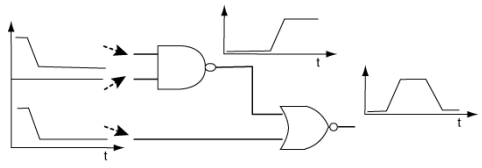
Power optimization

- Glitches cause unnecessary power consumption.
- Logic network design helps control power consumption:
  - minimizing capacitance;
  - eliminating unnecessary glitches.

Glitching example

- Gate network:

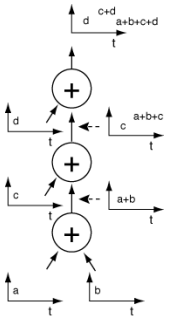


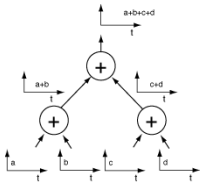
Glitching example behavior

- NOR gate produces 0 output at beginning and end:
  - beginning: bottom input is 1;
  - end: NAND output is 1;
- Difference in delay between application of primary inputs and generation of new NAND output causes glitch.

Adder chain glitching

bad





good

Explanation

- Unbalanced chain has signals arriving at different times at each adder.
- A glitch downstream propagates all the way upstream.
- Balanced tree introduces multiple glitches simultaneously, reducing total glitch activity.

Signal probabilities

The signal probability P_s is the probability that signal *s* is "1".

- Glitching behavior can be characterized by signal probabilities.
  - Delay-independent probabilities
  - Delay-dependent probabilities
- Transition probabilities can be computed from signal probabilities if clock cycles are assumed to be independent.

Delay-independent probabilities

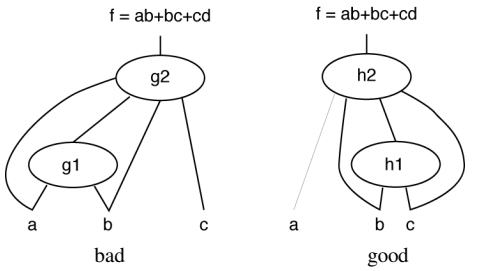
- Compute output probabilities of primitive functions:
  - $P_{NOT} = 1 - P_{in}$
  - $P_{OR} = 1 - \Pi(1 - P_i)$
  - $P_{AND} = \Pi P_i$
- Can compute output probabilities of reconvergent fanout-free networks by traversing tree.

Delay-dependent probabilities

- More accurate estimation of glitching. Glitch accuracy depends on accuracy of delay model.
- Can use simulation-style algorithms to propagate glitches.
- Can use statistical models coupled with delay models.

Factorization for low power

- Proper factorization reduces glitching.



Factorization techniques

- In example, a has high transition probability, b and c low probabilities.
- Reduce number of logic levels through which high-probability signals must travel in order to reduce propagation of glitches.

Power estimation tools

- Power estimator approximates power consumption from:
  - gate network;
  - primary input transition probabilities;
  - capacitive loading.
- May be switch/logic simulation based or use statistical models.

Manufacturing testing

- Errors are introduced during manufacturing.
- Testing verifies that chip corresponds to design.
- Varieties of testing:
  - functional testing;
  - performance testing (binning chips by speed).
- Testing also weeds out infant mortality.

Testing and faults

- Fault model:
  - possible locations of faults;
  - I/O behavior produced by the fault.
- Good news: if we have a fault model, we can test the network for every possible instantiation of that type of fault.
- Bad news: it is difficult to enumerate all types of manufacturing faults.

Fault model

- Stuck-at-0/1
- Stuck-open
- Delay fault

Stuck-at-0/1 faults

- Stuck-at-0/1: logic gate output is always stuck at 0 or 1, independent of input values.
- Correspondence to manufacturing defects depends on logic family.
- Experiments show that 100% stuck-at-0/1 fault coverage corresponds to high overall fault coverage.

Testing procedure

- Testing procedure:
  - set gate inputs;
  - observe gate output;
  - compare fault-free and observed gate output.
- Test vector: set of gate inputs applied to a system.

Assignment

- Give at least one test for Stuck-at-0 and Stuck-at-1 faults for the following static gates
  - *(a+b+c)'*
  - *[(a+b)c]'*

Stuck-at faults in gates

a	b	OK	SA0	SA1
0	0	1	0	1
0	1	1	0	1
1	0	1	0	1
1	1	0	0	1

NAND

a	b	OK	SA0	SA1
0	0	1	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	0	1

NOR

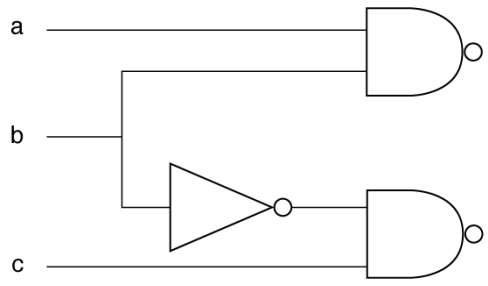
Testing single gates

- Three ways to test NAND for stuck-at-0, only one way to test it for stuck-at-1.
- Three ways to test NOR for stuck-at-1, only one way to test it for stuck-at-0.

Testing combinational networks

- 100% coverage: test every gate for
  - stuck-at-0;
  - stuck-at-1.
- Assume that there is only one faulty gate per network.
- Most networks require more than one test vector to test all gates.

Multiple test example

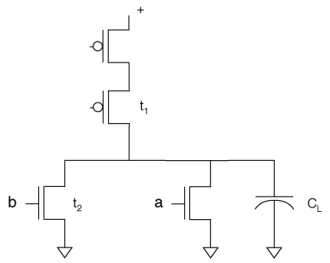


Example

- Can test both NANDs for stuck-at-0 simultaneously (abc = 000).
- Cannot test both NANDs for stuck-at-1 simultaneously due to inverter. Must use two vectors.
- Must also test inverter.

Stuck-at-open/closed model

- Models transistors always on/off.



Stuck-open behavior

- If t_1 is stuck open (switch cannot be closed), there can be no path from V_{DD} to output capacitance.
- Testing requires two cycles:
  - must discharge capacitor;
  - try to operate t_1 to see if capacitor can be charged.

Delay fault

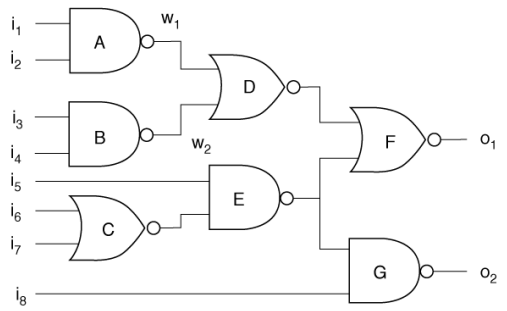
- Delay falls outside acceptable limits:
  - gate delay fault assumes that all delays are lumped into one gate;
  - path delay fault models delay problems along path through network.
- Delay problems reduce yield:
  - performance problems;
  - functional problems in some types of circuits.

Combinational network testing

Two parts to testing:

- controlling the inputs of (possibly interior) gates;
- observing the outputs of (possibly interior) gates.

Combinational testing example



Testing procedure

- Goal: test gate D for stuck-at-0 fault.
- First step: justify 0 values on gate inputs.
- Work backward from gate to primary inputs:
  - w1 = 0 (A output = 0);
  - i1 = i2 = 1.

Testing procedure, cont'd

- Observe the fault at a primary output:
  - o1 gives different values if D is true/faulty.
- Work forward and backward:
  - F's other input must be 0 to detect true/fault.
  - Justify 0 at E's output.
- In general, may have to propagate fault through multiple levels of logic to primary outputs.