ECEN454 refsheet

https://github.com/joshua-wright

specifically for final

- exam 1 was on March 8 (2017.03.08)
- nothing specifically from previous exam

Propagation vs Contamination delay

- contamination delay (t_{cd}): time from input change to **any** output changing value
- kind of a best-case delay or smallest possible delay
- time counted when input crosses 50% of logical high voltage level
- propagation delay: time from when all inputs are stable to when all outputs are stable
- kind of like a worst-case delay

static vs dynamic power

- static power
    - consumed when circuit is not active
    - e.g. leakage current
    - static because no inputs are changing
- dynamic power
    - power consumed due to gates switching and stuff
    - e.g. charging/discharging parasitic caps, etc. . .

Latch vs Flip Flop

- Latch is level-sensitive, flip-flop is edge-sensitive
- register is flip-flop # Latch/flip flop designs
- pass transistor latch
    - pros: tiny, low clock load
    - con: voltage drop across is V_t, it's non-restoring
    - con: not really a latch because doesn't hold signal
- transmission gate
    - pro: no V_t drop
    - con: requires inverted clock
- inverting buffer
    - TODO
- tristate feedback
    - first complete latch here
    - risk of backdriving: downstream could change state if it is very strong
- buffered output
    - no backdrifing problem

- – widely used
  - – rather large and slow though, and large load on clock
- a flip-flop is just two latches back-to-back

Metastability

- stable works if it's not at a strong high or low but will settle somewhere
- metastable is where it might sit there, but if it must settle, you don't know which one it will settle to when perturbed

2017.03.27

Sequential Circuits

- sequential means that the circuit holds state: the output depends on both current and past input
- to model a state machine, you can unroll it to [register] $\rightarrow$ [combinational] $\rightarrow$ . . .
  - – this way you can use regular timing stuff for it: arrival time, slack, etc. . .
- Mealy FSM: output of circuit is from combinational logic
- Moore FSM: output is from registers
- pipelined circuit: uses registers to hold state between clock cycles, because not all of the combinational logic can happen fast enough to work in a single clock cycle
  - – pipelined circuits can use flip-flops or latches?
- the clock consumes 20-30% of the power on a chip
- the whole reason that registers are needed is because data (signals) moves through components at non-constant speed
- reset
  - – can be sync or async
  - – force low output when reset is high
- sequencing: it is (generally) equivalent to split the sequential logic in two, and then use two latches (one halfway through) instead of one large section of sequential logic with flip-flops at either end
  - – this is called two phase clocking
  - – you have to make sure the middle latch operates on clock-bar instead of clock

symbol	definition
t_{pd}	logic propagation delay
t_{cd}	logic contamination delay
t_{pcq}	Clk$\rightarrow$Q propagation delay
t_{ccq}	Clk$\rightarrow$Q contamination delay
t_{pdq}	D$\rightarrow$Q propagation delay
t_{setup}	setup time of flip-flop/latch
t_{hold}	hold time of flip-flop/latch

- timing:
  - – D$\rightarrow$Q delay only makes sense for latches, since for flip-flops it is simply one clock cycle
- sequencing overhead and max delay:
  - – T_c: cycle time
  - – need for combinational logic to be fast enough
  - – for single phase flip flop: $t_{pd} < T_c - (T_{setup} + T_{pcq})$
  - – for two-phase latch: $t_{pd} = t_{pd1} + t_{pd2} < T_c - 2t_{pdq}$
  - – max delay is dictated by cycle time and sequencing overhead; sequencing overhead does not effect minimum delay
- minimum delay:
  - – for single phase flip flop: $t_{cd} > t_{hold} - t_{ccq}$

* combinational stuff must be slow enough that it doesn't violate hold time of second flip flop
    – for two-phase latch: $t_{cd1}, t_{cd2} > t_{hold} - t_{ccq} - t_{non-overlap}$
      * hold time applies twice each cycle, once per each combinational circuit
      * $t_{non-overlap}$: phase between clock signals?
  • time borrowing: for two-phase latch-based system, you can borrow time from one segment to the other while the latch is transparent
    – works (is possible) as long as the total circuit still completes within one clock cycle
  • clock skew:
    – when uncertain: causes tighter required CL propagation delay and minimum CL contamination delay, and reduces opportunity for time borrowing.
    – when know constant skew: can help with same effect as time borrowing, except can also have effect for flip-flop based systems
  • skew tolerance:
    – flip-flop circuits are not very skew tolerant because the setup and hold times define a specific window of time in which the data must arrive
    – latch circuits are more hold-time-tolerant because the data can arrive any time that the latch is transparent
  • if setup times are violated, you can reduce clock speed to fix it
  • if hold times are violated, it won't work at any clock speed

2017.03.29

Clocking

  • H-Tree (clock distribution network)
    – laid out in such a way that the delay from the center to any leaf node is equal
  • Grid clock distribution
    – grid on two or more levels
    – ensures low local skew, but there could still be larger chip-wide skew
  • generally, the longer the delay between two point, the larger the worst-case clock skew
  • Domino circuit: TODO
    – a kind of dynamic logic
    – when the clock pulse cascades through the circuit, activating the pMOS at each stage?

2017.04.03 (+more)

Memory

  • random access or sequential access, or content addressable
  • CAM: content-addressable memory
    – basically like an associative array, or database
    – designed to search all memory at once in a single operation (wikipedia)
    – used in caches
  • SRAM:
    – 12T and 6T cell designs
    – 12T design: just a latch connected to a bitline
      * boring and impractical because it's too large
    – 6T design: cross-coupled inverters (3T each) +2T for bit and bit-bar lines
      * we focus on 6T design
    – read: pre-charge bit and bit-bar, raise wordline, read result
      * (need to pre-charge or else read is destructive?)
      * need sense amplifiers at the end of the bitlines because the signal will be weak, and the parasitic capacitance of the bitline depends on how many cells are 0 or 1
    – write: drive data onto bit and bit-bar and raise wordline to write
    – cross coupled inverters must be much larger than the bitline transistors so that it doesn't flip while reading

- per each column, you need:
  * piece of decoder
  * bitline sense amplifier
  * column multiplexer
- multi-port: means you can read from and write to the same cell in the same clock cycle and it works
  * implement by having multiple sequential read/write per cycle?
- decoders:
  - must be pitch-matched to SRAM cell width for layout efficiency/simplicity (requires very skinny gates)
  - large ($n > 4$) decoders are inefficient, thus it is often helpful to use predecoding: factor out common gates into a second stage of decoders.
    * smaller area as alternative, but same logical effort (since same number of gates traversed)
- twisted bitlines: bitlines are long and right next to each other, so there will be significant crosstalk and parasitic cap
  - solution: swap adjacent bit and bit-bar lines occasionally
  - reduces Miller factor (MCF)
- DRAM:
  - one transistor and one capacitor
    * transistor gates capacitor
  - no charge is 0, charge is 1
  - open transistor to read/write
  - read is destructive
  - must be refreshed periodically due to leakage
    * thus, it's always consuming power
- shift register:
  - can be implemented with cascade of flip-flops
    * can lead to hold time violations because no delay from clock to next data line? (TODO what? how?)
    * not very dense
  - store data in SRAM and store begin/end pointers
    * more dense than individual flip-flops
- Tapped Delay Line
  - shift register with programmable number of stages
  - implement using sequence of flip-flops, each with a bypass MUX from data in to out
- Queues: are a thing (TODO)
- ROM: Read Only Memory:
  - mask-programmed ROMs are one transistor per bit
- PROMs, EPROMS, etc
  - generally burn out specific fuses to program ROM
  - E for erasable

2017.04.10, 2017.04.12

Low Power Design

- not so important when running from a battery since batteries store energy, not power
- peak power
  - determines power/ground wiring and packaging limits
  - impacts signal/noise ratio
- reduce dynamic power: (AKA active power)
  - use smaller transistors
  - clock gating
  - reduce V_{DD}
  - reduce frequency
    * but be careful because that will make the computations take longer
- reduce static power
  - power gating

- selectively use lower V_t devices
        - stacked devices
        - body bias
- clock gating: disable the clock signal to an expensive component
        - eliminates that component's dynamic power (but of course you can't use it)
        - doesn't change static power
- voltage islands: different V_t for different devices
        - allow low V_t devices to run slower because they don't need to be fast (maybe they're not in the critical path)
        - often different voltage levels are alternated row by row in standard-cell layouts
        - you must convert from one logic level to another, because high/low will be different
                * sometimes it's possible to avoid conversion, e.g. by using a buffer that you would have anyway, with inverters built from half high-V_t transistors and half low-V_t transistors
        - there are all kinds of tricks for determining which components should be high/low voltage and where to put voltage islands and level converters
- DFVS: Dynamic Frequency and Voltage Scaling
        - requires:
                * programmable clock generator (PLL)
                * supply regulation loop to for setting minimum V_{DD}
                * software support for deciding best stable frequency and voltage for given load and performance goals
        - voltage regulation loop
                * voltage divider would be horribly inefficient
                * instead, rapidly charge/discharge capacitor
- stacked devices:
        - stacking devices reduces leakage
        - because leakage is exponentially proportional to V_{ds}
- body bias AKA dual V_t
        - changing V_t at runtime is called Adaptive Body-Biasing (ABB) or Dynamic Threshold Scaling (DTS)
                * requires triple well fabrication process, because what would have been lowest level substrate must itself sit inside a well that will be biased
        - design with low and high V_t
        - low V_t is faster than high V_t but consumes 10x more power
                * because reducing V_t increases sub-threshold leakage exponentially
                * but reducing V_t decreases gate delay (increasing performance)
        - to increase V_t, put a negative bias on V_{SB} of nMOS
                * use DC charge pump to get desired voltage
- power gating: use sleep transistors
        - basically cut off V_{DD} from power-hungry circuits when not in use
        - in sleep mode, the sleeping transistors are stacked with the transistors that are causing them to sleep, thus reducing leakage even more
- MTCMOS sleep transistors
        - combines power gating and dual V_t
        - TODO what is this?

2017.04.17

Packaging

2017.04.26

Testing

- logic verification is very important
        - consumes 50% of total IC development time