CS69001: Computing Lab – I
Autumn 2009

Assignment 7

Parallel Boolean circuit evaluation using POSIX threads

Due: November 13, 2009 (Friday)

Consider Boolean circuits with m input variables and n output variables. Assume that the circuit is made up of OR, AND and NOT gates only. In this assignment, you are required to use threads in order to evaluate the n output values in parallel for any given set of input values.

Part 1 **(20)**

Read information about a circuit from a file, and store the data in a global data structure. The organization of the input file is explained in the next page.

Part 2 **(20)**

Create n threads to evaluate the n output values in parallel. The threads run independently, and recursively compute the output values. After the output values are computed, the threads are joined by the master thread.

Part 3 **(30)**

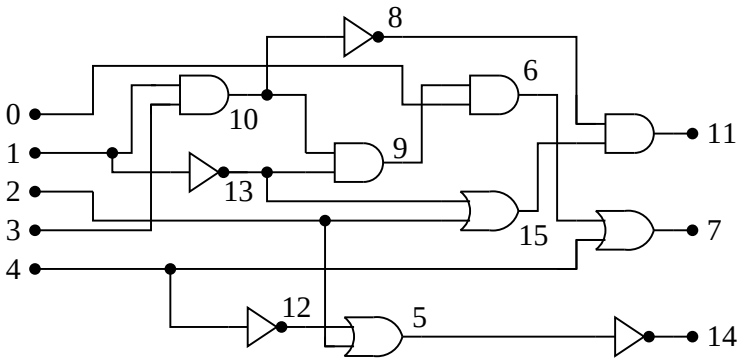
Repeat Part 2 with memoization. Some (intermediate) values may be used by different threads. For example, in the circuit given in the next page, the value x_6 is needed for the computation of both x_{13} and x_{14}. If one thread has already computed a shared value, other threads may use that value straightaway without recomputing them. Keep a global table of values, accessible to every thread. When a thread computes the value at a particular point x_i, it populates the i-th entry in the table by that computed value. Later, if a thread (same or different) wants to use the value of x_i, it reads that value from the table. Of course, the table must maintain some flags against values that are not yet computed by any thread. If x_i is needed by some thread, and the i-th location in the table is uninitialized, then that value is computed by the thread.

Part 4 **(30)**

Repeat Part 3 with an additional constraint. If a thread is currently computing a value x_i, it indicates that using a marker shared by all the running threads. When a thread needs the value of some x_i, it makes multiple checks. First, if the value of x_i is already available in the global table of Part 3, the thread simply reads the value. Second, if the value is uninitialized, the thread sets the marker against x_i, computes this value, and updates the i-th entry in the global table by the computed value. Finally, if the thread sees that another thread is currently computing the value of x_i, it waits until that other thread finishes updating the i-th entry in the global table. In this part, you need to use some synchronization mechanism.

Submit a single C/C++ file solving all the above parts. The file must contain your name and roll number.

A combinational circuit is shown in the figure below. It has five input variables, three output variables, and eleven gates. Therefore, we need to keep track of $5 + 11 = 16$ points in the circuit. The inputs are numbered $0, 1, 2, 3, 4$. The gate outputs are numbered $5, 6, \ldots, 15$. The output points are $11, 7, 14$.



Boolean values at the five input points are provided. The gate outputs are recursively computed as follows.

$$
\begin{array}{lcllcllcl}
x_5 & = & x_{12} + x_2, & x_9 & = & x_{10}x_{13}, & x_{13} & = & x_1', \\
x_6 & = & x_9 x_0, & x_{10} & = & x_1 x_3, & x_{14} & = & x_5', \\
x_7 & = & x_6 + x_4, & x_{11} & = & x_8 x_{15}, & x_{15} & = & x_{13} + x_2. \\
x_8 & = & x_{10}', & x_{12} & = & x_4', & & &
\end{array}
$$

This information is stored in the input file as follows. An explanation of each line is provided beside the line.

```
5                   The count of input values
3                   The count of output values
11 7 14             The serial numbers of the output values
11                  The count of gates in the circuit
5 OR 12 2           Gate 5 computes the OR of x12 and x2
6 AND 9 0           Gate 6 computes the AND of x9 and x0
7 OR 6 4            ...
8 NOT 10            Gate 8 computes the complement of x10
9 AND 10 13         ...
10 AND 1 3
11 AND 8 15
12 NOT 4
13 NOT 1
14 NOT 5
15 OR 13 2
```