

California Institute of Technology
Department of Computer Science
Electronic Design Automation

CS137a, Winter 2002

Assignment #4

Monday, March 11

Due: Wednesday, March 20, 5:00pm.

Resources You are free to use any books, articles, notes, or papers as references. Provide citations in your writeup as appropriate.

Collaboration Each student is expected to do his/her own work.

Writeup Writeup should be in an electronically readable format (HTML or PDF preferred, Word tolerated).

Problems

1. Develop an algorithm to map a logic expression into a pseudo 2-level Asynchronous PLA implementation which minimizes the *expected delay* of the output.
 - **Input:** logical equation or truth table and a probability associated with each minterm
 - We use pseudo-2-level here to mean the implementation will be limited to a sum of products, but the sum stage should be decomposed into 2-input gates. The shape of this stage determines your (controllable) path delay. Assume each 2-input gate has unit delay.
 - The PLA should compute the true and complement of the output (you may assume a single-output function).
 - Output encoding is two bits: 00=no result, 01=false, 10=true.
 - Imagine a pre (or post)-charge implementation where the output is precharged 00 (no-result); true and complement logic compute; one of them will eventually become true; precharge resets.
 - Once you have a cube cover, you might think about adapting Huffman encoding to minimize expected delay based on cube probability.
 - Assign each minterm probability to a single cube to avoid overcounting; note that high probability cubes will increase your ability to use asymmetric delay paths through the sum stage to reduce expected delay.
 - State any additional assumptions or clarification you find necessary in order to solve the problem.
 - Provide pseudocode for your algorithm along with rationale and explanation to support its operation.
 - Identify the runtime of your algorithm.
 - In what ways, if any, is the algorithm suboptimal? (I don't necessarily expect an optimal solution here.)

Probability	minterm	value
0.0125	0000	0
0.0125	0001	0
0.0125	0010	0
0.0125	0011	0
0.0125	0100	0
0.0125	0101	0
0.1250	0110	1
0.0125	0111	0
0.1250	1000	1
0.0125	1001	0
0.1250	1010	1
0.1250	1011	1
0.1000	1100	0
0.1000	1101	0
0.1000	1110	0
0.1000	1111	0

Note: 4 on minterms, 8 off minterms. If we keep minterms and use a separate, balanced OR-tree, the $E(\text{delay}) = 0.5 \cdot 2 + 0.5 \cdot 3 = 2.5$

Computing prime implicants:

Output	Probability	minterm
true	0.125	0 1 1 0
	0.250	1 0 - 0
	0.125	1 0 1 -
false	0.4000	1 1 - -
	0.0500	0 0 - -
	0.0250	- - 0 1
	0.0125	0 - 0 0
	0.0125	- 1 1 1

Note: carefully assigning each minterm probability to a single cube here. A flat encoding of these cubes:¹

$$\begin{aligned}
 O_{true} &= (\bar{a}\bar{b}c + a\bar{b}\bar{d}) + \bar{a}bc\bar{d} \\
 O_{false} &= ((ab + \bar{a}\bar{b}) + (\bar{c}d + \bar{a}\bar{c}\bar{d})) + bcd \\
 E(\text{delay}) &= 2 \cdot 0.125 + 2 \cdot 0.250 + 1 \cdot 0.125 + 3 \cdot 0.4 + 3 \cdot 0.05 + 3 \cdot 0.025 + 3 \cdot 0.0125 + 1 \cdot 0.0125 \\
 &= 2.35
 \end{aligned}$$

Balancing so short paths favor high probability cubes

$$\begin{aligned}
 O_{true} &= \bar{a}\bar{b}\bar{d} + (\bar{a}bc\bar{d} + \bar{a}\bar{b}c) \\
 O_{false} &= ab + (\bar{a}\bar{b} + (\bar{c}d + (\bar{a}\bar{c}\bar{d} + bcd))) \\
 E(\text{delay}) &= 1 \cdot 0.250 + 2 \cdot 0.125 + 2 \cdot 0.125 + 1 \cdot 0.4 + 2 \cdot 0.05 + 3 \cdot 0.025 + 4 \cdot 0.0125 + 4 \cdot 0.0125 \\
 &= 1.425
 \end{aligned}$$

¹I should, perhaps, assign the probabilities to cubes differently for this calculation, but did not want to complicate the example any further.

2. Consider scheduling a DAG onto a homogeneous array of resources (*e.g.* DPGA, ALU-array) in time-multiplexed fashion. We want to find a schedule that minimizes the live state required to evaluate the DAG.

- DAG nodes are already “feasible” for mapping to the array; that is, each DAG node can map onto one array compute node.
- Each time-slice stage may hold at most S_n total operations.
- Each stage may cascade logic up to depth S_d .
- We start by computing a schedule constraining only the compute capacity and depth as above to determine a feasible number of time-steps, N_{step} .
- We then pick a target $N_{target_steps} \geq N_{step}$
- Our goal is to find a schedule that minimizes the number of live state bits within this time.

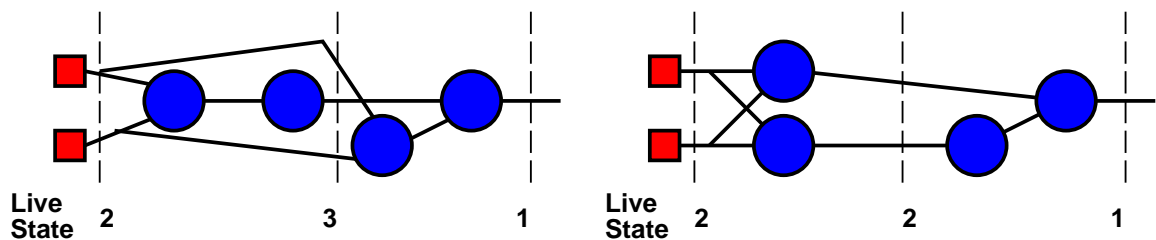
Develop four approaches to this problem based on:

- Resource constrained scheduling with 2 resources (logic blocks and state bits) using branch-and-bound scheduling with pruning; include constraints which allow you to limit branching factor.
- Maxflow/Mincut
- C-slow Retiming ($C = S_d$)
- different method of your choice

Each of the suggested base approaches attacks a piece of this directly, but may not address the entire problem.

For each algorithm:

1. What does the base algorithm attack directly?
2. How can we apply it (perhaps with heuristic extensions) to address this problem?
3. Describe your algorithm, rationale, runtime.
4. Describe the relative strengths and weakness relative to other approaches?



Two schedules for a graph mapped onto $S_n = 2$, $S_d = 2$. One mapping requires only two live state bits while the other mapping requires three.