an electric field generated that attracts electrons to the bottom of the SiO2. Eventually electrons outnumber holes in substrate, and a conductive channel under gate acts as n-type silicon.
- Recent logic families have lowered VDD from 5V to 3.3, 2.5, 1.8, …. and now below 1V
- GND or VSS usually set to 0V

3 Designing Gates

- CMOS - you know the drill. PUN and PDN are complementary w/ DeMorgan's law
- If both PUN and PDN are *off*, then the output is in high impedance "Z" state
- If both PUN and PDN are *on*, then the output is in a *crowbarred* or *contention* state "X"
  - Causes static power dissipation, indeterminate output level - usually unwanted
- OAI gate - OR together the number of inputs listed, then NAND those groups
  - AOI reverses the coalescing order
- *Signal Strength* is how closely signal approximates the reference logical value (how close to VDD or GND)

  - Stronger signals can source or sink more current with less peturbation

- Pass transistor nMOS passes "degraded" or *weak* 1, pMOS passes weak 0
  - transmission gate can acceptably drive both, but requires both input and its complement (*double rail logic*)
- Due to nature of nMOS and pMOS transistors, SCMOS (static CMOS) gates must be inverting
- Transmission gate is *non-restoring* - if its input is noisy or degraded, output is also degraded
  - CMOS gates are *restoring*
- *TSB* is restoring, transmission gate is not. Same logical function
- *D Latch* is transparent for CLK = 1
- *Edge-Triggered Flip Flop*, or *register*, copies D to Q on rising edge of CLK
- To improve bidirectional conduction between metal layer/substrate, a heavily doped substrate tap is used to connect substrate to power rails

4 Fabrication

- Chips are inexpensive since chips can be *printed* - fabrication consists of steps in which layers of the chip are defined through *photolithography*
- Cost of chip is proportional to chip area, not transistor count - smaller transistors reduces per-transistor cost with approximately similar chip cost
  - Smaller transistors faster because electrons travel smaller distance to go from source to drain, and less energy needed because capacitances are smaller per transistor
- *Masks* - determine where on chip the layers will be drawn
  - Inverter will have 6 masks - n-well, poly, n+ diffusion (implant), p+ diffusion (implant), contacts, and M1
- *Example*: Creating N-Well and Transistor Gates
  - Oxide grown on p-substrate. Photoresist layered on top. Photoresist is exposed to light through n-well mask. We then etch oxide with hydrofluoric acid where photoresist softened. Substrate is exposed. We can now dope it. Remaining oxide is now etched off with HF.
  - Thin gate oxide grown on top of substrate, then polyxilicon grown. Polysilicon doped to make it conductive. Photoresist is layered, and poly mask is used to soften PR this time. Etch poly and gate oxide where mask is not occluded.
  - n+ region is doped with similar photoresist method. Used to be done with *diffusion* but down with *ion implantation now* (hence n-implant).

  - Do same with p+ (pimplant)
  - Field oxide is now grown to insulate the wafer and poly from metal layer. Patterned with the contact mask (e.g contacts are where to remove field oxide).
  - Aluminum is sputtered over entire waver, also filling contact cuts. Metal is then plasma etched with mtal mask to remove unnecessary metal.

5 Design Partitioning

- Digital VLSI usually has 5 levels - architecture, microarch, logic design, circuit design, physical design
  - Architecture: system function
  - Microarchitecture: how is architecture partitioned into regs and FUs
  - Logic Design: how to build FUs
  - Circuit Design: how transistors used to build logic
  - Physical Design: chip layout
- Hierarchy: *cores –> units –> functional blocks –> cells –> transistors*
- Design seaprates into three broad categories - structural, physical, and behavioraal
  - Structural is essentially the module/hardware *hierarchy*, like processor -> ALU -> leaf cell -> transistor
  - Logical would refer to more abstract design attributes like algorithm, FSM, module descriptions, etc.
- 8-bit PC, 32-bit system (registers) based on RISC
- Only 8 registers, and R0 is wired to GND
- Standard RISC architecture support - standard ALU operations, comparator, jump, and mem
  - `add`, `sub`, `and`, `or`, `slt` (set less than), `beq`, `j` (jump), `lb`, `sb`
  - *write MIPS instruction format on cheatsheet*
- Programs written in *assembly*, translated to *machine language* (binary encoding). Compiler translates from *source code* in a *high level language* to *object code*.
- Assume MIPS has its architecture, and we already have a predetermined microarchitecture
- *Two Phase Clocking* - clock and clock bar are supported by two offset generators wher ethe clocks are never overlapping. Avoids "multiple driver" issues in simpler designs
- Top level interface - processor receives both clock phases, a reset, and MemData. Spits out MemRead, MemWrite, Adr, WriteData

- General block diagram separates into a controller to manage state and a datapath (we assume this MIPS processor is multicycle). There is also a small aludec block to decode ALU configurations
- MIPS processor can be separated into submodules - controler and alucontrol can be built with standard cells. Datapath can be separated into bitslice and zipper units, then alu and inv4, etc. etc.
- HDL - hardware description language. Describe hardware using code, and run analysis on intended design prior to tapeout.
- Logic synthesis dtermines how to implement logic (NAND vs NOR vs AND, what the fanin/fanout should be, transistor sizing, etc.)
  - Logic Synthesis determines the transistor sizing and transistor-level implementation, PNR actually chooses how to turn that into an implementation
- Physical Design - floorplan, standard cells + PNR, datapaths (slice planning), area estimation
- *MIPS Floorplan* - Entire thing fits into a $(5000\lambda)^2$ square
  - Perimeter is IO pads
  - The die itself separates into two large blocks - control and datapath. In between, there is a wiring channel about 25 tracks wide
- HDL is synthesized into a gate-level netlist, which is then PNR'd with a standard cell library
- Synthesized controller area is mostly wires - design is smaller if wires run through the cells or over the cells. Smaller design means faster and lower power
  - Design snap-together cells (our cell library) for datapaths and arrays
  - Wires are planned into cells, and connection is purely on the border of the two cells
- Datapath consists of a regfile decoder, zipper (convert control signals to datapath drivers), and all the FUs
  - We can create a "slice plan" with standard cells
- After PNR, we need area estimates - can compare it to another block that has already been designed, or estimate from transistor count. Make sure to budget room for wiring.
- On most chips, verification is >50% of effort spent - we want it to be right when fabricating, and avoid unnecessary expenses from fabricating incorrect chips
  - Debugging fabricated chips are hard, since you have limited visibility into the die
- Fabrication - you tape out many copies of the final layout on a silicon wafer, then cut it into an individual die
  - Packaging will bond gold wires to the large IO pads on the die

- Chip also needs to be tested once fabricated - want to detect design errors and manufacturing errors
  - A single dust particle can kill a die – failure rate wil vary based on die size and process maturity
- Physical Design steps
  - Floorplan - where does everything generally go on the die?
  - Standard Cells - how do i build the high level blocks with cells I already have?
  - Pitch Matching - Standard cells that "snap together" must have same size along connecting edge (in our case left/right)
  - Slice Plan - given logical slices of the block (ex: datapath), we draw a high level stick-diagram-like figure showing the general vertical/horizontal routing of signals
    - Also helps estimate area

6 Detailed Transistor Mechanics

- There are different channel models in a transistor
  - *Accumulation* - negative gate voltage, holes are attracted (no channel) [cutoff]
  - *Depletion* - positive gate voltage, holes are *repelled*, depletion region [cutoff]
  - *Inversion* - Above a critical gate voltage, electrons are attracted and holes are repelled sufficiently that there now a small conductive channel with free electrons [linear/sat]
    - Once channel is pinched off, $\mathrm{I_{ds}}$ stops affecting the current - saturation mode
- Charge in the channel of a transistor: $Q_{channel} = C_g(V_{gc} - V_t)$ where $\mathrm{V_c}$ is the channel voltage
- $C_g = C_{ox}WL$
  - C_{ox} is capacitance per unit area of gate oxide
- Average carrier velocity in channel: $v = \mu E$ with electric field E and *mobility* mu
  - $E = V_{ds}/L$
- $I_{ds} = \frac{Q}{t} = \frac{Q}{L/v} = \frac{Qv}{L}$
  - In "triode" - $I_{ds} = \beta(V_{gt} - \frac{V_{ds}}{2})V_{ds}$ with $\beta = \frac{\mu C_{ox} W}{L}$
  - In "saturation", $V_{ds} > V_{dsat} = V_{gt}$ - you get $I_{ds} = \beta V_{gt}^2$

6.1 Non-Ideal and Short Channel Effects

- Velocity saturation - if V_{dsat} is lower thn both the gt and ds voltages, current saturates (won't increase anymore)
  - Happens due to excessive mobility degradation past a certain gate voltage
- *Channel Length Modulation* - Higher V_{ds} increases depletion region size around the drain, effectively shortens the channel
  - formula on page 78
- Increasing source-body voltage causes *body effect* to effectively increase the threshold volatge
  - page 79
- Increasing the drain voltage causes *DIBL*, which reduces threshold voltage
  - V_{ds} creates an electric field that affects threshold voltage
  - page 80: $V_t = V_{t0} - \eta V_{ds}$
- Increasing channel length raises threshold voltage because of *short channel effect*
- Subthreshold Leakage current and some gate leakage current exists
- Source/Drain to body diodes are usually reverse-biased, experience *some* junction leakage
  - Page 84 formula
- Increasing temperature will reduce mobility and threshold voltage - mobility effect dominates for on transistors, but in off transistors the leakage increases
  - Temperature bad basically
- Critical electric field - $V_c = E_c L$ where $v_s at = \frac{\mu_{eff} E_c}{2}$
  - $E_c = \frac{2 v_{sat}}{\mu_{eff}}$
- Carrier velocity when not at saturation characterized as $\frac{\mu_{eff} E}{1 + \frac{E}{E_c}}$
- Short channel model is on page 77 - assume you're given I_{dsat} and V_{dsat}
- Subthreshold leakage insignificant above 180nm transistors - in 90/65nm, threshold voltage is very small now so subthreshold leakage is more significant now
  - For a gate-source voltage under threshold but above 0, you have *weak inversion* - leakage increases significantly with Vds because of DIBL
  - Thermal voltage - $v_T = \frac{kT}{q}$. 26mV at room temp
  - Page 81/82

6.2 DC charateristics

- Table on 89 is good
- Skewing the ratio of beta (basically W/L balance) will skew the V_m ($r = \frac{\beta_p}{\beta_n} \neq 1$)
  - Inverter threshold (V_m) generalized as $V_{inv} = \frac{V_{dd}+V_{tp}+V_{tn}\frac{1}{r}}{1+\frac{1}{r}}$
- Noise margins - how far from the output level can we deviate on the input while retaining the deterministic behavior of the output
  - Unity gain point is where the slope of the VTC is ± 1 Pick the two points where this is the case in an inverter as your low and high levels. These points have the same slope

7 Delay

- Delay Characteristics
  - t_{pd}: propagation delay - maximum time from input crossing 50% to output crossing 50%
  - t_{cd}: contamination delay - minimum of the same thing PD measures
  - t_r: rise time - time for waveform to rise from 20% to 80%
  - t_f: fall time - time for falling from 80% to 20%
  - t_{rf}: edge rate - $\frac{t_r+t_f}{2}$
  - t_{arr}: actual time of arrival
  - t_{req}: required time of arrival to meet circuit timing
  - t_{su}: setup time
  - t_h: hold time
- Critical path can be optimized at 4 levels: arch/microarch, logic level, circuit level, and layout level
- Elmore delay - analyze the current path to understand which capacitances to accumulate, but only look at the resistance until you hit the desired node
- *Linear Delay Model*: $d = f + p$
  - d is normalized (e.g $d = \frac{d_{abs}}{\tau}$) - τ is the approximate elmore delay of an unloaded inverter (3RC for equally conductive PMOS/NMOS)
  - $f = gh$
  - f is effort delay. p is parasitic delay, intrinsic to the gate. g is logical effort, h is effective fanout or *electrical effort*, where h is characterized as $h = \frac{C_{out}}{C_{in}}$
  - g is characterized as the sum of effective gate capacitances divided by effective gate capacitances for an equally sized inverter - e.g inverter has $g = 1$. For NAND, each input drives a size n NMOS and a size 2 PMOS, so you have $\frac{n+2}{3}$ logical effort

        * Indicates how much worse a gate is at producing output current compared to inverter:w
    - p, or parasitic delay, is ratio of capacitance on output of an equally ratio'd inverter vs. the capacitance on the output of our gate

- *Minimal Path Delay*: \$D = NF$^{1/N}$ + P
    - $F^{1/N}$ is geometric mean of path effort, such that all stages have same path effort $\widehat{f}$
    - Path Logical Effort G: $G = \prod g_i$
    - Path Electrical Effort H: $H = \frac{C_{out(path)}}{C_{in(path)}}$
    - Path Branching Effort B: $B = \prod b_i$
        * $b_i = \frac{C_{onpath}+C_{offpath}}{C_{onpath}}$
    - $F = GBH$

- Given that you know F, optimal sizing for each gate (working backwards from output load) is $C_{in_i} = \frac{g_i C_{out_i}}{F^{1/N}}$
    - If final input capacitance is what we had to begin with, then we did it right…

8 Power

- TDP = Thermal Design Power
- $P = IV = \frac{E}{T}$
    - Capacitor energy to charge output of gate on rising transition is CV^2 - half is dissipated as heat
- Static Power - usually from leakage currents (subthreshold, junction, gate) and contention current in ratioed circuits (registers, SRAM, etc.)
- Dynamic/Switching Power: $\alpha f C V^2$
    - Switching capacitance is from transistors and wires
    - α is probability of a *rising* transition during that cycle
    - Usually, V proportional to f
    - For a clock signal, $\alpha = 1$. For a signal swapping once per cycle, $alpha = \frac{1}{2}$
    - Ignore short circuit power
- $P = P_{static} + P_{dyn}$, $P_{dyn} = P_{switching} + P_{short} \approx P_{switching}$
- Clock gate registers in unused block to eliminate switching activity - also saves clock activity
    - Can dynamically be determined, like by looking at instructions in pipeline
- *Voltage Domains* - different supplies to different blocks, level converters required when crossing

  - Ex: High voltage for processor (high freq), lower voltage for lower-freq I/O
- *DVFS* - run at minimum voltage/frequency to meet perf requirements
  - Reduced V_{dd} can improve P_{sw}, but increases delay
- Reduce switching capacitance with better PnR, smaller wire lengths, smaller transistors
- *Power 7 Voltage Guardband* - **maybe look at lecture??**
  - V_{dd} usually higher than needed to account for process variations, temps, etc.
  - We want to dynamically lower it while maintaining timing correctness
  - Use critical path monitor (CPM) to measure delay over a series of buffers
    * Adjust V_{dd} and frequency as needed using a microcontroller with performance controller
    * Microntroller will take in a target frequency, and then peturb the system voltage as needed. System will spit out its current frequency based on propagation delay, then microcontroller will peturb voltage in response.
- Leakage current lower for series transistors
- Power gating to turn off idle blocks to save leakage power (reduce static power)
  - The sleep transistors managing power connection have a voltage drop, which reduces performance of the gated block - transistor must be very wide to minimize impact
  - Switching very wide transistor costs lots of power and takes a while - circuit should only sleep for long periods. Frequency wakeups/sleeps bad.

9 Interconnects

- Chips are mostly made up of interconnect wires - there are many layes of interconnects
  - Wires need to be fast, low power overhead, and have low noise during transmission
- Wires have a dielectric of height h between different metal layers
- Wires have a "pitch" defined as width + spacing
  - Ex: M4 has Pitch of 28nm
- Aspect Ratio defined as thickness/width, where thickness is the physical height (z axis) of the wire.
  - Modern processes have $AR \approx 2$, old ones have $<< 1$
- Wires used to be neglibile since they had a high t and w, leading to low resistance
- AMI 0.6um process has 3 metal layers - M1 for cell routing, M2 for vertical between cells, M3 for horizontal between cells

  - Modern process have 6-10+ layers
  - Topmost layers for VDD, GND, CLK
  - Thicker layers faster
- Die-to-Die interconnects important for chiplets and 3D stacking (communication)
- π model of interconnect - R interconnect resistance with two adjacent caps of size $\frac{C}{2}$
  - Used in Elmore delay as a single-segment model
  - N segment π model computed by literally linking N copies of the basic π model for a $\frac{1}{N}$ component of the wire. Make sure to compound capacitors between the segments
- With wire resistivity ρ, $R = \frac{\rho l}{tw}$, with length l, width w, thickness t
  - again, t is the z axis, w is x axis, l is y axis
  - Longer wires have more resistance, higher horizontal cross section area reduces it
  - Can also compute $R = R_{\square}\frac{l}{w}$ - **important**
- Old process' wires aluminum, new ones have copper wires, but must be surrounded by a diffusion barrier to prevent FET damage
  - Copper wires' diffusion barriers have high resistance, Copper can get dished (dented from top) when polished. Effective resistance is actually $R = \frac{\rho}{t - t_{width} - t_{barrier}} \frac{l}{w - 2t_{barrier}}$
- Contacts/Vias have 2-20 ohm resistance - many contacts (larger contacts) have lower R
- Wire has capacitance per unit length based on distance to neighbors - wires on same layer provide adjacent capacitance, above/below layers provide top/bottom capacitance
- Wire capacitance follows parallel plate equation trend, but not exact to that since there are more complex thingies going on at the wire level
- Crosstalk/capacitive coupling - capacitance to neighbor means that when a wire switches, its neighbor also tends to switching
  - Presents as noise on non-switching wires, and increased delay onswitching wires
  - $\Delta V_{victim} = \frac{C_{adj}}{C_{gnd-v} + C_{adj}} \Delta V_{aggressor}$
  - If noise exceeds noise margin, SCMOS logic will eventually settle to correct output but this can cause extra delay and power
    - Dynamic logic cannot cover from glitches, memory/sensitive circuits may also produce wrong results
- Repeater used to improve delay when driving a wire/interconnect
  - Is usually an inverter or buffer between wire segments
  - RC delay is proportional to l^2, so breaking it into pieces means we instead get $\frac{l^2}{N}$ proportion for overall delay
    - $W = \sqrt{\frac{RC_w}{R_w C}}$ for optimal repeater width.
    - $N = \frac{l}{\sqrt{2RC/(R_w C_w)}}$

10 Combinational and Sequential Circuits

- Can separate MIPS into control/datapath, as well as memories and interconnect
  - Datapath has adders, multipliers, shifters, muxes, regs (data-driven combinational)
  - Control will have FSMs, decoders, ROM, etc. (managing state)
  - Memories will have caches, regfiles, DRAM, etc. (large storage pieces)
  - Interconnects will make up buses and external interfaces
  - Do not want to custom design & layout every component. Want subdivision
- Find repetition of design elements, optimize at each level and maximize reuse
- *Bitslice* - divide logic into slices providing per-bit outputs (and potentially state)
  - Optimize the slice
  - Optimize interconnects between slices
  - Common in older chip, where entire processor was bitsliced
- Example: IBM TrueNorth
  - 64 identical cores, and then a uniform hierarchy in the crossbar betweeen chips
- Combinational circuits with STD cells made by finding simplest/best boolean expression, then finding optimal arrangement of STD cells
- DeMorgan's Law = bubble pushing on basic AND/OR gates
  - Note that when you push a bubble through a gate, AND $\rightarrow$ OR and vice versa
- May want asymetric transistor sizing if an input is less likely to switch
  - Low switching probability inputs can be sized larger to offer less degeneration through series transistors, or sized smaller to reduce parasitic delay in parallel drivers at the output
- Reminder: logical effort is ur gate capacitance over default inverter's total gate capacitance
- *HI-skew* gate - favor rising transition (better PUN conductance)
- *LO-skew* gate - favor falling transition (better PDN conductance)
- Skew factor, where $s < 1$, is how many times smaller than normal the noncritical path is sized
- g_u is relative to an inverter with the same PMOS drive strength, g_d is relative to inverter with same NMOS drive strength
- CMOS expensive (large area due to NMOS + CMOS)
  - Can consider ratioed circuits like pseudo nMOS - one PMOS with grounded gate, NMOS PDN will be structured like normal, but sized large enough to contest PMOS

  - Dynamic circuits precharge/pre-discharge first then evaluate later (selectively discharge or charge)

- Latch is level-triggered by CLK, FF is edge-triggered by CLK
  - FF is basically two latches in serial
- 2-phase transparent latches: sequential latches with logic in between driven by non-overlapping clock phases
- Pulsed latches - duh
- Max delay is propagation delay (t_{pd}), Min delay is contamination delay (t_{cd})
  - for an ff, we have t_{pcq} and t_{ccq} for contamination and propagation of clock to q

11 Datapath Idiocy

- Full Adder accepts carry-in, half adder does not
  - Default implementation is C_{out} as a majority gate, S is a 3-way XOR
- Critical Path is usually carry-in to carry-out - want to minimize this
- Generic N-Bit adder is called "Carry-Propagate Adder"
- Carry-Ripple Adder: cascaded FAs
- If we factor the summation to be $S = ABC_{in} + (A + B + C_{in})\overline{C_{out}}$, a CMOS implementation with cascaded C_{out} is called a **mirror adder**, since PUN and PDN look the same
  - Can consider the carry-out a "Minority Gate" in this case
  - In this implementation, S computation delay is longer but since critical path is in computing C_{out} this is okay, we're just balancing our critical paths now
  - Can use asymmetric transistor widths to prioritize the transition on C_{in} - minimize transistor sizing on all transistors in the sum logic (minority gate) to lower branching effort, use large transistors on critical path
  - Can also reduce routing size/amount on sum output to lower interconnect capacitance
  - Remove inverters on critical path (C_{out}) and alternate positive/negative logic
- Can use transmission gates to form XOR and MUX structures such that C_{in} acts as a select on two MUXes selecting between two options for S and C_{out} respectively
  - Page 434
- *PGK* - you can either *propagate* carry to carry, *generate* a carry-out based on inputs, or *kill* the carry-out if double-0 input for AB
  - Propagate is an XOR, generate is an AND, kill is NOR (all on AB)

  - Generate and Propagate can be generated for *groups* spanning $i : j$ - this generates to finding intermediate G and P bits then recursing

- Base cases for grouped G, P are $G_{i:i} = G_i$ and $P_{i:i} = P_i$
  - $G_{i:j} = G_{i:k} + P_{i:k}G_{k-1:j}$, $P_{i:j} = P_{i:k}P_{k-1:j}$
  - Carry-in for some bit i is simply $G_{i-1:0}$
  - Can form an FA by simply generating the PG bits and then saying $S_i = P_i \oplus G_{i-1:0}$
  - Can cascade the FAs with PG bits to form a CA
  - Total Delay of Sum is $t_p = t_{pg} + (N-1)(t_{AO}) + t_{xor}$ - t_{pg} is time to generate PG from each AB pair, t_{xor} is final sum computation, and t_{AO} is the AO21 gate used to combine $G_{i:0}$

- PG CRA latency: $t = t_{pg} + (N-1)t_{AO} + t_{xor}$
- PG CSA latency: $t = t_{pg} + 2(n-1)t_{AO} + (k-1)t_{mux} + t_{xor}$
- PG CLA latency: $t = t_{pg} + t_{pg(n)} + [(k-1) + (n-1)]t_{AO} + t_{xor}$
- PG CSeA latency: $t = t_{pg} + [n + (k-1)]$

12 Memory

- Metastability - Latch has a metastable state at input V_m where it will also output V_m, and will settle to either 0 or V_{dd} when noise shifts it
- *Volatile* - data lost when power off
- *RAM* - read/write memory, can be dynamic with capacitors or static with bistable circuits
- *ROM* - read/write memory
- *CAM* - content addressable memory
- Memory usually structured as arrays, with 2^n rows and 2_m columns - e.g rows with columns is words with bits
  - Commonly SRAM and DRAM
- SRAM is still relatively fast access times with better density than flip-flops, but you can only access one cell at a time, whereas registers have random access/multiple access inherently
  - Compatible with standard CMOS
  - Slower than FFs, faster than DRAM
  - Easier to use/better timing than DRAM

12.1 SRAM

- 12T SRAM is basically a latch with a extra enable on its input and output on a common single bitline
- 6T SRAM has differential column lines and a single word line
  - To read, precharge both bitlines to V_{dd}, see which one drops.
  - If bit drops, $Q = 0$. If bit_b drops, then $Q = 1$
  - Driver needs to be stronger than access transistor to prevent corruption on read
  - To write $Q = 1$, pre-discharge bit_b to 0, and precharge bit to V_{dd}
  - Dual non-overlapping phases for operations: phase 2 precharge, phase 1 operation, phase 2 precharge
  - Data PMOS weak (to be overwritten), Data NMOS (e.g drivers) should be strong (to fck with bitline), access transistors somewhere in between
- SRAM requires row decoders - using address bits (row address), unpack into wordline high/low signals
  - Word generation should meet the clock constraints of the operation phases
  - Making everything POS or SOP expensive, factor out NAND gates into tree structure
    - This factorization is called **pre-decoding** - or in Rosenbaum terms, non-monkey design
- Large SRAM bad in terms of wire length, noise, and delay - separate it into subarrays and banks to mitigate. Subarrays are formed with banks
  - Banks can use the same wordline decoder and column decoder between different banks - can also share wordlines and column lines as a result - wordline signals must be gated with the clock signals and bank select signals

12.2 DRAM

- Each DRAM cell is only one transistor - one terminal is wired to GND, one terminal is wired to a shared bitline
  - The gate of these transistors are tied to a wordline
  - The capacitances of the cell can be charged through the bitline
  - Assume that capacitance is between the source terminal of the transistor and GND
  - Capacitors discharge over time due to subthreshold leakage and other shii, so much periodically *refresh* rows by reading and then writing back (every 64ms on DDR4)
- DRAM subdivided into banks, which are formed by subarrays, which are formed by an array of 1T cells attached to bitlines and wordlines in a cubicle fucking grid because lazy design idfk improve your goddamn custom design Micron L ratio bozo fuck off
- *Basic DRAM Operations*

  - ACT: Activate - opens a row (loads into row buffer)
    * Bit lines at $\frac{V_{dd}}{2}$ already, decode the row, read row to buffer
    * ACT->PRE time is tRAS (row access strobe)
    * ACT -> R/W time is tRCD (Row to Column Delay)
  - CAS: Column Access - read or write to a column in the row buffer
    * R -> Data is tCL (CAS Latency, or Column Access Latency)
    * W -> Data is tCWL (CAS Write Latency)
  - PRE: Precharge - close the row, write back from the row buffer to the cells, precharge bitlines to $\frac{V_{dd}}{2}$
    * PRE-> CT time is tRP (Row Precharge Time)

- Banks are individually addressable, and the smallest addressable unit from the memory controller - can have one or multiple subarrays
  - Banks form independently addressable Ranks
    * Each has an independent command and data bus, but the same memory controller is issuing them so…
  - Ranks are grouped into DIMMs, which are physical memory models
  - Indpendent memory channels used to transfer data from DIMMs - each has its own memory controller. Memory channels are fully independent.

- Memory controller used to order and service R/W requests efficiently from host end, issues low level ACT PRE CAS commands to the DRAM
  - Has a memory command queue and timing scorebaord per-bank
  - Manages power and row refresh as well
  - Many scheduling policies: FCFS, FR-FCFS, request priorities, etc.

- Open Page - keep open until another row is opened
  - More hits, but if the wrong row is open you now need to PRE, then ACT instead of just ACT

- Close Page - close after every memory operation
  - Fewer hits, but less likely to run into precharge latency at the beginning of a request

12.3 HBM

- Kunjesh memory frfr - used in gamer GPUs
  - Apparently also being used in server CPUs now

- HBM consists of 3D-stacked memory dies with very high bandwidth - 1024-8192 bandwidth vs 64bit DDR
  - There are multiple stacks

  - CPU less likely to benefit from high bandwidths compared to Kunjesh
- Way more channels than DDR5 - 32-96 channels vs DDR5, due to higher parallelism in GPU computation
- No ranks, independently accessible pseudo-channels
  - Requires per-bank refresh like LPDDR
- Fixed size, so bad for computing which requires flexibility in how much you have (pluggable DIMMs)
- HBM page policy has more banks, e.g more misses, e.g fewer conflicts so maybe have open-page
- HBM is limited in size, and not all models can add more DIMMs

12.4 Shit Registers

- You know the FF implementation
- You can have a dual ported SRAM with two counters - one counter is write address, one is read address. On every "shift", a new value is written to write counter, and read counter is read out and both counters increment.
  - Shift signal acts as write enable in a sense
  - Read counter resets to last entry, write counter resets to first entry (or vice versa, just needs to be fully oppositional)
- Tapped delay register is essentiallly a bunch of shift registers in serial with muxes in between - shift registers are receding powers of 2
  - All the delay signals selected are summed

12.5 CAM

- 6T SRAM cell but with 4 extra transistors underneath to match em. Those transistors are enabled with the bitline
  - Matchline either precharged or pulled high with pseudo-NMOS topology - if pulled down, 'tis not a match
  - Key is shown on bitlines
- Wordline raised for write
- Miss is essentially a NOR of every matchline
- If both bitlines are low during matching operation, that bit is considered a don't care - can potentially help create wider CAM arrays?
- When in a two-array design, like a TLB, the SRAM wordlines are quite literally the matchlines ANDed with a delayed version of the clock, called a "strobe"

12.6 ROMania

haha get it the pun is funny i'mma kms bye

- Basic kind of ROMs are not programmable, and are typically pseudo-NMOS topologies - the wordline will be attached to a weak PMOS pull up transistor.
- NOR ROM: each bitline has many parallel transistors, each one hooked up to either VDD or GND on the other end. When wordline is held high, all the transistors turned on act as pass transistors. Row decoder will be active high.
- NAND ROM - Every grid position will either have an NMOS transistor or it won't. When a wordline is pulled low, on each bitline if there is an NMOS there then the current cuts off and the output is high. If there is no transistor there the NMOSes will continue to conduct, and output remains low. Therefore, the row decoder is designed to be active-low.
- NOR ROM is faster, but takes up more area. NAND ROM is compact but slower.
  - NAND ROM doesn't have much metal routing, it's just a bunch of series transistors (or lack thereof)

12.7 PLA

- PLA is programmable logic array
- lets you implement boolean expressions as SOP (inputs are inverted, go to AND, then recombined in NOR)
- Formally: AND plane computes minterms, NOR plane computes outputs, inputs are inverted prior to the AND plane
- Basic PLA will use a pseudo-NMOS design

13 Process Variation

- Variation sources:
  - Process: process variation at manufacturing
  - Voltage: supply voltage at operation
  - Temperature: operating temperature based on environment
- Designed needs to make sure that chip can function in shitty PVT
- Categories of Silicon Variation: Lot to Lot, Wafer to Wafer, Die to Die, Within Die
  - Caused by uncertainties in manufacturing stages - can be spatially correlated (*systematic*) or pretty random
  - Affects your V_t and your effective channel length
  - Design corners for Fast, Typical, and Slow are used for both NMOS and PMOS

- EDA tools let you model uncertainties from process variation
- Silicon Aging - processors easily last 7-12 years before aging failure
  - *MTTF* mean time to failure - much longer MTTF than expected service lifetime for silicon chips usually
  - Notable Effects that Cause Breakdown:
    * *Negative Bias Temperature Instability* - dangling bonds called traps develop at silicon oxide interface. V_t for PMOS will incrase over time
    * *Hot Carrier Injection* - HCI will trap "hot" electrons in gate oxide, gradually increasing NMOS Vt
    * *Time Dependent Breakdown* - TDDB gradually increases leakage over time due to electric field at gate oxide. Can potentially short circuit the gate
- How to account for these effects and variations?
  - Voltage Guardband: Set conservatively higher V_{dd} to account for process variaton and aging
  - Frequency Guardband: Set conservatively lower f to provide timing slack
  - Fault Tolerance/Redundancy - Duplicate key circuit components, can swap to a working one if error is dynamically idenitified