Logic Restructuring for Timing Optimization

Outline:

• Definitions and problem statement
• Overview of techniques (motivated by adders)
  - Tree height reduction (THR)
  - Generalized bypass transform (GBX)
  - Generalized select transform (GST)
  - Partial collapsing (?)
Timing Optimization

Factors determining delay of circuit:

- Underlying circuit technology
  - Circuit type (e.g. domino, static CMOS, etc.)
  - Gate type
  - Gate size

- Logical structure of circuit
  - Length of computation paths
  - False paths
  - Buffering

- Parasitics
  - Wire loads
  - Layout
Problem Statement

Given:
• Initial circuit function description
• Library of primitive functions
• Performance constraints (arrival/required times)

Generate:
an implementation of the circuit using the primitive functions, such that:
• performance constraints are met
  - circuit area is minimized
Current Design Process

Behavioral description

Logic and latches

Logic equations

Gate netlist

Layout

Behavior Optimization (scheduling)

Partitioning (retimeing)

Logic synthesis

- Technology independent
- Technology mapping

Timing driven place and route
Technology mapping for delay
Overview of Solutions for delay

- Circuit re-structuring
  - Rescheduling operations to reduce time of computation
- Implementation of function trees (technology mapping)
  - Selection of gates from library
    - Minimum delay (load independent model - Kukimoto)
    - Minimize delay and area (Jongeneel, DAC'00)
      (combines Lehman-Watanabe and Kukimoto)
- Implementation of buffer trees
  - Touati (LT-trees)
  - Singh
- Resizing

Focus here on circuit re-structuring
Circuit re-structuring

Approaches:
Local:
• Mimic optimization techniques in adders
  - Carry lookahead (THR tree height reduction)
  - Conditional sum (GST transformation)
  - Carry bypass (GBX transformation)
Global:
• Reduce depth of entire circuit
  - Partial collapsing
  - Boolean simplification
Re-structuring methods

Performance measured by
  • levels,
  • sensitizable paths,
  • technology dependent delays

• Level based optimizations:
  - Tree height reduction (Singh '88)
  - Partial collapsing and simplification (Touati '91)
  - Generalized select transform (Berman '90)

• Sensitizable paths
  - Generalized bypass transform (Mcgeer '91)
Re-structuring for delay: tree-height reduction

Critical region

Collapsed Critical region

Duplicated logic
Restructuring for delay: path reduction

Singh '88
**Generalized bypass transform (GBX)**

- Make critical path false
  - Speed up the circuit
- Bypass logic of critical path(s)

\[ f_m = f \rightarrow f_{m+1} \rightarrow \cdots \rightarrow f_n = g \]

McGeer '91
GBX and KMS transform

GBX gives little area increase, BUT have now created an untestable fault (on control input to multiplexer)

KMS transform: (remove false paths without increasing delay)

- $f_k$ is last node on false path that fans out.
- Duplicate false path $\{f_1, ..., f_k\} \rightarrow \{f'_1, ..., f'_k\}$
- $f'_j$ fans out to every fanout of $f_j$ except $f_{j+1}$, and $f_j$ just fans out to $f_{j+1}$
- Set $f_0$ input to $f_1$ to controlling value and propagate constant (can do because path is false and does not fanout)

KMS results

- Function of every node, except $f_1, ..., f_k$ is unchanged
- Added k-1 nodes
- Area added in linear in size of length of false paths; in practice small area increase.
KMS (Keutzer, Malik, Saldanha '90)

Delay is not increased
End of lecture 20
Generalized select transform (GST)

Late signal feeds multiplexor

\[
\begin{array}{c}
\text{out} \\
\downarrow \\
\text{out}
\end{array}
\]

Berman '90
Note:
Boolean difference =
\[ \frac{\partial h}{\partial a} = h_a \oplus h_{\bar{a}} \]
GST vs GBX

- Select transform appears to be more area efficient
- But Boolean difference generally more efficiently formed in practice
- No delay/speedup advantage for either transform
- Need
  - one MUX per fanout in GST,
  - only one MUX in GBX
Technology independent delay reductions

Generally THR, GBX, GST (critical path based methods) work OK, but not great

Why are technology independent delay reductions hard?

Lack of fast and accurate delay models

- # levels, fast but crude

- # levels + correction term (fanout, wires,...): a little better, but still crude (what coefficients to use?)

- Technology mapped: reasonable, but very slow

- Place and route: better but extremely slow

- Silicon: best, but infeasibly slow (except for FPGAs)
Clustering/partial-collapse

Traditional critical-path based methods require
- Well defined critical path
- Good delay/slack information

Problems:
- Good delay information comes from mapper and layout
- Delay estimates and models are weak

Possible solutions:
- Better delay modeling at technology independent level
- Make speedup, insensitive to actual critical paths and mapped delays
Clustering/partial-collapse

Two-level circuits are fast
- Collapse circuit to 2-level - but
  - Huge area penalty
  - Huge capacitive loading on inputs (can be much slower)

To avoid huge area penalty
- Identify clusters of nodes
  - Each cluster has some fixed size
- Perform collapse of each cluster
- Simplify each node

Details
- How to choose the clusters?
- How to choose cluster size?
- How to simplify each node?
Lawler's clustering algorithm

- **Optimal in delay:**
  - For a given clustering size
- **May duplicate nodes (hence possible area penalty):**
  - Not optimal w.r.t duplication
  - Use a heuristic
- **Fast: $O(m \times k)$**
  - $m =$ number of edges in network
  - $k =$ maximum cluster size
Clustering algorithm - overview

- **Label phase:** (k is cluster size)
  - If node u is an input, label(u) := L := 0
    - Else L := max label of fanin of u
  - If (# nodes in TFI(u) with (label = L) >= k)
    label(u) := L+1

- **Cluster phase:** (outputs to inputs)
  - If node u is an output, L := infinity
    - Else L := max label of fanouts of u
  - If (label(u) < L) then create a new cluster with “root” u and with members all the nodes in TFI(u) with label = label(u)

- **Collapse phase:** (order independent)
  - Collapse all nodes in a cluster into a single node
  - Note: a node may be in several clusters (causes area increase
Example of clustering

Result: Lawler's algorithm gives minimum depth circuit

Typically,
• we decompose initial circuit into 2-input NANDs and invertors.
• then cluster size $k$ reflects # 2-input NANDs to be collapsed together.
Choosing $k$

- $I(k)$: number of levels, given $k$
- $d(k)$: duplication ratio
  - Number of gates in cluster network divided by number of gates in original network
- Determine $k_0$ where $k_0/d(k_0) \approx 2.0$
- For every $k$ from 2 to $k_0$, compute $d(k)$, $I(k)$
  - Use exhaustive enumeration: label and cluster (without collapse) for each $k$.
  - Each iteration is $O(|E|k)$
- Choose $k$ such that
  - $I(k)$ is minimized
    - Break ties using $d(k)$
      - Minimize $d(k)\)
Area recovery

Area increase is due to node duplication -
  - this occurs when node is in multiple clusters

Two solutions:
  - Break clusters into smaller pieces off critical path
  - After cluster and collapse, recover area
Relabeling procedure:

Attempt to increase node labels without exceeding cluster size

In reverse topological order

Start: assign

\[ \text{new-label}(O_i) = \max_{j \in PO} \{ \text{label}(O_j) \} \]

Increase label(u) if

- new-label(u) \( \leq \) label(v) for each fanout v and
- new-label(u) = new-label(v) for each fanout v only if label(u) = label(v) before relabeling, and
- no cluster size is violated
Relabeling example

before

after

27
Post-collapse area recovery

- Do algebraic factorization, but
  - Undo factorization if depth increases
- Full_simplify
  - Only consider node \( v \) as possible fanin of a node (\( v \) introduced by \( \phi \lor \lambda \_ \alpha \mu \pi \lambda \phi \psi \) using don’t cares) if level of \( v < \) level of node.
- Redundancy removal
Conclusions

• Variety of methods for delay optimization
  - No single technique dominates (KJ Singh PhD thesis)
• When applied to ripple-carry adder get
  - Carry-lookahead adder (THR)
  - Carry-bypass adder (GBX)
  - Carry-select adder (GST)
  - ? (partial collapse)
• All techniques ignore false paths when assessing the delay and critical regions
  - Can use KMS transform to eliminate false paths without increasing delay (area increase however).