Heuristics in ILOC

William Bricken
September 2001

Almost all problems in Boolean minimization and circuit optimization are exponentially complex. Thus, ILOC relies on heuristics to manage resources and computational effort.

LOSP Heuristics

Losp reduces parens forms. It incorporates only one heuristic, level of effort. The *level of effort* parameter limits the amount of computational effort used to minimize a parens form. Each of the seven levels of the effort hierarchy applies all previous levels to the problem at hand.

1. Transcribe	Convert logic to parens
2. Clean and sort	Remove redundant boundaries; put in canonical ordering
3. Extract Literals	Extract literals for all deeper spaces
4. Extract Bounds	Extract compound bounds from deeper spaces
5. Cancel Bounds	Cancel structures at different depths
6. Insert Bounds	Insert bounds for virtual extraction
7. Minimize	Apply the above rules recursively until stable

Examples:

Transcribe	(and a b) ==> ((a)(b))
Clean	(((a)) (b ())) ==> (a)
Sort	((b a) d c) ==> (c d (a b))
Extract Literals	(a (b (a c))) ==> (a (b (c)))
Extract Bounds	((a b) (c (a b))) ==> ((c) (a b))
Cancel Bounds	((a b) (a (b))) ==> a
Insert Bounds	((a b) (c (a (d (b)))) ==> ((a b) (c (a (d))))
Minimize	((a b)(a (b))((a) b)((a)(b))) ==> ( )

In addition, the application of Distribution can be controlled at two levels, full distribution and partial distribution. Full distribution would make no changes to the second example below:

Full distribution	((a b) (a c) (a d)) ==> a ((b)(c)(d))
Partial distribution	((a b) (a c) (b d)) ==> ((a ((b)(c))) (b d))

Pun Heuristics

Pun manages and minimizes circuit structures. It calls Losp to reduce parens forms, which may be seen as subgraphs of a circuit. As well, Pun rearranges circuit structure to meet technology mapping objectives.

In the pun format, the circuit body consists of cells. Each cell is an id and a parens form. When each parens form can be logically interpreted as a gate, the circuit has been technology mapped.

Pun's primary responsibility is managing the expansion of parens forms across cells. This is equivalent to adding and removing internal wires in a conventional circuit, with the added flexibility that the parens form can express an arbitrary logic function. Some architectures, such as a systolic array, cannot be fully expanded into a parens representation; the internal branchiness is exponential.

In addition to managing exponential growth in parens forms in cells, Pun manages the size of parens form sent to Losp. Pun can call Losp with a parametric level-of-effort, and/or it can elect to give Losp a parens reduction problem of a certain size.

It is always desirable to fully expand circuits when possible, since Losp would have the full functionality as a reduction context. In converting between parens and pun, the amount of expansion and partitioning of parens forms is controlled by several parameters, including the following:

maximum number of variables in a parens form
maximum depth of nesting of a parens form
maximum size of a parens form which can be substituted
maximum depth of a parens form which can be substituted
maximum number of replicated variable references in a parens form
maximum number of variable references within the entire Pun body
amount of redundancy of all variable references in a parens form
maximum size of a parens form substituted into
maximum depth of a parens form substituted into
maximum number of occurrences of a substituted form in the entire body

These and other constraints can be combined functionally under the parameter *definition-of-too-big*. Some examples of these combinations:

literals-by-occurrences
depth-by-size
depth-by-occurrences-by-size
depth-by-size-by-redundancy
totaloccurs-by-size

Technology Mapping

Pun forms can be configured to meet structural constraints. These include

- maximum fanout from a cell
- maximum fanin to a cell
- XOR and MUX abstraction
- library abstraction
- spatial OR expansion
- coalesce cells

Other technology mapping options include:

- distinction network
- specific gate combinations
  - AND-OR-NOT
  - NOR
  - NAND
  - AND-OR-NOT-XOR-MUX
  - NOR-XOR-MUX
  - XLCB
- critical path length

Sorting

Putting each parens form in Losp in canonical order reduces the effort of all pattern-matching algorithms. Ordering is perhaps the most powerful tool for reducing computational effort.

Cells in Pun are also ordered, permitting efficient pattern matching and technology mapping across cells.

Filters

Internal loops in Losp are guarded by input filters intended to reduce computational effort. The cost of a filter is always less than the cost of the guarded process. In many cases of void-based programming, identifying a structure and reducing it require the same amount of effort, so that search for candidate structures is wasteful.

In rarer cases, the code following a filter involves many smaller tests of equal cost as the filter. Here filters as used to simplify traces and calls.

All filters in Losp use simple structural information about a parens form. Some of these filters and their uses follow:

- no-duplicate-variables
  suppress bound operations (extract, cancel, insert)
- unate-or-binate-variable
  select for insertion, for case analysis
- most-common-variable
  select for case analysis, for partial distribution
- shallowest-variable
  select for case analysis

Filters in Pun use structural information about the parens form in each cell:

- fully-expanded
  no substitutions are possible
- special structures
  treat as special:
    XOR, MUX patterns
    OR patterns
    coalesced cells
    registers