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

ESE535, Spring 2011 Assignment #8 Monday, April 25

Due: Monday, May 9, 12PM (noon).

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

Collaboration There is no collaboration on this exercise.

Please include a statement on your final submission:

> I, *your-name-here*, certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this final exercise.

You can review the Code of Academic Integrity here: `http://www.upenn.edu/academicintegrity/ai_codeofacademicintegrity.html`

Questions Email instructor with any questions you have to clarify the assignment problems. Instructor will be traveling May 1–6, so do not expect a timely response during that week. Recommendation is to pose your questions before May 1st.

Writeup Turn-in assignments on blackboard (PDF preferred). See details on course web page. No handwriting or hand-drawn figures. State any assumptions you need to make.

Lateness This assignment cannot be turned in late for partial credit.

Grading You will be graded on the best 3 answers turned in. Each problem is worth 5 points, for a total of 15 possible points. You may choose to complete only three answers or to complete all four.

Problems

1. Formulate Instantaneous Single-Corner Turn Routing as a SAT decision problem. Use the architecture model from Assignment 5, except we are only concerned with minimizing $ct_{instant}$:

$$ct_{instant}(x,y) = \max_{\text{all timesteps } j} \left(ct_{instant_j}(x,y)\right)$$

$$ct_{instant} = \max_{\text{all PEs } (x,y)} \left(ct_{instant}(x,y)\right)$$

The SAT decision problem will be for a particular number of corner turns and timesteps; the problem should be satisfiable *if-and-only-if* the placed netlist is routable in the specified number of corner turns and timesteps. Your answer should describe how the SAT formula is composed when you want to ask for any particular number of corner turns and timesteps. Assume you have already placed the design and you have already determined the ASAP and ALAP times for each of the nodes.

   (a) define the Boolean variables you will use
   (b) identify the classes of constraints to impose
   (c) describe how to generate the clauses to implement each constraint
   (d) describe how to go from a satisfying assignment to a legal route

2. Develop an optimization routine to reduce the logic required for datapath synthesis by sharing Common Operator Subgraphs between two hyperblocks that are known not to execute simultaneously. In Callahan's compilation of C to reconfigurable logic, he generated a set of hyperblocks that he then mapped to spatial logic on an FPGA. Only one of the hyperblocks is ever active at a time. Consequently, it might be beneficial to share logic between hyperblocks. The most trivial case would be two hyperblocks that contained equivalent dataflow graphs (here equivalence would be the choice of operators and their connectivity; the input control flow and input variables might be different between the two hyperblocks; similarly, the output variables and control flow might be different as well). A more general case would be to share some subset of the dataflow graph (*e.g.* maybe one graph computes Z=(A+B+C*D)>>2 and another computes Q=(R+S+T*U)%7; both graphs share the subgraph i1+i2+i3*i4).

   Provide an algorithm to minimize the amount of unique datapath logic required by maximally sharing subgraphs among a pair of hyperblocks. That is, your goal is to minimize the area of the operators that must be spatially implemented, including any muxes that you add to the logic to allow subgraph sharing.

   Assume you have:

   - A finite set of operators (e.g., +, *, -, %, >> c), and an area cost for each (e.g., area(+), area(*)).
   - An area cost for the 2-input multiplexer (i.e., area(mux2)).

   (a) Provide an English description of your strategy.
   (b) Provide pseudocode for your mapping algorithm.
   (c) Show how your algorithm works on:

	Hyperblock 1		**Hyperblock 2**
1	t1=A-B		t1=Q-R
2	t2=abs(C)		t2=abs(t1)
3	t3=ilog2(D)		t3=S<< 2
4	t4=t2+t3		t4=T+U
5	t5=E*t4		t5=t3+t4
6	t6=t1*t5		t6=t5*t2
7	Y=t6+F		Y=t6+V

when we have the costs:

Operator	**Area**
mux2	2
+, -	3
<<2	1
abs	3
ilog2	4
*	30

3. Perform two-level logic optimization to minimize Pterm switching energy. In addition to the unoptimized two-level logic, assume that you have as input:

   - $P(m_i, m_j)$=probability that input switches from minterm m_i to minterm m_j for all minterm pairs.

   - $P(i)$=the switching probability for input i.

$$E(p) = P(p) \times C_{or} \times N_{puse}(p) + C_{and} \times \left(\sum_{\text{all inputs } i \text{ appearing in } p} (P(i)) \right) \quad (1)$$

$$E = \sum_{\text{all pterms } p} (E(p)) \quad (2)$$

   $N_{puse}(p)$ is the number of outputs that use Pterm p

   (a) Define $P(p)$ in terms of $P(m_i, m_j)$'s.
   (b) Revise Prime Implicant covering to solve this problem:
      i. Provide an English description of your strategy
      ii. Show pseudocode for your revised algorithm

4. Optimize an FSM by identifying and removing redundant states.

   (a) Define the conditions under which two states are equivalent.
   (b) Describe how you would efficiently verify this condition.
   (c) Develop an algorithm to remove all redundant states from an FSM
      i. Provide an English description of your strategy
      ii. Show pseudocode for your algorithm
      iii. Identify the runtime complexity of your algorithm as a function of $|S|$, the number of states in the original FSM and $|E|$, the number of edges in the original FSM.