Discussion Section 6

Sean Huang

February 19, 2021

Berkeley
UNIVERSITY OF CALIFORNIA

Bit-Serial Interface

- Accepts packets coming in with a specific format

  `<src addr 4B><dst addr 4B><payload 512B><chksum 1B>`
- Design block that performs two checks
  - Destination check: Make sure our addr is the dst addr
    - Match dst addr field with local addr
  - Checksum: Make sure packet is not corrupted
    - If packet is fine, sum of all the previous bytes should be complement of checksum
      - *This was an error in the homework (I said bit-wise instead of byte-wise)*
      - *Sum should have been done 2's complement instead of 1's complement*
- Processing should be done in-stream (i.e. don't buffer the whole packet)

Bit-Serial Interface

- Make a “story” out of the specification
  1. Packet comes in
  2. Wait for dst field…
  3. Check against local addr
     1. If check fails, wait until end of packet then repeat
  4. Wait for end of packet…
  5. Check against checksum
  6. Repeat

Bit-Serial Interface

- Make a "story" out of the specification
  1. Packet comes in
  2. Wait for dst field…
  3. Check against local addr
     1. If check fails, wait until end of packet then repeat
  4. Wait for end of packet…
  5. Check against checksum
  6. Repeat

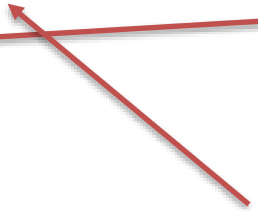
This can be an IDLE state

Design performs different actions in these steps, so these can be distinct states

Return to IDLE

Bit-Serial Interface

- Make a “story” out of the specification
  1. Packet comes in
  2. Wait for dst field…
  3. Check against local addr
     1. If check fails, wait until end of packet then repeat
  4. Wait for end of packet…
  5. Check against checksum
  6. Repeat



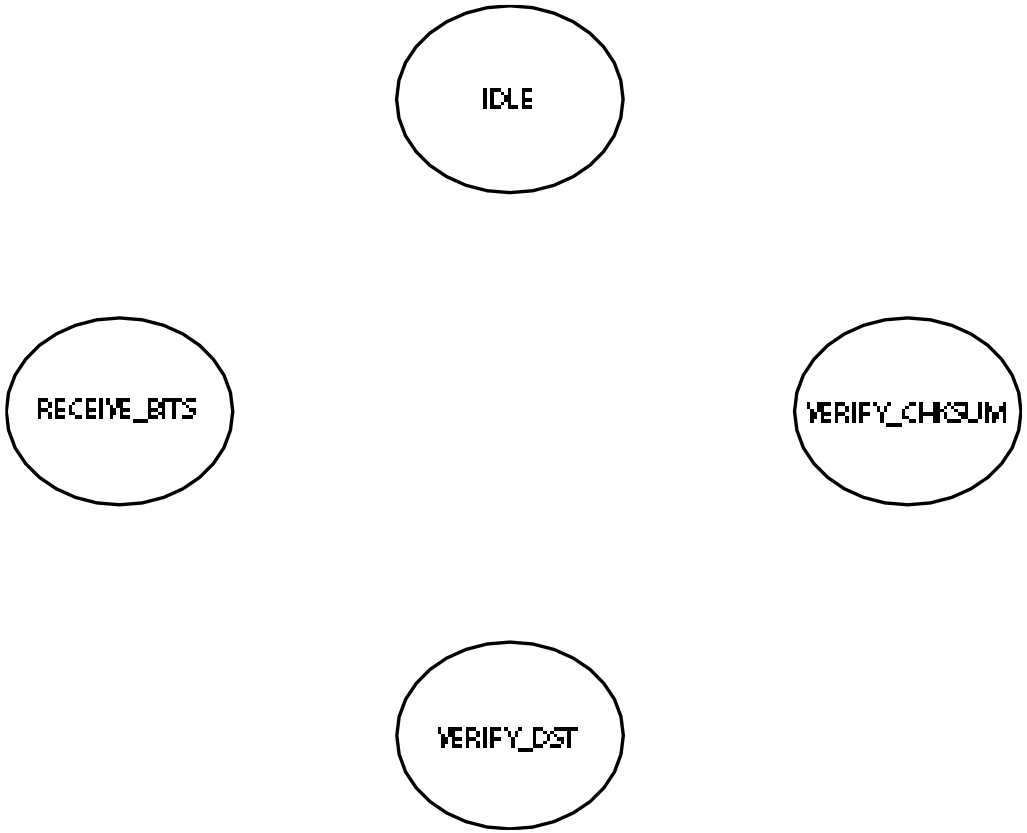
Both of these steps involve just waiting, but for different periods of time, so can be the same state

This “mini-step” is also just a wait step so this can also reuse the same state as the other waiting steps

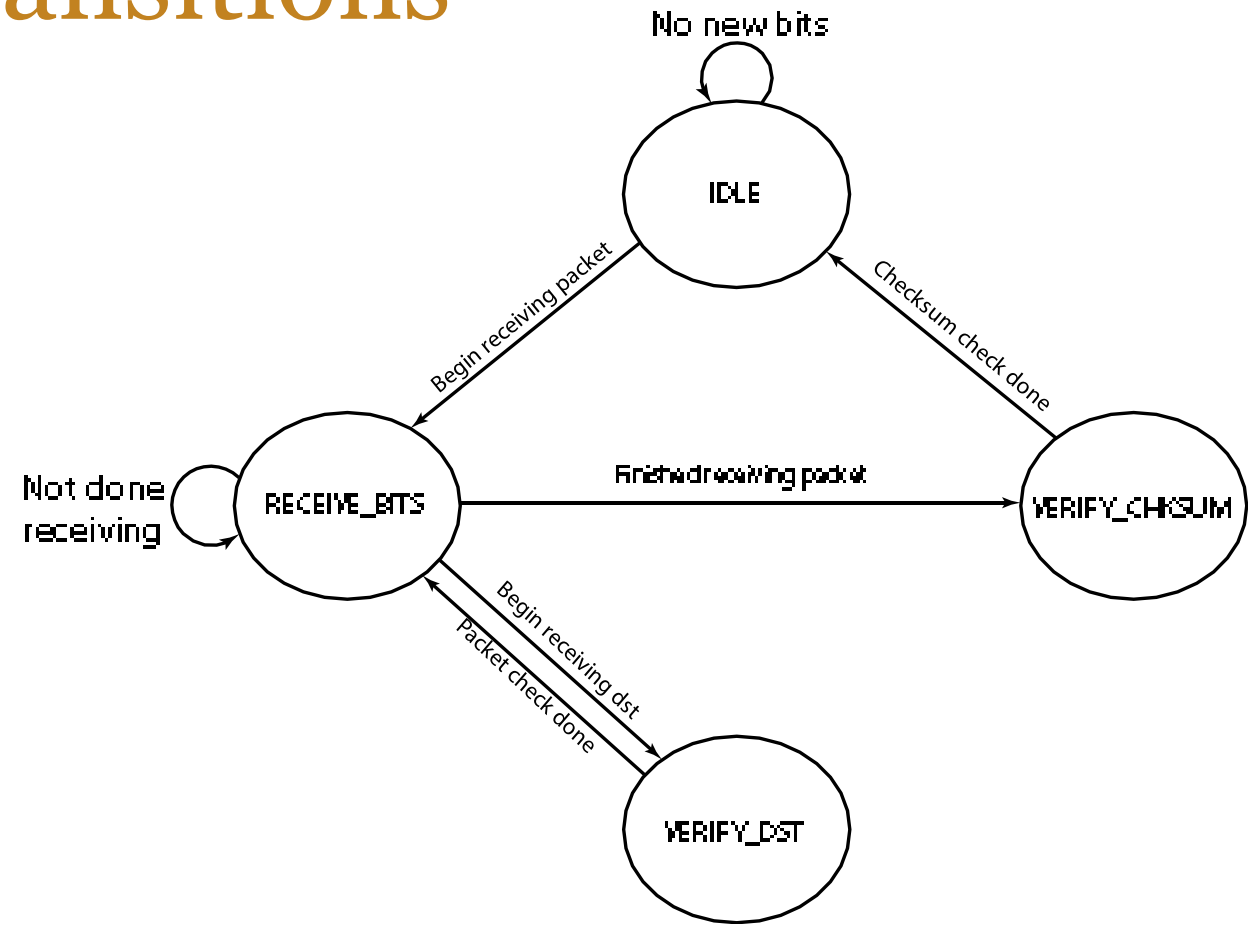
Bit-Serial Interface

- List of states
  1. IDLE (Waiting for packet to arrive)
  2. RECEIVE_BITS (Waiting until relevant field arrives)
  3. VERIFY_DST (Perform dst check)
  4. VERIFY_CHKSUM (Perform checksum check)

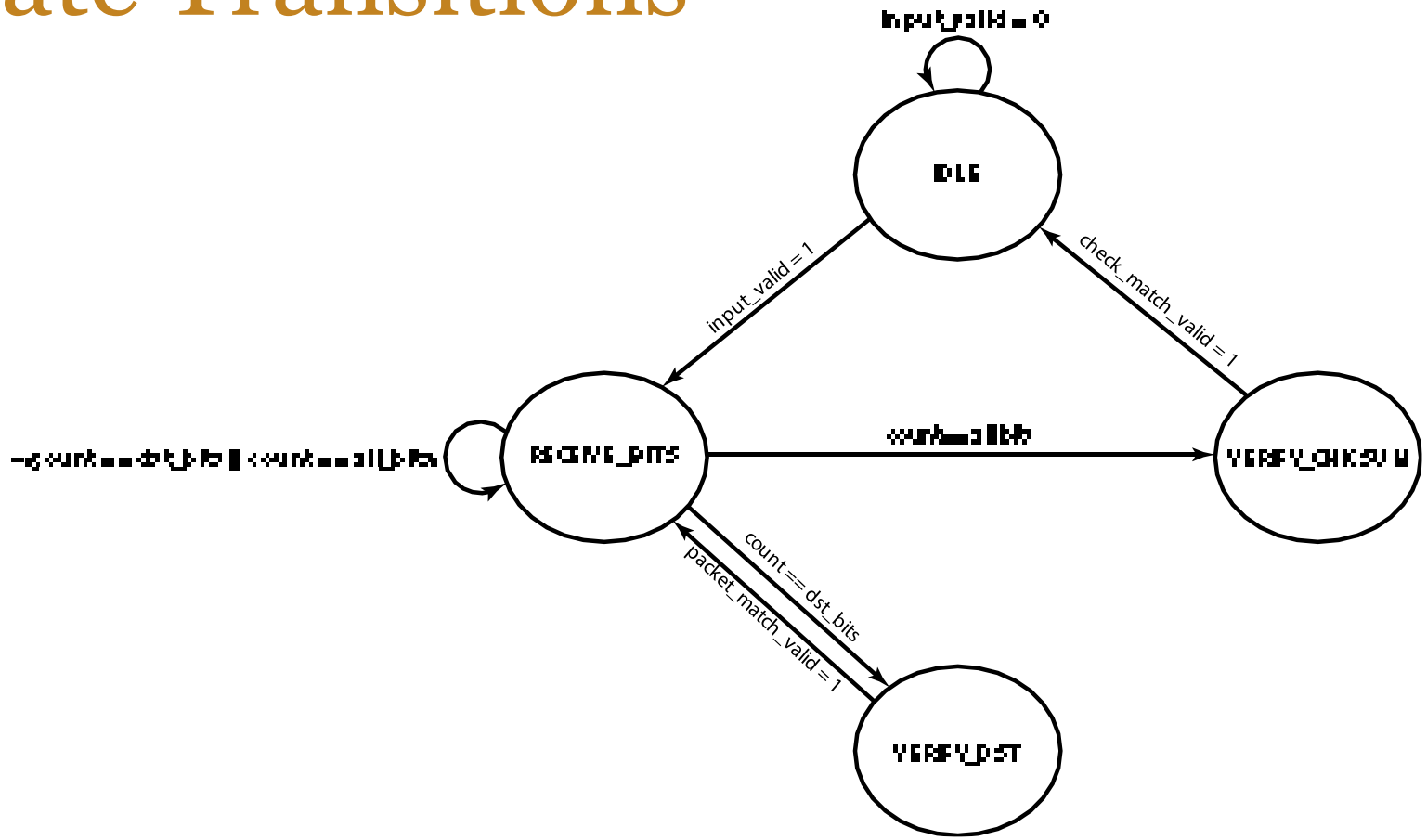
Bit-Serial Interface



State Transitions



State Transitions



IDLE State

- Actions:
  - Wait for new packet to arrive
- Outputs:
  - Both valid signals should be low
  - packet_match and check_match low

RECEIVE_BITS State

- Actions:
  - Start counting bits from packet as they come in
    - Begin summing bytes of the packet for the checksum check later
  - Move to packet check after finished receiving src field
  - Move to checksum check after finished receiving entire packet
  - If packet check failed, wait until end of packet
- Outputs:
  - Both valid signals should be low
  - packet_match and check_match low

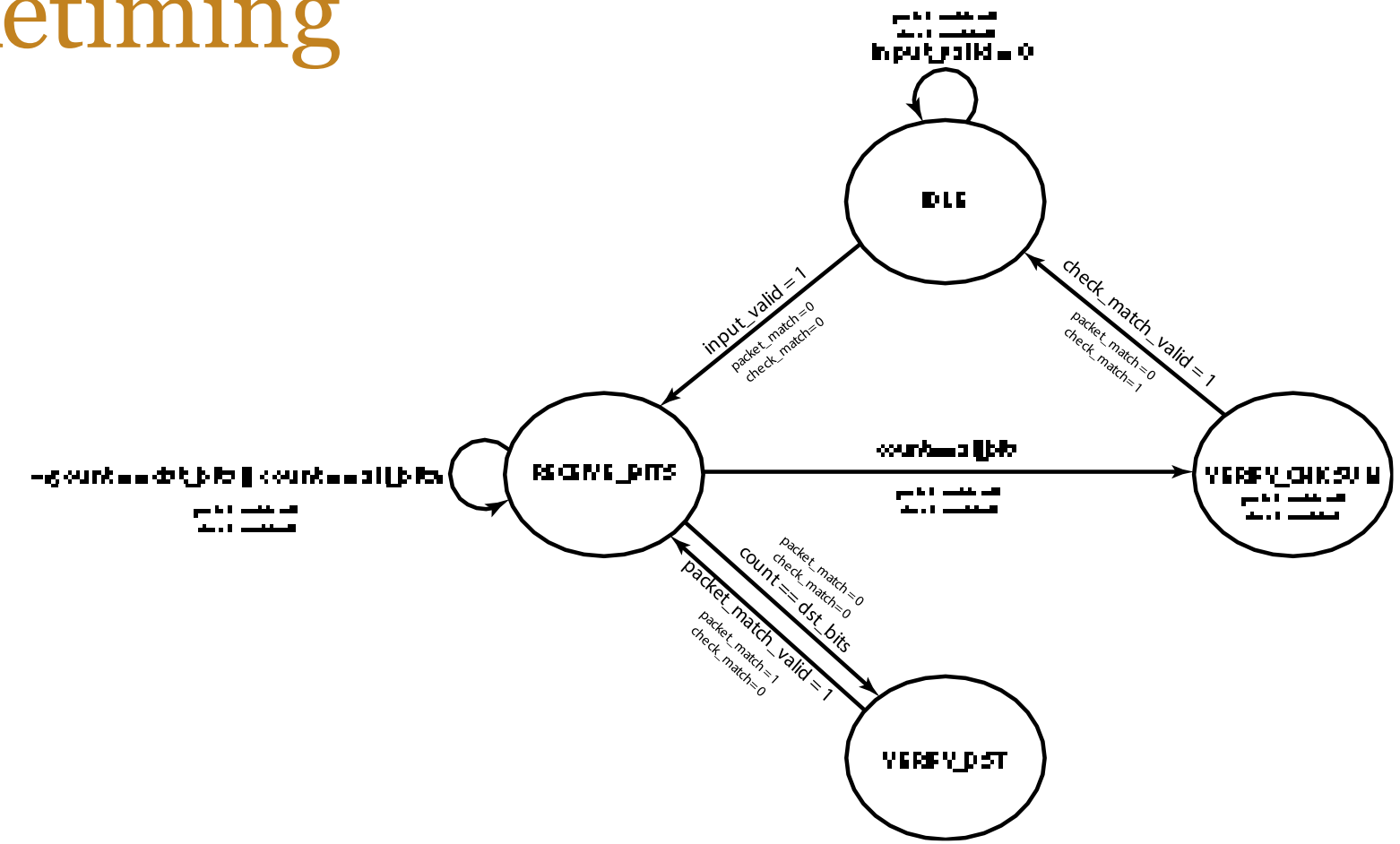
VERIFY_DST State

- Actions:
  - Start streaming in dst field
    - Compare field bit-by-bit with the local addr
  - Keep summing bytes for checksum check
  - Count number of bits until end of dst field
  - Move back to RECEIVE_BITS after done with packet check
- Outputs:
  - packet_match_valid expressed high after done streaming in dst field
  - packet_match = 1 if the match passes

VERIFY_CHKSUM State

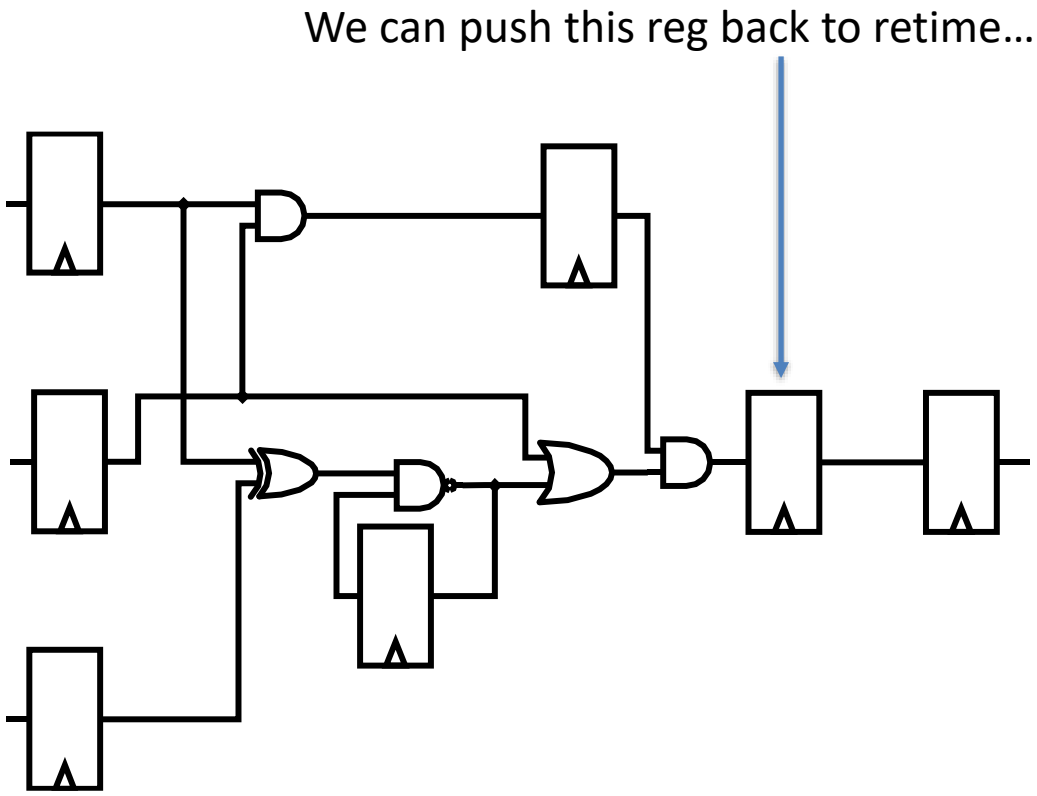
- Actions:
  - Check output of byte accumulator
    - This should have been running ever since we first got to RECEIVE_BITS
- Outputs:
  - check_match_valid can be expressed immediately once we reach this state
  - check_match = 1 if the match passes

Retiming


IDLE
input_valid = 1
packet_match=0
check_match=0
check_match_valid = 1
packet_match=0
check_match=1
count == dst_bits
packet_match=0
check_match=0
packet_match_valid = 1
packet_match=1
check_match=0

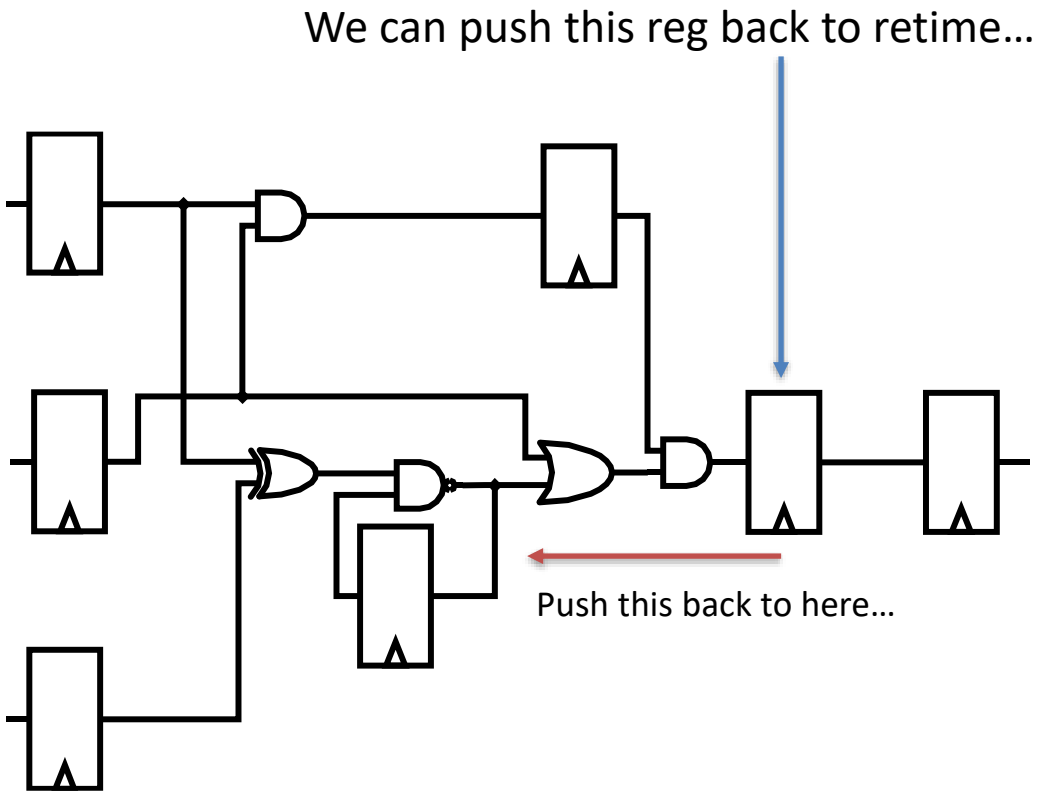
Retiming

- Pushing registers around to change critical path



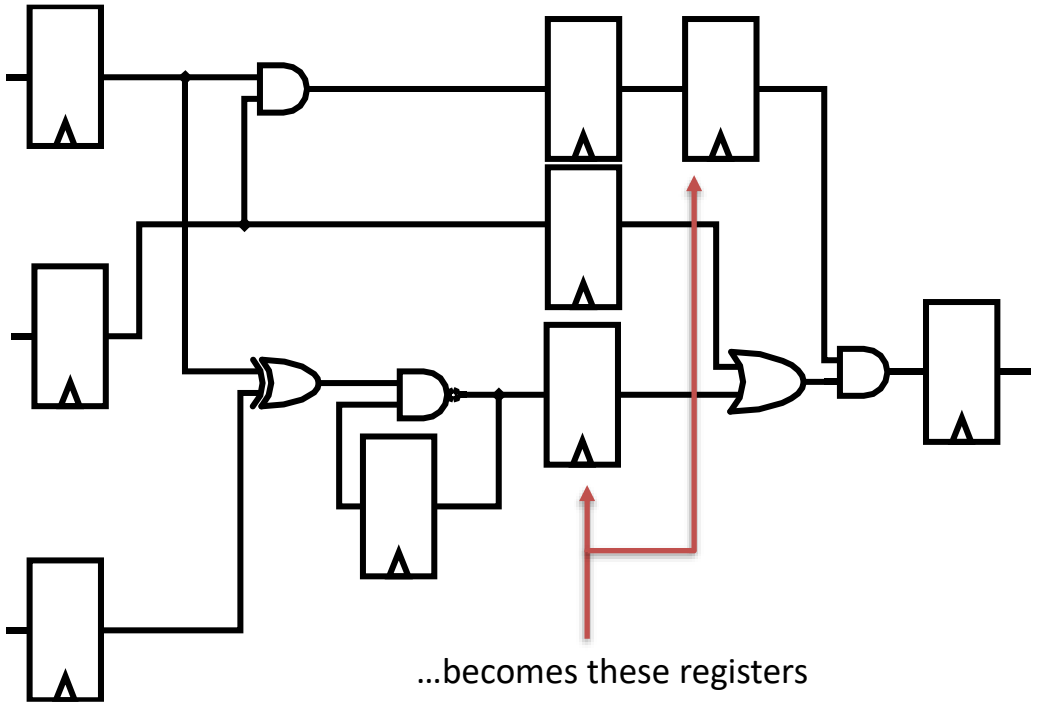
Retiming

- Pushing registers around to change critical path
- Must reg all inputs to gate when pushing reg back



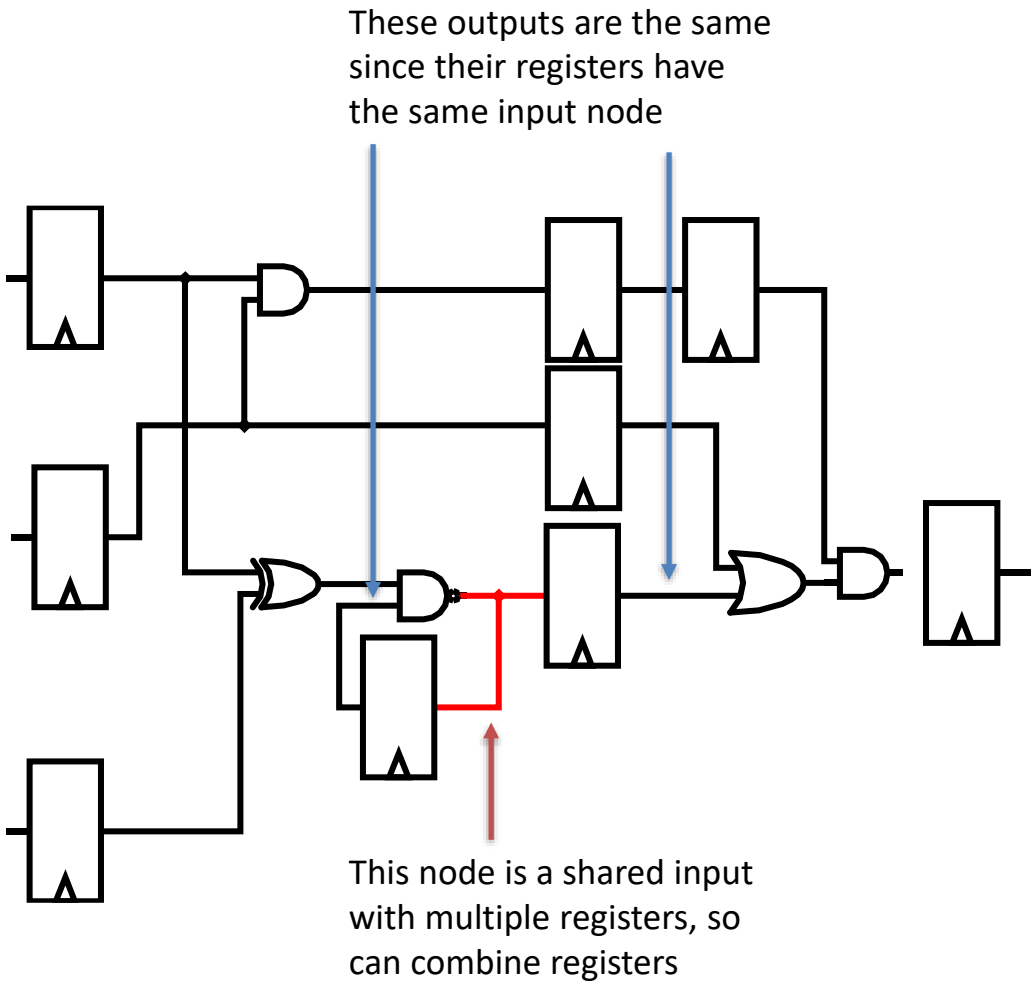
Retiming

- Pushing registers around to change critical path
- Must reg all inputs to gate when pushing reg back



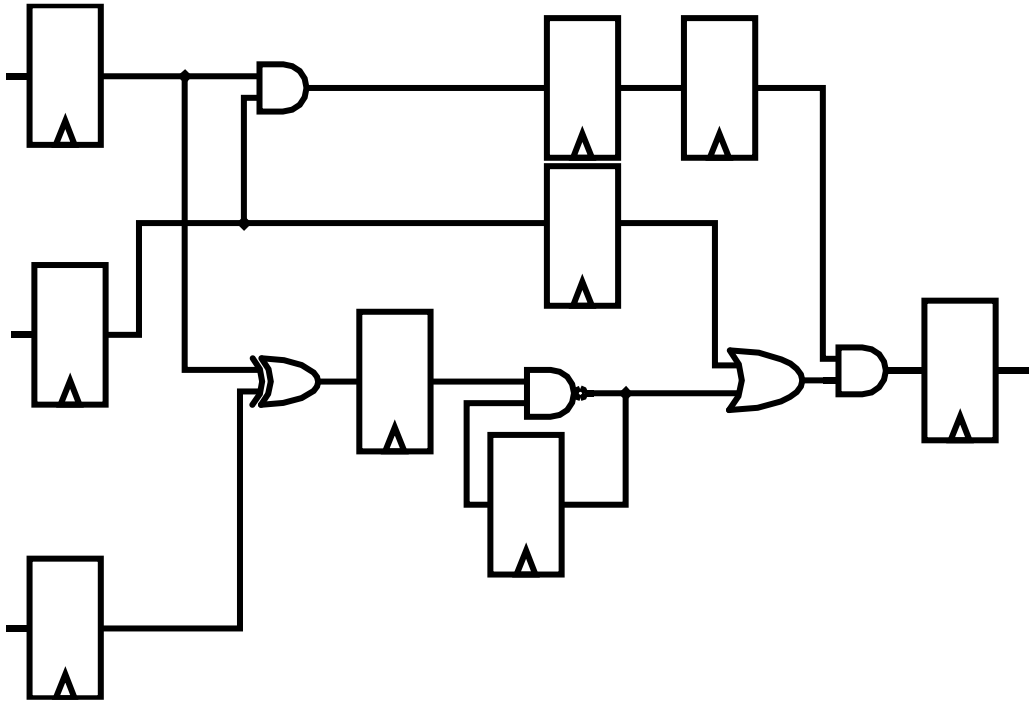
Retiming

- Pushing registers around to change critical path
- Must reg all inputs to gate when pushing reg back
- Combine registers that share inputs
  - Outputs are the same, so can make just one reg



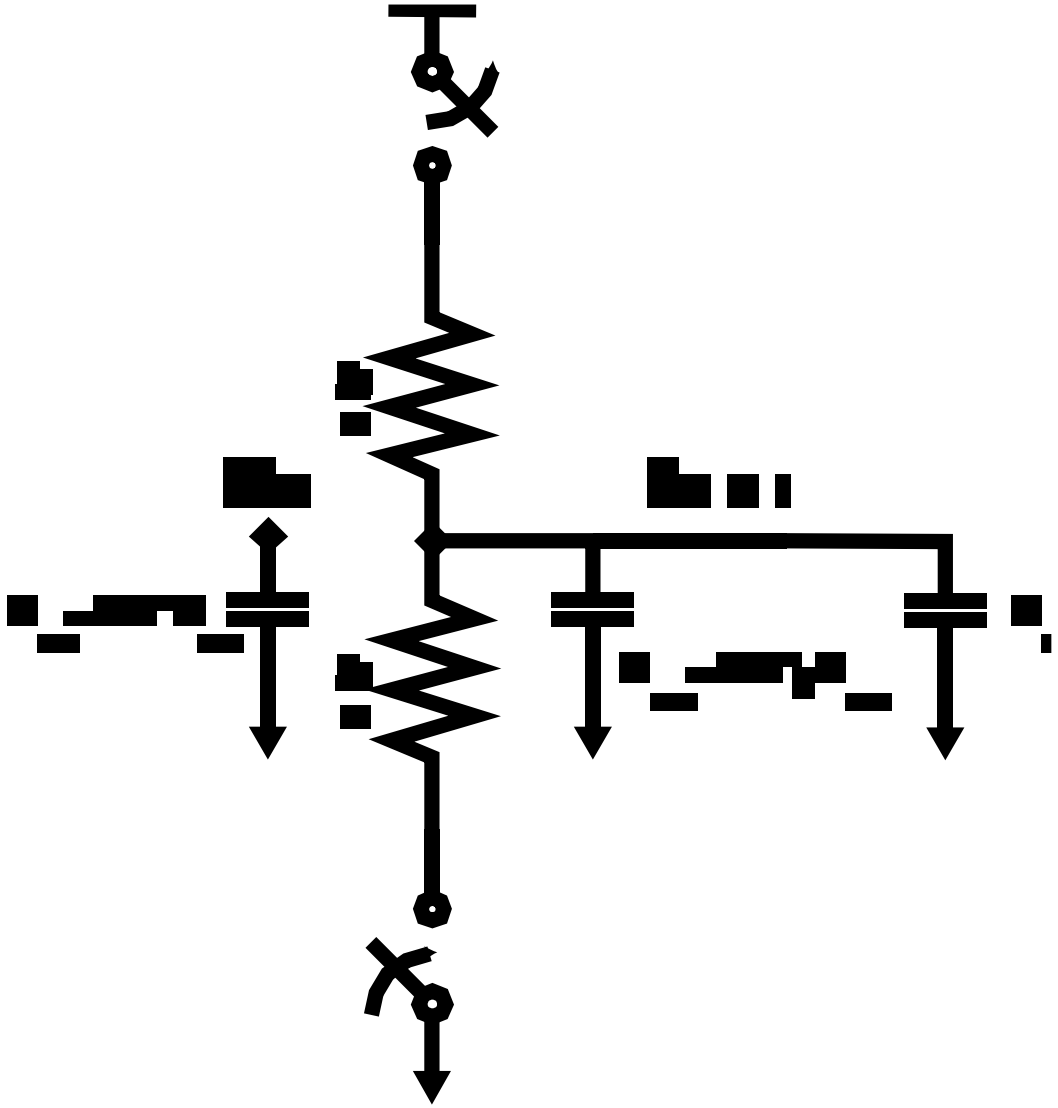
Retiming

- Pushing registers around to change critical path
- Must reg all inputs to gate when pushing reg back
- Combine registers that share inputs
  - Outputs are the same, so can make just one reg



Gate Delay

- Input is 3W because PMOS ~2x more resistive than NMOS in planar
  - May vary depending on technology
- γ is process-dependent parameter
  - relates input and output capacitance



Gate Delay

$$t_p = 0.69\left(\frac{R_N}{W}\right)(C_{int} + C_L)$$

$$= 0.69\left(\frac{R_N}{W}\right)(3W\gamma C_G + C_L)$$

Factoring out $3W\gamma C_G$,

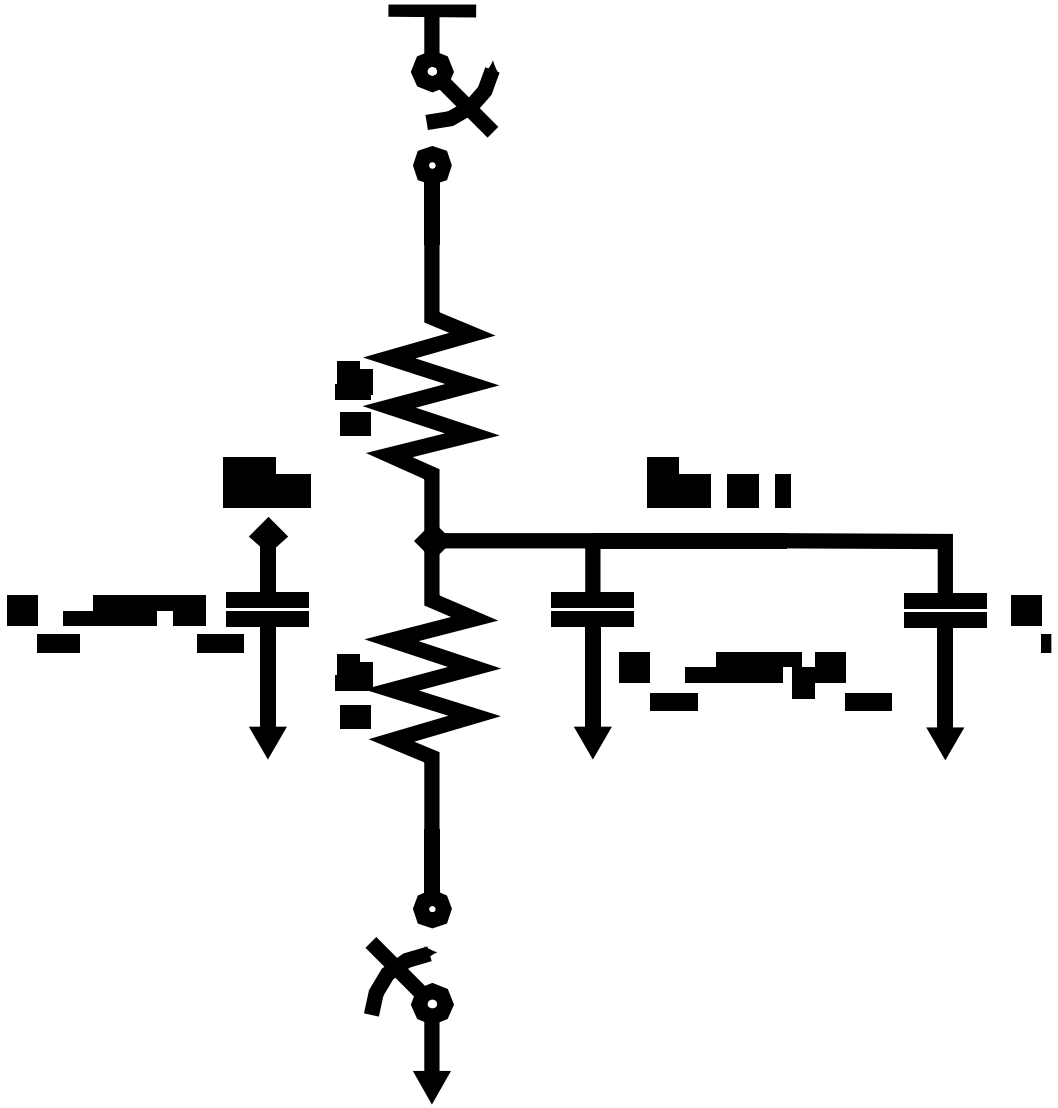
$$\underbrace{0.69(3\gamma R_N C_G)}\left(1 + \frac{C_L}{\gamma C_{in}}\right)$$

$$= t_{p0}\left(1 + \frac{C_L}{\gamma C_{in}}\right)$$

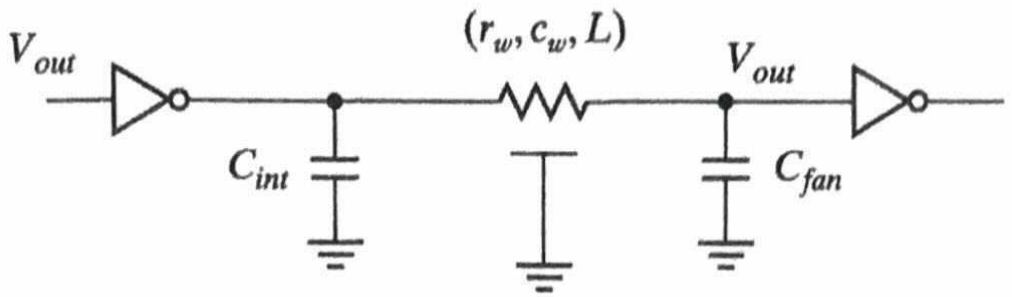
$$= t_{p0}\left(1 + \frac{f}{\gamma}\right)$$

fanout

$$f = \frac{C_L}{C_{in}}$$



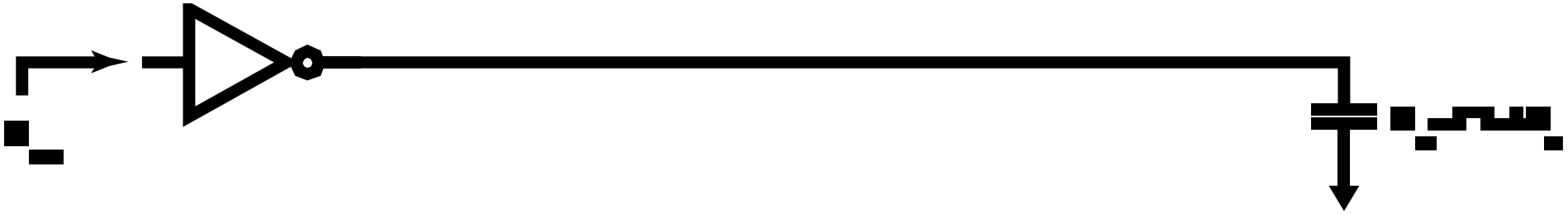
Wire Delay



$$R_w = r_w L$$
$$C_w = c_w L$$

$$t_p = 0.69R_{dr}C_{int} + 0.69R_{dr}C_w + 0.38R_wC_w + 0.69R_{dr}C_{fan} + 0.69R_wC_{fan}$$
$$= 0.69\left(R_{dr}\left(C_{int} + C_w + C_{fan}\right) + R_wC_{fan}\right) + 0.38r_wc_wL^2$$

Buffer Insertion



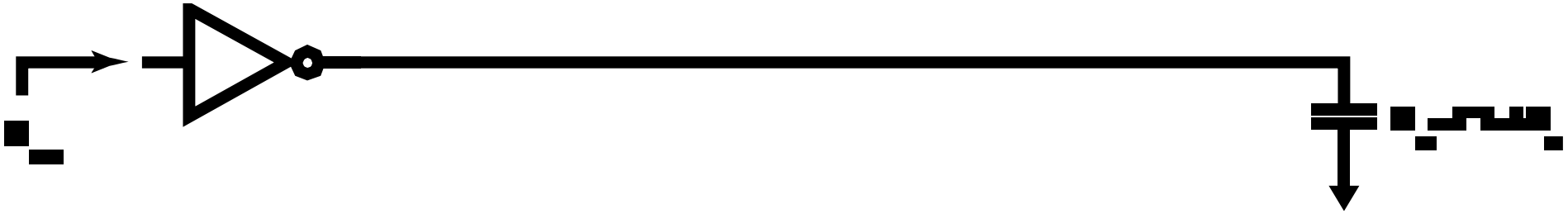
- How can we drive this load?

$$t_p = t_{p0}\left(1 + \frac{256}{\gamma}\right) \approx 257 t_{p0}! \quad \gamma \approx 1$$

- Fanout should be evenly distributed over all inverters

$$f = \sqrt[N]{C_L/C_{in}}$$

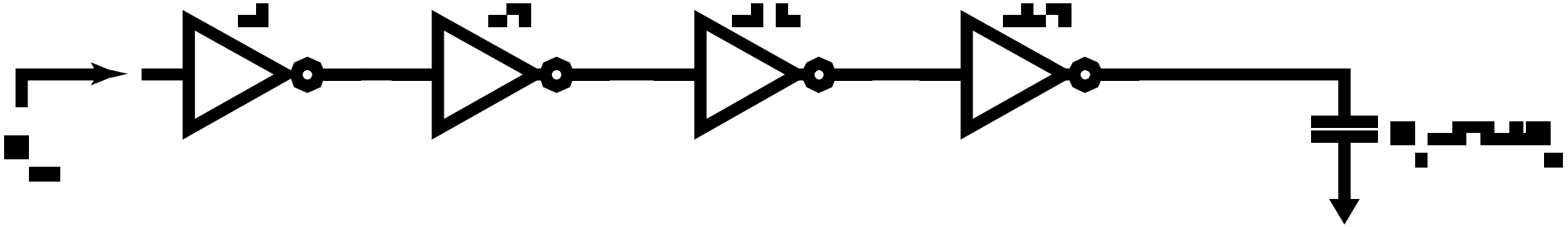
Buffer Insertion



- From lecture slides, $f \approx 4$ for minimal delay when $\gamma = 1$

$$4^N = 256$$
$$N = \log_4 256$$
$$N = 4$$

Buffer Insertion



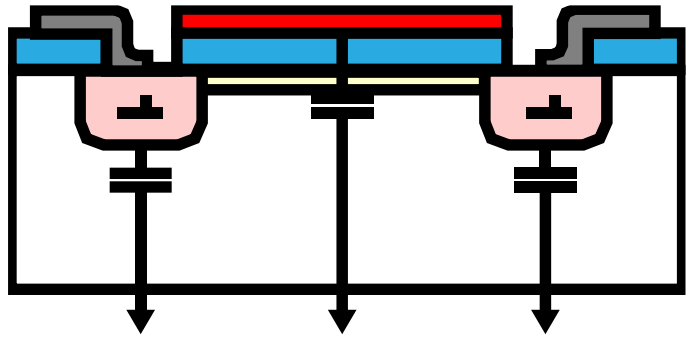
- From lecture slides, $f \approx 4$ for minimal delay when $\gamma = 1$

$$4^N = 256$$
$$N = \log_4 256$$
$$N = 4$$

- Need 4 drivers, each with fanout of 4
  - Each subsequent driver is 4x the previous one

MOSFET Capacitances

- Each terminal of MOSFET as some parasitic capacitance to the substrate
  - Source/Drain capacitance from depletion region at pn boundary
  - Gate capacitance from poly-oxide-channel



MOSFET Capacitances

- Gate capacitance
  - Area x Oxide Capacitance
    - $C_G = C_{ox} \cdot W \cdot L$
- Source/Drain (Diffusion) capacitance
  - Sidewalls and bottom both contribute to capacitance
    - Bottom cap: $C_{bot} = C_j \cdot L_S \cdot W$
    - Sidewall cap: $C_{sw} = C_{jsw} \cdot (2L_s + W)$
- For sidewall cap, do not calculate side facing channel!