Topics Chapter 4
Combinational Logic Networks

- Standard cells (already discussed in Chapter 2).
- Managing the delay of combinational networks ("logical effort", "buffer insertion" and "crosstalk minimization" skipped).
- Power optimization.
- Testing of combinational logic (postponed).

Fanout

- Fanout adds capacitance.

Ways to drive large fanout

- Increase sizes of driver transistors. Must take into account rules for driving large loads.
- Add intermediate buffers. This may require/allow restructuring of the logic.

Buffers
### Wire capacitance

- Use layers with lower capacitance.
- Redesign layout to reduce length of wires with excessive delay.

### Placement and wire capacitance

Unbalanced load

More balanced

### Path delay

- Combinational network delay is measured over paths through network.
- Can trace a causality chain from inputs to worst-case output.

### Path delay example

Network

Graph model: nodes and edges modeling network; however, only the delays of nodes matter, not their function.
Delay model

- Nodes represent gates.
- Assign delays to edges—signal may have different delay to different sinks.
- Lump gate and wire delay into a single value.

Critical path

- Critical path = path which creates longest delay.
- Can trace transitions which cause delays that are elements of the critical delay path.

Critical path through delay graph

Reducing critical path length

- To reduce circuit delay, must speed up the critical path—reducing delay off the path doesn’t help.
- There may be more than one path of the same delay. Must speed up all equivalent paths to speed up circuit.
- Must speed up cutset through critical path.
Logic rewrites

- Deep logic
- Shallow logic

Logic transformations

- Can rewrite by using subexpressions.
- Flattening logic increases gate fanin.
- Logic rewrites may affect gate placement.

False paths

- Logic gates are not simple nodes—some input changes don’t cause output changes.
- A false path is a path which cannot be exercised due to Boolean gate conditions.
- False paths cause pessimistic delay estimates.

Logic optimization

- Logic synthesis programs transform Boolean expressions into logic gate networks in a particular library.
- Optimization goals: minimize area, meet delay constraint.
Technology-independent optimizations

- Works on Boolean expression equivalent.
- Estimates size based on number of literals.
- Uses factorization, resubstitution, minimization, etc. to optimize logic.
- Technology-independent phase uses simple delay models.

Technology-dependent optimizations

- Maps Boolean expressions into a particular cell library.
- Mapping may take into account area, delay.
- May perform some optimizations on addition to simple mapping.
- Allows more accurate delay models.

Power consumption (Chapter 3)

- A single cycle requires one charge and one discharge of capacitor: $E = C_L(V_{DD} - V_{SS})^2$.
- Clock frequency $f = 1/t$.
- Power = $E f = f C_L(V_{DD} - V_{SS})^2$.

Power optimization

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

Gate network:

- 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

- 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

- Glitching behavior can be characterized by signal probabilities.
- Transition probabilities can be computed from signal probabilities if clock cycles are assumed to be independent.
- Some primary inputs may have non-standard signal probabilities—control signal may be activated only occasionally.

Transition Probability

- $P_s$: probability that signal $s$ is 1.
- Transition probability of this signal:
  \[ P_{tr,s} = 2P_s(1 - P_s). \]

Delay-independent probabilities

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

Delay-dependent probabilities

- Can use simulation-style algorithms to propagate glitches.
- Can use statistical models coupled with delay models.
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.

Factorization for low power

- Proper factorization reduces glitching; $a$ has the highest transition probability.

\[
f = ab + bc + cd
\]

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.

Layout for low power

- Place and route to minimize capacitance of nodes with high glitching activity.
- Feed back wiring capacitance values to power analysis for better estimates.