University of Pennsylvania
Department of Electrical and Systems Engineering
Electronic Design Automation

ESE535, Spring 2009 Assignment #4 Monday, March 16

Due: Wednesday, April 8, beginning of class.

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

Collaboration: Please work independently on this assignment. For problem 2 only, you may discuss general algorithmic strategies and help each other with the compiler, build environment, and debugging, but each student should develop his or her own solution. If you do discuss strategy or get debugging help, please acknowledge in your writeup.

Writeup: Writeup should be in an electronically readable format (HTML or PDF preferred—I do not want to decipher handwriting or hand-drawn figures). State any assumptions you need to make.

Problem 1 (offline algorithm development) [3pts]

Verify FSM Communication: Consider a pair of communicating Finite-State Machines (FSMs) each controlling an associated datapath. Let's call them A and B. There is a single channel in each direction between them (AtoB and BtoA). State machine A has the control signals SendAtoB and RcvBtoA. State machine B has the control signals SendBtoA and RcvAtoB. Both state machines have a single designated start state and share a common reset signal that returns them to the start state; the two machines operate from the same clock signal. Some inputs are common to the two machines.

For correct operation, machine B should signal RcvAtoB in the same cycle as machine A signals SendAtoB (and similarly machine A should signal RcvBtoA in the same cycle as machine B signals SendBtoA). If the machines do not activate these complementary signal pairs simultaneously data may be lost (*e.g.* if A sends and B does not receive) or one machine may receive garbage (*e.g.* if A receives but B does not send). Since the FSMs run independently after reset, it is possible that an incorrectly designed pair of FSMs might not always simultaneously present the complementary communication signals.

Provide a verification algorithm that takes in two such FSMs, A and B, and a description of their common input signals and determines whether or not all communication operations are correctly paired.

Problem 2 (programming) [4pts]

Develop a routine to perform common subexpression elimination [CSE] (node substitution) on graphs in the format we have used for previous assignment.

Provided starting point is available in ~`ese535/spring2009/assign4.tar` on `eniac`. This uses similar buildings blocks as the assignment 2 and 3 framework. Put your routine in `your_cse.c`.

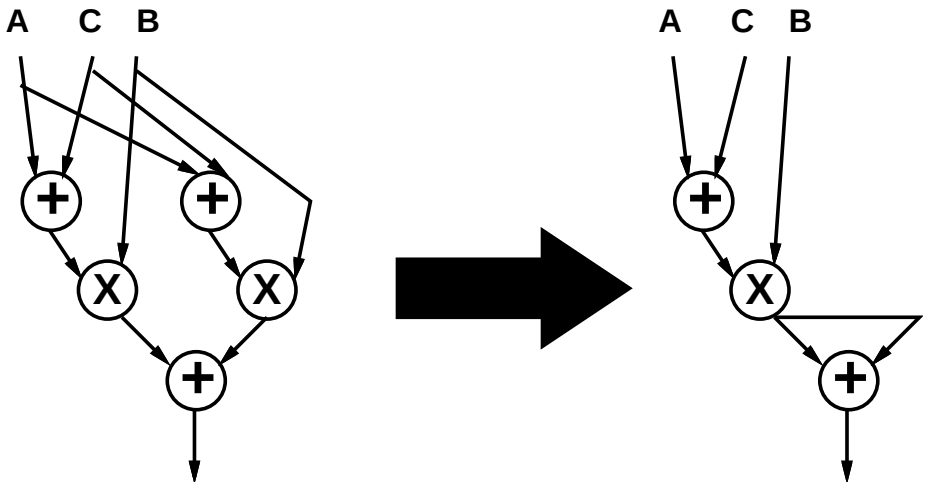
Note the routine `remove_unused_nodes` in `graph.c` and called from `driver4.c` and the new library node type `UNUSED`. This is provided to deal with the logistics of removing nodes from the graph. It allows you to mark a node you no longer need as `UNUSED` during your CSE optimization so that this routine can clean it up before the graph is written out to the file by the provided `write_graph`.

With this cleanup of unused nodes done for you, the hardest part of the algorithm may be properly changing the graph to support substitution. Graphs written out by this optimization should be usable with the your previous assignments.

The benchmark set is the same as before (with one small example that will benefit from CSE for illustration and debugging). Some of the benchmark designs have no CSE opportunities. Some have as many of 6% of their nodes that can be removed.

Turnin:

1. Your code (a tar file as on previous assignments that can be unpacked and built)
2. Summary of your results across the provided benchmark set including:
   - nodes before your CSE
   - nodes after your CSE
   - percentage reduction in nodes due to CSE



Problem 3 (offline examples and algorithm) [3pts]

For this problem we will use a simple energy model. We lump all capacitance for the nodes to the LUT input and assume this is the same for all LUTs (so this ignores effects of wire lengths, just as the unit delay assumption does). We assume the dominant energy is the energy taken to switch each of these inputs. The energy is thus:

$$E_{circuit} \propto \sum_{\text{all gate inputs}} (P_{switch}(\text{input}))$$

For covering, assume the input netlist is already annotated with the switching probability of each gate. Particularly, gate inputs in the input netlist which are hidden inside a mapped gate during covering do **not** contribute to the energy for the mapped circuit.

(a) Show an example where all three optimization criteria would give rise to different optimal coverings:

- area (in LUTs)
- delay (in LUT delays; you may assume fanout does not affect delay)
- energy (as described above; you may assign switching probabilities to the gate inputs in the input netlist as you need to construct your example)

(b) Show an example where the mincut from a target node is 3 but the best cut to use for k=4-LUT area minimization is of size 4.

(c) Give the trivial $O(n^{k+1})$ algorithm for exploring all k-feasible cuts. (FYI: [1] gives a more complicated algorithm which is significantly faster in practice.)

References

[1] Jason Cong and Yuzheng Ding. On area/depth trade-off in LUT-based FPGA technology mapping. *IEEE Transactions on VLSI Design*, 2(2):137–148, June 1994.