Topics

- CAD systems.
- Simulation.
- Placement and routing.
- Layout analysis.

CAD systems

- Tools aren’t very useful if they don’t talk to each other.
- Design interchange languages:
  - VHDL (TM), Verilog (TM) (function and structure);
  - EDIF (netlists);
  - GDS, CIF (masks).

CAD tool interactions

Back annotation

- Often want to iteratively improve design.
- Back annotation updates a more-abstract design with information from later design stages.
  - Example: annotate logic schematic with extracted parasitic Rs and Cs.
- Back annotation requires tools to know more about each other.
Event-driven simulation

- Event-driven simulation is designed for digital circuit characteristics:
  - small number of signal values;
  - relatively sparse activity over time.
- Event-driven simulators try to update only those signals which change in order to reduce CPU time requirements.

Event-driven simulator structure

- An event is a change in a signal value.
- A *timewheel* is a queue of events (data structure is circular, hence the term *wheel*).
- Simulator traces structure of circuit to determine causality of events—event at input of one gate may cause new event at gate’s output.

Event-driven simulation example

- Events at primary inputs:
  - A changes at t=1;
  - B changes at t=2.
- Immediate causality:
  - C changes at t=3 when both inputs to NOR are 0.
- Event propagation:
  - D changes at t=4.
Delay models

- Unit-delay simulators assume that each component has a one-unit delay. Model function but not performance.
- Variable-delay simulators allow each component to have its own delay. Accuracy of performance estimates from variable-delay simulators depends on how well circuits can be extracted to digital model.

Switch simulation

- Special type of event-driven simulation optimized for MOS transistors.
- Treats transistor as switch. Takes capacitance into account to model charge sharing, etc.
- Can also be enhanced to model transistor as resistive switch.

Switch simulation example

- Node g may be connected to either power supply, but signals on that node are terminated by gate of transistor.
- To solve for values of a and b nodes, must first solve for value of g node.
  - If g=1, then a=b.
  - If g=0, other parts of circuit determine a and b independently.
Switch simulation and charge sharing

- Closed transistor connects source and drain nodes. Want to determine voltages of source/drain nodes taking into account capacitance.
- Capacitance determines node size. Use size of connected nodes to determine new value of nodes.
- Result may be X (unknown).

Layout synthesis

- Two critical phases of layout design:
  - placement of components on the chip;
  - routing of wires between components.
- Placement and routing interact, but separating layout design into phases helps us understand the problem and find good solutions.

Placement metrics

- Quality metrics for layout:
  - area;
  - delay.
- Area and delay determined in part by wiring.
- How do we judge a placement without wiring? Estimate wire length without actually performing routing.

Wire length as a quality metric

- bad placement
- good placement
Wire length measures

- Estimate wire length by distance between components.
- Possible distance measures:
  - Euclidean distance ($\sqrt{x^2 + y^2}$);
  - Manhattan distance ($x + y$).
- Multi-point nets must be broken up into trees for good estimates.

Placement techniques

- Can construct an initial solution, improve an existing solution.
- Pairwise interchange is a simple improvement metric:
  - Interchange a pair, keep the swap if it helps wire length.
  - Heuristic determines which two components to swap.

Placement by partitioning

- Works well for components of fairly uniform size.
- Partition netlist to minimize total wire length using min-cut criterion.
- Partitioning may be interpreted as 1-D or 2-D layout.

Min-cut bisecting partitioning

- Works well for components of fairly uniform size.
- Partition netlist to minimize total wire length using min-cut criterion.
- Partitioning may be interpreted as 1-D or 2-D layout.
Min-cut bisecting partitioning, cont’d

- Swapping A and B:
  - B drags 1 net;
  - A drags 3 nets;
  - total cut increase: 4 nets.

- Conclusion: probably not a good swap, but must be compared with other pairs.

Simulated annealing

- Powerful but CPU-intensive optimization technique.
- Analogy to annealing of metals:
  - temperature determines probability of a component jumping position;
  - probabilistically accept moves.
  - start at high temperature, cool to lower temperature to try to reach good placement.

Routing

- Major phases in routing:
  - global routing assigns nets to routing areas;
  - detailed routing designs the routing areas.

- Net ordering is a major problem. Order in which nets are routed determines quality of result. Net ordering is a heuristic.

Maze (or area) routing

- Will find shortest path for a single wire, if such a path exists.
- Two phases:
  - Label nodes with distance, radiating from source.
  - Use distances to trace from sink to source, choosing a path that always decreases distance to source.
Maze routing example

Mask-Level Layout Tools

- **Layout compaction**: push mask polygons as close to each other respecting design rules.
- **Design-rule checking**: verify that mask patterns obey the design rules.
- **Layout extraction**: reconstruct transistor circuit from the layout including parasitics.
- **Layout versus schematic** (LVS): compare extracted transistor netlist with original one.

Topics

- Timing analysis.
- Logic synthesis.
- Sequential machine optimizations.
- Hardware/software co-design.

Timing analysis

- Unlike simulation, timing analysis is value-independent—doesn’t require specifying inputs.
- Simulation can be optimistic—you may not apply worst-case input vector.
- Timing analysis can be pessimistic, but that is safer than optimistic.
Signal delay example

Must apply worst case to find longest delay:

Timing analysis procedure

- Two major steps:
  - build graph with elemental delays;
  - traverse graph to find longest path.
- Must model 0-1 and 1-0 delays independently for more accurate total delay.
- Use value analysis to prune impossible paths.

Switch circuit example

- Make assumptions about primary inputs.
- **Primitive path delay**: RC delay from power supply or primary input to transistor gate or primary output.
- Primitive path (p0, p1, p2, p3) delays computed from RC analysis.
- Each path forms an edge in timing analysis graph.
Switch circuit example, cont’d

- Timing analysis graph is analyzed to find worst-case delay through entire circuit.
- Timing graph structure:
  - nodes are sources and sinks of primitive delay paths;
  - edges represent primitive delay paths.

Timing analysis pessimism

- False paths create unexcercisable paths which make delay pessimistic. Can be identified using analysis algorithms.
- Some transistor configurations only allow current flow in one direction—other direction of current/signal flow is a false path.

Current/signal flow analysis

False current path example

shift₁ and shift₂ are never simultaneously active.
False current path example, cont’d

- Many paths in barrel shifter cannot be exercised.
- Driver gates enforce current flow in one direction on data lines, eliminating some paths through the pass transistors.
- Path analysis which does not take into account feasible current flow will identify infeasible long paths.

Transistor sizing

- Once transistor-level critical path has been identified, transistors can be sized to optimize delay.
- Transistor sizing is cast as optimization problem to meet performance goal while minimizing total active area.

Logic synthesis

- Goal: create a logic gate network which performs a given set of functions.
- Input is Boolean formulas; gates also implement Boolean functions.
- Logic synthesis:
  - maps onto available gates;
  - restructures for delay, area, testability, power, etc.

Logic synthesis phases

- Technology-independent optimizations work on logic representations that do not directly model logic gates.
- Technology-dependent optimizations work in the available set of logic gates.
- Transformation from technology-independent to technology-dependent is called library binding or technology mapping.
Boolean network

- A Boolean network is the main representation of the logic functions for technology independent optimizations.
- Each node can be represented as sum-of-products (or product-of-sums).
- Provides multi-level structure, but functions in the network need not correspond to logic gates.

Boolean network example

Terms

- **Support**: set of variables used by a function.
- **Transitive fanout**: all the primary outputs and intermediate variables of a function.
- **Transitive fanin**: all the primary inputs and intermediate variables used by a function. Transitive fanin determines a *cone* of logic.

Technology-independent logic optimization

- **Simplification** rewrites node to simplify its form.
- **Network restructuring** introduces new nodes for common factors, collapses several nodes into one new node.
- **Delay restructuring** changes factorization to reduce path length.
Cost in the Boolean network

- Don’t know exact gate structure, but can estimate final network cost:
  - area estimated by number of literals (true or complement forms of variables);
  - delay estimated by path length.

Simplification

- Rewrites a node to reduce the number of literals in the node.
- Function defined by:
  - on-set: set of inputs for which output is 1;
  - off-set: set of inputs for which output is 0;
  - don’t-care-set: set of inputs for which output is don’t-care.

Functions, covers, and cubes

- Each way to write a function as a sum-of-products is a cover since it covers the on-set.
- A cover is composed of cubes—product terms which define a subspace cube in the function space.

Cover example
Covers and optimizations

- Larger cover:
  - \( x_1' x_2' x_3' + x_1 x_2' x_3' + x_1' x_2 x_3' + x_1 x_2 x_3 \)
  - requires four cubes, 12 literals.

- Smaller cover:
  - \( x_2' x_3' + x_1' x_3' + x_1 x_2 x_3 \)
  - requires three cubes, seven literals;
  - \( x_1' x_2 x_3' \) is covered by two cubes.

Partially-specified functions

- Don’t-cares can be implemented in either the on-set or off-set.
- Don’t-cares provide the greatest opportunities for minimization in many cases.

Partially-specified function example: problem

- \( = \) element of ON-SET
- \( = \) element of DC-SET

Solution requires: two cubes, three literals.

Partially-specified function example: cover

- \( = \) element of ON-SET
- \( = \) element of DC-SET

Solution requires: two cubes, three literals.
Espresso

- Well-known two-level logic optimizer.
- Espresso optimization loop (see p. 479):
  - expand;
  - make irredundant;
  - reduce.
- Optimization loop is designed to refine cover to reduce its size.

Partial collapsing

Before and after collapsing of a network.

Factorization

- Based on division:
  - formulate candidate divisor;
  - test how it divides into the function;
  - if $g = f/c$, we can use $c$ as an intermediate function for $f$.
- Algebraic division: don’t take into account Boolean simplification (e.g. $a \cdot 1 = 1$). Less expensive than Boolean division.

Factorization using division

- Three steps:
  - generate potential common factors and compute literal savings if used;
  - choose factors to substitute into network;
  - restructure the network to use the new factors.
- Algebraic/Boolean division can be used to implement first step.
Factorization for delay

- Remove factors from critical path, add them off critical path:

Library binding

- Also known as technology mapping.
- Rewrites Boolean network in terms of available logic functions.
- Can optimize for both area and delay.
- Can be viewed as a pattern matching problem. Tries to find pattern match which minimizes area/delay cost.

Technology mapping procedure

- Write Boolean network in canonical NAND form.
- Write each library gate in canonical NAND form. Assign cost to each library gate.
- If network is a tree, can use dynamic programming to select minimum-cost cover of network by library gates.

Breaking into trees

- Not optimal, but reasonable cuts usually work OK.
Technology mapping example

after three levels of matching

Technology mapping example, cont’d

after four levels of matching

Technology mapping example, cont’d

- Try to cover tree from primary inputs to primary outputs.
- Proceed one gate at a time. At the next level, select minimum-cost cover at that point.
- Latest selection may require removing some earlier matches.

Sequential machine optimizations

- State assignment: choose codes for states.
- Create common factors in states by assigning them close codes:
  - $s_0 = 110 \ (x_0 x_1 x_2')$; $s_1 = 111 \ (x_0 x_1 x_2)$;
  - $s_0 + s_1 = x_0 x_1$.
- Program KISS: minimizes symbolic state machine to find states which should be coded close together.
Coding procedure

- Symbolic minimization creates constraints on coding—sets of states which should be coded closely together.
- If all constraints cannot be satisfied in minimum number of bits, can add bits to code to allow constraints to be satisfied.

Adding code bits

- Code groups: s1, s2
  s2, s3
  s3, s4, s2

3-bit encoding necessary if states in all code groups should have distance 1.

Hardware/software co-design

- Use programmable CPUs along with specialized logic blocks to implement a particular application.
- CPUs can implement background functions much more efficiently than dedicated logic: higher utilization of logic.
- Important in systems-on-silicon.

Co-synthesis styles

- Hardware/software partitioning:
  - Architectural template consists of 1 CPU + n ASICs.
  - Put operations on CPU or ASIC.
- Distributed system synthesis:
  - No fixed architectural template.
  - Allocate function units and communication, schedule operations.