Transistor-Level Circuit Analysis

Sol Swords

Talk Outline

- Motivation
- What's a transistor? (optional)
- Bryant's MOSSIM model
- Demo
- Improving the tool and the model

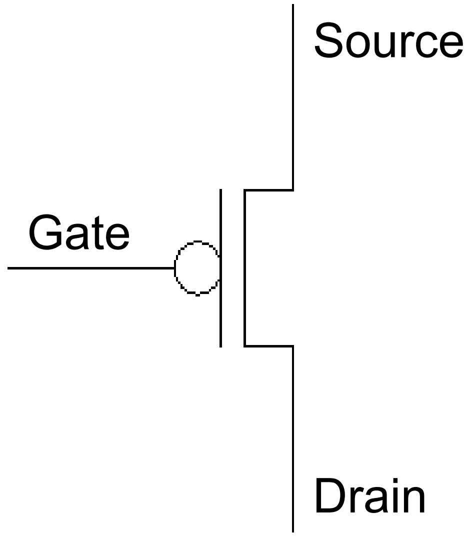
Why transistor-level verification

- rather than gate-level?
  - Much logic is designed at the gate level or higher, using well-tested standard cells
  - But custom logic, arrays, ... are designed at the transistor level, not the gate level
  - Opens possibility of verifiying a third-party artifact without access to higher-level representation
    - Very hard.
    - Not a focus of this talk.

Why transistor-level verification

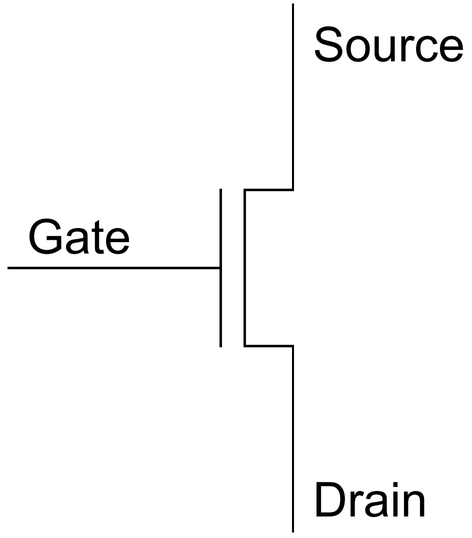
- rather than analog level?
  - Spice-based analog simulators are used to resolve tricky timing issues, characterize gates, check power usage
  - But gives much more information than necessary for functional correctness – just need Boolean outputs, not continuous voltages
  - Transistor-level is faster – can handle much larger designs and longer simulations
  - Symbolic simulation impractical at analog level

What's a transistor?



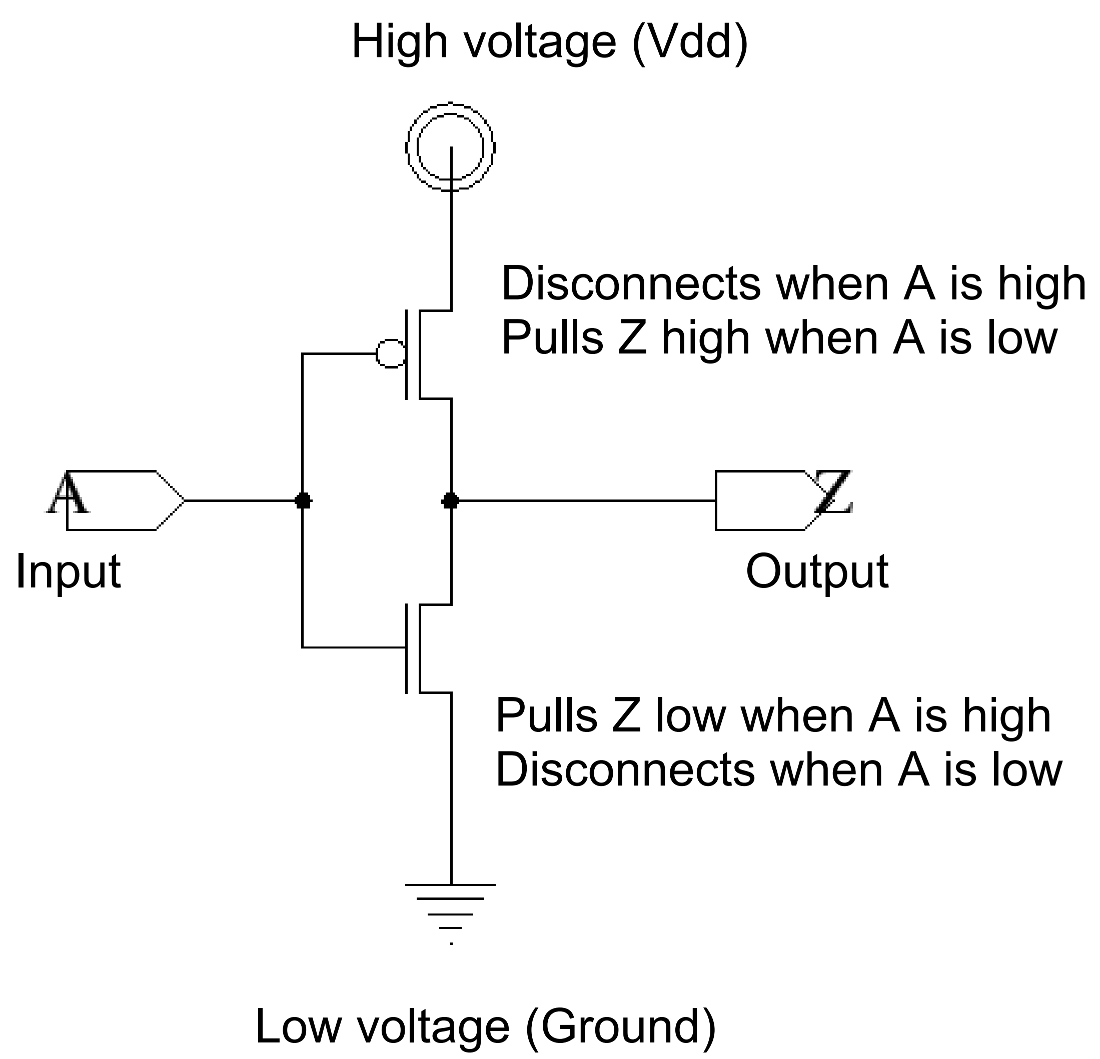
PMOS transistor:
- Gate low: connects source/drain
- Gate high: disconnects source/drain

For our purposes:
Three-terminal electronic component modeled as a voltage-controlled switch.



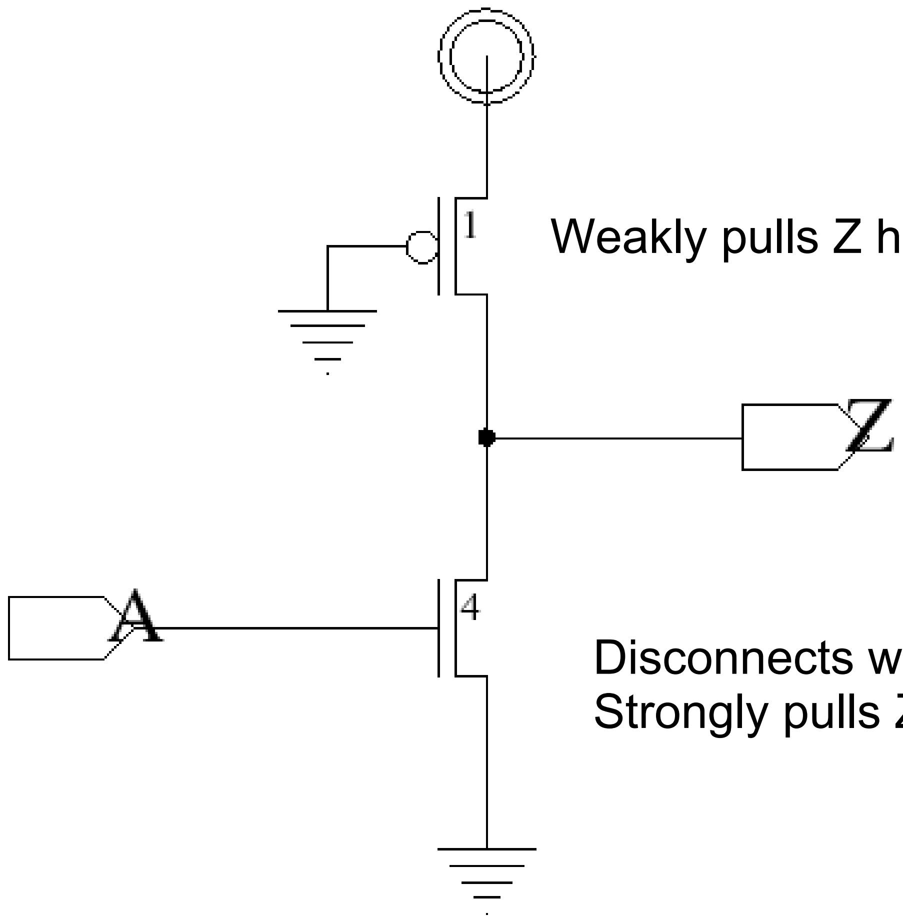
NMOS transistor:
- Gate high: connects source/drain
- Gate low: disconnects source/drain

Simplest Possible Digital Circuit


High voltage (Vdd)
Disconnects when A is high
Pulls Z high when A is low
A
Input
Z
Output
Pulls Z low when A is high
Disconnects when A is low
Low voltage (Ground)

Z = not(A)

Transistor Strengths



Weakly pulls Z high always

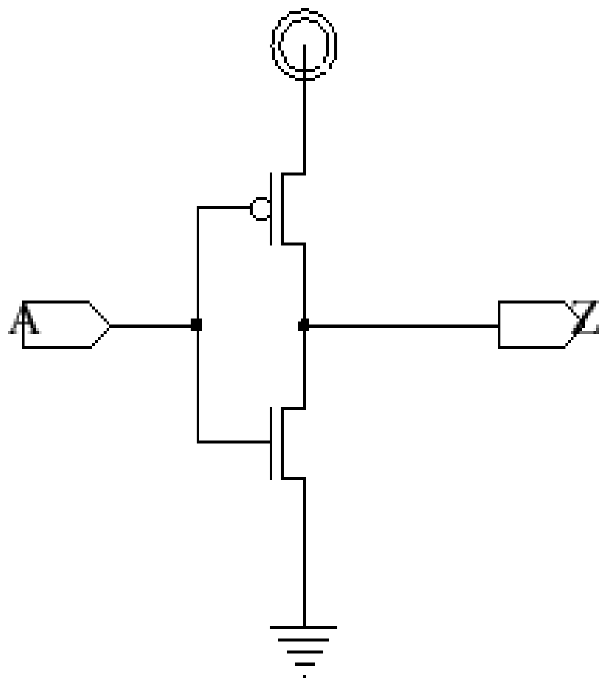
Z = not(A)

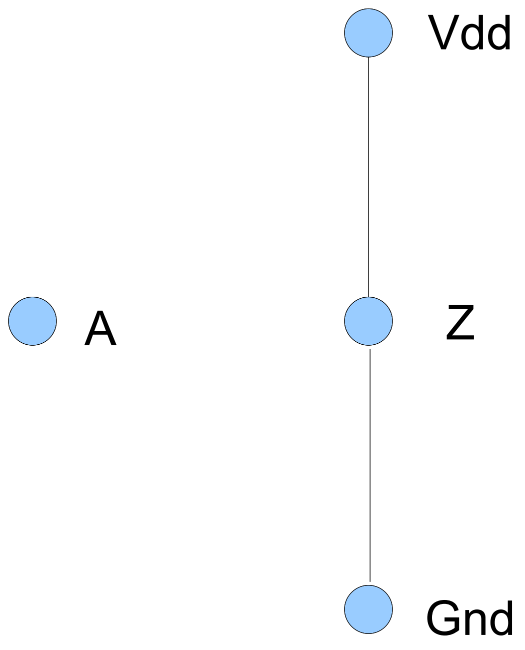
Disconnects when A is low
Strongly pulls Z low when A is high

Pseudo-nMOS style – no longer widely used, but illustrative.

Model: Graph representation

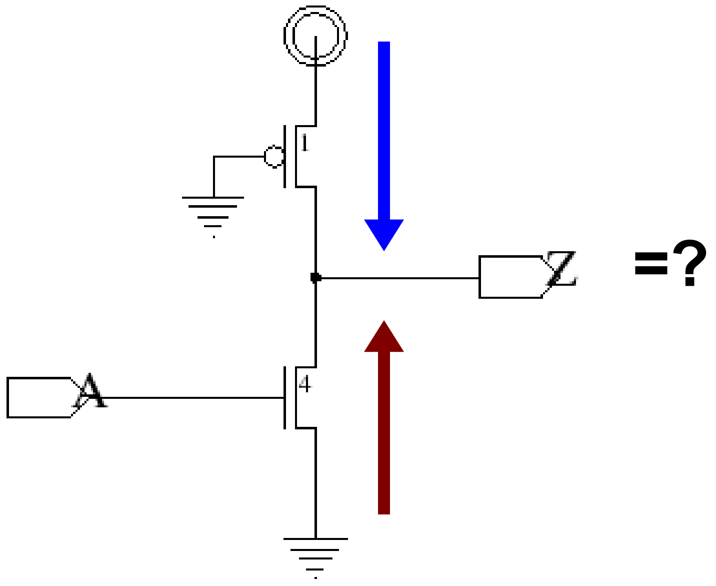
- Circuits are represented as undirected graphs:
  - Each wire and voltage supply is a vertex
  - Each transistor source-drain channel is an edge.
    - Why? Current can flow between source/drain.





Charging Paths

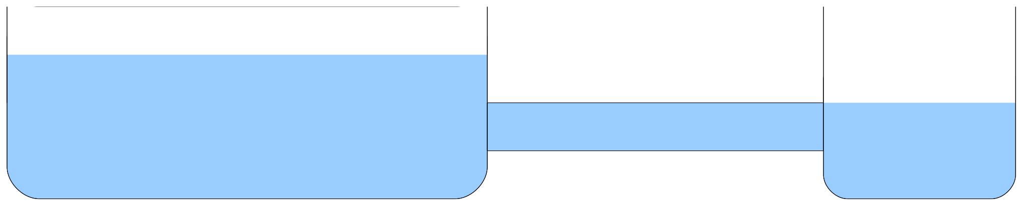
- A wire can be charged or discharged through a path of transistors.
- A path is only active if each transistor in the path is on (gate is high for N, low for P.)
- Multiple active paths may go to a single node – which one sets the node's value?



- May cancel each other and pull to an intermediate voltage
- But don't always – i.e., pseudo-nMOS
- “Comparably strong” paths cancel, or “much stronger” wins

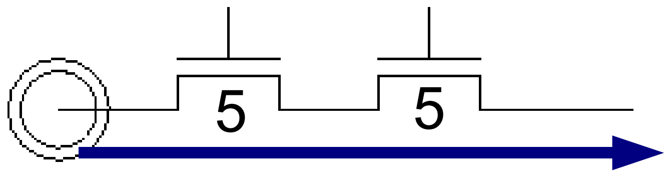
Sharing Charge

- A wire need not be connected to one supply or the other – may float or be connected only to other non-supply wires.
- Wires (nodes) share charge with each other when connected by a path of active transistors.
- Charge flows over connected region until it converges to a uniform voltage.
- Complication: Some wires hold much more charge than others.
- A much larger node can overwhelm small nodes.



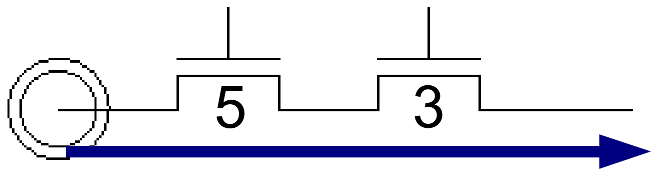
Hierarchy of Paths

From supply, through strong transistors



(rank 5)

From supply, through weaker transistors

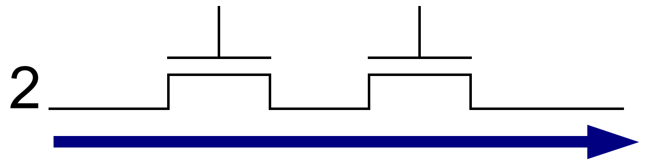


(rank 3)

Paths from supply nodes

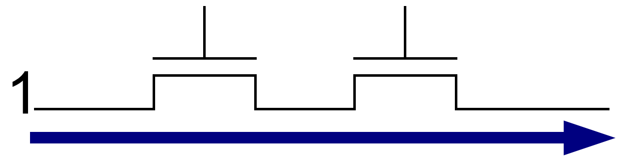
Paths from non-supply nodes

From large wire



(rank 2)

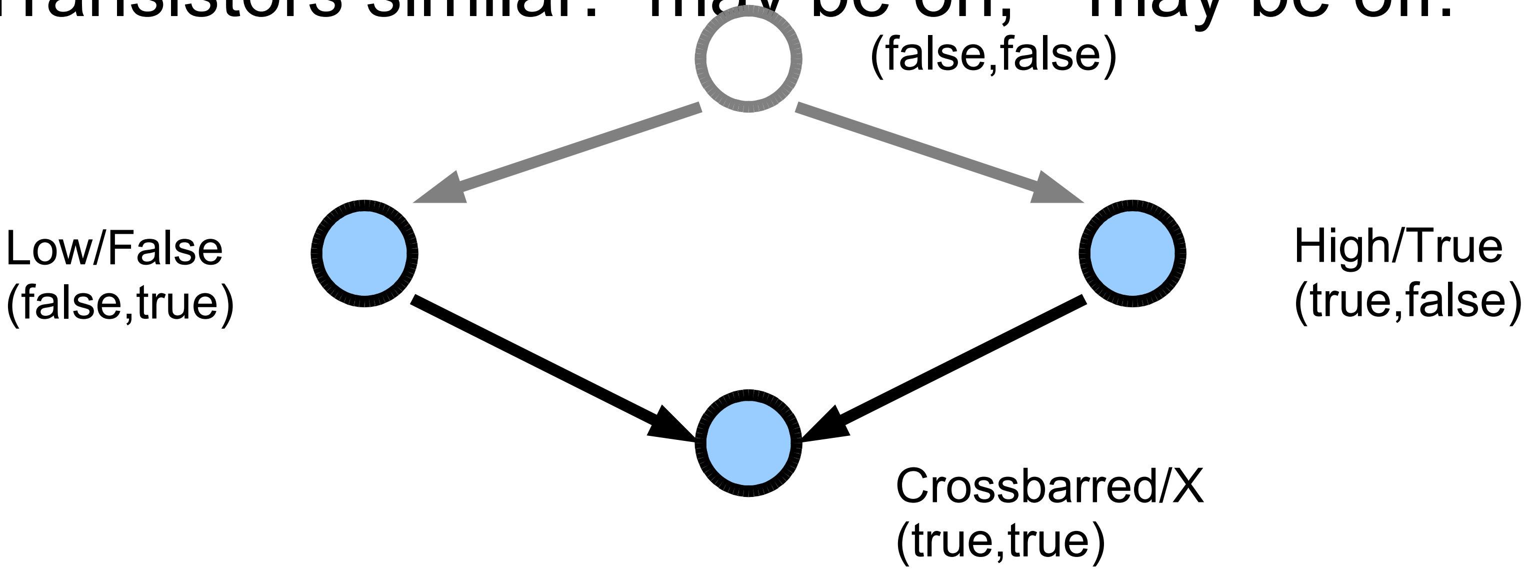
From small wire



(rank 1)

Digression on three-valued logic

- Need more possibilities than just high/low, true/false
- Wires can settle to intermediate voltages when paths of comparable strengths fight.
- Use a lattice model:
  - represent values by onset-offset pairs: “may be high,” “may be low”
- Transistors similar: “may be on,” “may be off.”



Final model

For each non-supply node *n*, find the set of highest-ranked paths to *n* with all transistors on. If all such paths begin at nodes in the same state, then *n* gets that state, otherwise X.

- Every node has a path to it: at minimum, the path *n*->*n* through no transistors.
- New configuration as a function of the initial configuration can be calculated symbolically.
- Is a transistor on or off when its gate is X?
  - On, to be conservative.

Implementation Details

Definitions:

- *Possible path*: one where all transistors may be on
    - “may be on” for a P transistor: gate is low or X
    - Meaning: The path may set the state of its end node.
    - *Possible high path*: possible path beginning at a node with onset true.
    - *Possible low path*: possible path beginning at a node with offset true.
- *Definite path*: one where no transistors may be off
    - (“may be off” for a P transistor: gate is high or X)
    - The path definitely sets the state of its end node.

Implementation Details

- Iterate down through all possible path ranks. For each node *n*, update three Boolean symbolic values, considering only paths of the current rank:
  - n_c (settable) – *n* may be set by a lower-ranking path.
    - Starting value: true
    - New $n_c = n_c \wedge \neg$(exists definite path to n)
  - n_1 (onset) – *n* may be pulled to high voltage.
    - Starting value: false
    - New $n_1 = n_1 \vee (n_c \wedge$ (exists possible high path to n))
  - n_0 (offset) – *n* may be pulled to low voltage.
    - Starting value: false
    - New $n_0 = n_0 \vee (n_c \wedge$ (exists possible low path to n))

Calculating $n_{c,1,0}$

- Number of paths may be exponential in number of vertices
- Polynomial algorithm: Gaussian elimination
- Worst case – $O(v^3)$ BDD operations
- Typical case – linearly many BDD operations
- (Still exponential time if BDDs are bad.)

Symbolic Simulation

- Analysis described so far determines what we do in one symbolic simulation step.
- Current implementation:
  - Unit delay timing model
  - Flatten hierarchical circuits, no shared structure (representation size proportional to circuit size)
  - Start with most nodes set to X, inputs set to a Boolean constant or variable
  - At each step, update nodes affected by last step.
- Demo

Planned Improvements

- Hierarchy
  - Circuits are designed in reusable blocks; they should be analyzed that way as well
- Stored charge is only important on certain nodes
  - Smaller processes have more leakage, so nodes don't stay floating for long
  - (Conservatively) ignoring stored charge allows a large speedup; affects correctness in rare cases
  - Possible to recognize dynamic nodes automatically, or require user input?

Planned Improvements

- Combinational logic elimination
  - Get outputs as a function of inputs once and for all, then ignore intermediate nodes
    - Need intermediate nodes if we care about glitching
  - Requires adjustment to timing
    - Big logic cloud shouldn't count as a single gate delay
- Timing model improvement
  - Unit timing may be too inaccurate
  - Need to allow for wire delays, not just logic
  - Logical effort? Parse output from timing tools? User override?

Theorems about transistors

- Analyzer and simulator are written in ACL2, logic mode.
  - Will I prove them correct?
    - No.
    - Within the realm of possibility to prove that the Gaussian elimination algorithm is equivalent to iteration over all paths (for example), but not high priority.
  - Might be used for theorems about designs, like

```
(equal (eval-bdd (simulator '(...))
                 (list a b c))
       (spec-function a b c))
```

Bibliography

- Randal E. Bryant, "Boolean Analysis of MOS Circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 634-649, July 1987.
- Randal E. Bryant, "Algorithmic Aspects of Symbolic Switch Network Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 618-633, July 1987.

Path strength

- Rank *charging paths* to storage nodes using node and transistor ranks:

  Strength(p) = the minimum of the size of the starting node and the strengths of the transistors in the path.

  ```
  = Min(size[startnode[p]],
          Min({strength[d] | d in transistors[p]})).
  ```

Formalizing the model

- Rank node sizes and transistor strengths:
  - Supply node size (constant)
    > transistor strengths
    > storage node sizes
- All ranks are integers
- Ranks have “order of magnitude” semantics
  - Comparably sized nodes have same rank
  - Transistors of comparable strength have same rank

Supply nodes

- Supply nodes hold effectively infinite charge
- Supply node voltages are unaffected by the modeled circuit
  - Even if two opposite supply nodes are shorted.
- A non-supply node can be connected to two opposite supplies – what happens?
  - Maybe crossbarred (X – bad state)
  - But maybe not – e.g. pseudo-nMOS.
  - Path with stronger transistors wins

Implementation Details

- Iterate down through the path ranks. For each:
  - Record each node's updated value (onset and offset) as a symbolic formula (BDD) in terms of the previous circuit state. Consider only paths of the current rank.
  - Record whether each node may still be updated at a lower rank, also as a symbolic formula. True if there were no paths of the current rank or higher.
- Gaussian elimination algorithm instead of enumeration of all paths
  - $O(v^3)$ (usually fewer) BDD operations instead of exponentially many.