SYNTHESIS CAPABILITIES OF LOSP

William Bricken
November 1995

Standard reference books on the design, synthesis, optimization, and testing of digital circuits [see references] identify a small set of circuit configuration problems and research issues. This paper identifies the most prevalent of those problems, and maps them onto the existing and proposed capabilities of the Losp engine. I identify our innovations and strengths, and areas for which we have no substantive contribution.

Two-level Combinatorial Logic Optimization

Two-level optimization is a classic problem and is very well understood. We assume multiple output functions.

Uses: PLA arrays, intermediate step in multi-level optimization.

Goals: minimization of the number of terms and the number of literals.

Problem: exact minimization by finding the minimum Boolean cover (prime implicants). Heuristic minimization by applying a variety of transformations to the two-level representation.

Standard techniques: variants of Quine-McCluskey, using a canonical form of the logic function such as the sum-of-products.

Programs: ESPRESSO-EXACT, ESPRESSO

Weakness: Exact methods are exponential, requiring additional techniques to constrain the space of implicants. Inappropriate for more than about ten input variables, thus works only for semi-trivial circuits.

Recent developments: signature cubes, clever notations for storing and manipulating cubes.

Losp: Not particularly useful for two-level forms. Losp transformations map onto most heuristic tools with no particular improvements. Losp rules tend to convert problems to multilevel form.

Comments: two-level circuits are no longer highly motivated. Tautology algorithms identify covers directly. Losp tautology identification is extremely efficient, but probably equivalent to those which use smart search and storage notations.

Extensions: opcode minimization permits arbitrary input encoding. Losp and traditional approaches can use algebraic techniques which are independent of

input encodings. When output encodings are free, the problem is complex, requiring multi-level techniques.

Conclusions: We should replicate two-level tools in Losp for completeness, but expect no performance improvements.

Multi-level Combinatorial Logic Optimization

Multi-level optimization is relatively new, with no exact methods. Of utmost importance, particularly for circuits with >50K gates.

Uses: controlling manufacturing costs of most complex circuits

Goals: minimize area and delay, optimize testability, prepare for cell library mapping

Problem: identify critical path (slowest true path) in logic network, determine trade-offs between area and delay. Global optimization.

Standard techniques: transform by elimination, decomposition, extraction, simplification, and substitution. Both algorithmic and rule-based techniques are used. Kernel extraction, rectangle covering using matrices. Delay optimization through flow analysis.

Programs: MIS

Weakness: Heuristic approaches do not provide a systematic way to generate all equivalent multi-level circuits using logic transformations. No way to identify global optimum. Heuristics rely on iterative refinement (classical gradient search). Polynomial algebraic models (as opposed to Boolean models) are weak.

Recent developments: Observability and satisfiability don't-care analysis

Losp: Pervasion is ideal for multi-level optimization. We have a solution to systematic exploration of equivalent circuits. Identification of false critical paths appears to be straight-forward. Don't-care analysis is also straight-forward.

Comments: Losp approach is similar to BDD data-structuring, with less difficulty in determining variable ordering. Only difficulty is that the given formulation of circuits is often flat (few levels), so distribution is needed to build deeper forms. All the technical problems with variable ordering, distribution ordering, etc. show up for us, but in a much simpler context.

Extensions: Sequential circuits can be easily expressed as multi-level structures with registers as labeled coordination points.

Conclusions: Really big win. traditional problems seem to be greatly simplified, although still intractable.

Sequential Logic Optimization

Synchronous (clocked) models are well studied, but most methods are practical only for semi-trivial circuits. Asynchronous models fit naturally into dnets. Many open problems.

Uses: most circuits

Goals: state encoding and minimization, cycle-time optimization

Problem: determine valid states from FSM, find best weighted path in FSM, minimize register requirements.

Standard techniques: state transition diagram partitioning and restructuring, retiming via edge-weighted graph search

Programs: SIS

Weakness: FSM models do not correlate with optimization criteria (good for area improvement but not performance). FSM compatibility is not an equivalence relationship. State encoding depends on register type. Fanout and fanin (i.e. structure) interacts with optimization. Heuristic algorithms are hard to assess for quality.

Recent developments: FSM decomposition and factorization, interconnected FSMs, synchronous don't-care analysis.

Losp: Use time-labeled variables, deconstruct registers by collapsing time neighbors. Peripheral retiming appears to work well.

Comments: Retiming using time-labeled variables is a natural extension of Losp techniques. Rearranging registers appears to be straight-forward, since cycle-time is related to circuit topology. Registers appear to be used far more than necessary. Should connect this to programmed loops directly. Easy to estimate costs of unrolling non-abstract loops.

Extensions: Cyclic networks fit nicely into our model, but are relatively unexplored in conventional models.

Conclusions: Easy extension of Losp, no new techniques (other than transition analysis). Need to program the extension. Combine with logic optimization, since timing and combinatorial delay interact.

Technology-mapping, Library Binding

A necessary step in converting a mathematical structure (such as a logic network) to a physical structure (such as a transistor network).

Goals: optimize the physical resources mapped onto by a mathematical description.

Problem: graph covering with weighted nodes, determining the mapability of a library onto a logic network

Standard techniques: heuristic graph covering, rule-based models. Tree-covering using replicated logic.

Programs: specialized and proprietary

Weakness: Standard NP-hard issues. Binding algorithms are dependent on initial formulation of circuit, should be dependent only on behavior. Have circular dependency between scheduling, binding, and estimation of execution delay.

Recent developments: FPGA architectures, concurrent optimization and binding

Losp: Converting dnets to any arbitrary technology is straight-forward and fairly easy. Since optimization of area and delay are implicated in technology mapping, the Losp multi-level combinatorial strengths contribute strongly. FPGA technology mapping appears to be straight-forward.

Comments: Binding is essentially a multi-level issue, so Losp techniques should be expected to perform well. NOR-based decompositions are common, but not combined with pervasion techniques.

Extensions: various FPGA and cellular array hardware technologies

Conclusions: Big win due to the flexibility of dnet representations. Need to program a simple FPGA mapper in Losp.

Mapping Graph Theory to Dnets

Dnets express minimal logic graphs by confounding graph structure with logical implication. Dnet transformations, particularly the varieties of *pervasion* (extract, subsume, cancel), are deep in that they effect graph structure at a distance. This property makes pervasion particularly powerful for graph minimization.

As well, dnet arithmetic (with bound Boolean signals) is particularly efficient and parallel. Since optimization can be expressed as a series of

decision problems, and the tautology decision problem for dnets is very efficient, dnet search algorithms are inherently strong.

The standard graph optimization problems (below) do not become more simple with dnets. The advantage is that dnet notation and structure severely limits the space of graphs, making behavior invariant transformations efficient and direct. Since only distribution can be bidirectional, dnet algorithms have very constrained transformation spaces, again enhancing efficiency.

Graph-based algorithms

-- single-source shortest path (weighted connections)
  fundamental, shortest path from one place to another
  acyclic graphs done by topological sorting (ordered nodes)
  longest path = shortest by changing sign of weights
  relevance: longest as critical path, cycles are sequential

-- vertex cover
  each edge has at least one endpoint in set of vertices
  minimum size of vertex set
  vertex cover is NP while edge cover is P

-- vertex coloring
  no edge has same color/id on both ends
  minimum number of colors/ids

-- clique partitioning
  derivative of vertex coloring

In several cases (such as identification of critical paths and standardization for technology mapping), the dnet representation obviates the need for graph algorithms. Where they are needed (eg: covering and partitioning), the dnet structure minimizes the search inherent in standard algorithms.

Tasks to do within the DeMicheli graph theory model:

unify graph theory with Boolean algebra using distinctions
  done semi-adequately, lots of refinement needed
    losp has integrated Boolean and algebraic capabilities
solve fundamental problems with unified distinction model
  satisfiability done well
  covering done adequately but not implemented
  coloring not done but inherent in BM

apply solutions within task context
    scheduling and resource sharing not done
    Boolean minimization done as major focus, more to go
        two-level not done (subsumed)
        multilevel done well
        sequential not done but ready to do
    state minimization not done but studied
    testability mostly done but not implemented
    binding partially done
    routing not done
evaluate with pragmatics
    all modeled but not implemented

Sample Problems

We have sample (significant toy) problems done in boundary math for the following:

transcription and manipulation of form (MCNC benchmark set)
tautology and satisfiability identification
Boolean minimization
don't care optimization
mux technology mapping
true critical path identification
algebraic partitioning and rearrangement
hierarchical rearrangement and abstraction
selected layout tradeoffs (factorization, structure sharing)
BDD and all possible functions generation
selected evaluation and testability issues

References

Abramovici, Breuer, Friedman (1990) Digital Systems Testing and Testable Design. IEEE Press.

DeMicheli (1994) Synthesis and Optimization of Digital Circuits. McGraw-Hill.

Devadas, Ghosh, Keutzer (1994) Logic Synthesis. McGraw-Hill

Kain (1996) Advanced Computer Architecture. Prentice-Hall

Katz (1994) Contemporary Logic Design. Cummings

Murgai, Brayton, Sangiovanni-Vincentelli (1995) Logic Synthesis for Field-Programmable Gate Arrays. Kluwer