Problem 1: Use the simple model to compute the cost and delay (critical path length) of the inferred hardware for module behav_merge from Homework 5. This module has two inputs, a and b, each of which is an n-element sorted sequence of w-bit unsigned integer values. Output x is a 2n-element array of w-bit quantities. The module assigns elements of a and b to x so that x itself is a sorted sequence of the elements from a and b.

Show the cost and delay of behav_merge in terms of n and w. The Homework 5 module appears below. Use the tree implementation of multiplexors for cost and delay. (See the simple model notes.) Make reasonable optimizations, such as using the same multiplexor for a[ia] and a[ia++]. Avoid tedious optimizations such as varying the number of bits in ia and ib.

module behav_merge
  #( int n = 4, int w = 8 )
  ( output logic [w-1:0] x[2*n], input uwire [w-1:0] a[n], b[n] );

logic [$clog2(n+1)-1:0] ia, ib;
always_comb begin
  ia = 0; ib = 0;
  for ( int i = 0; i < 2*n; i++ )
    x[i] = ib == n || ia < n && a[ia] <= b[ib] ? a[ia++] : b[ib++];
end
endmodule

Problem 2: As was probably mentioned, a proper n-element Batcher odd/even merge module is constructed from $\frac{n}{2} \lceil \log n \rceil$ sort2 modules, and the critical path length through a merge module is $\lceil \log n \rceil$ sort2 delays.

If the previous problem was solved correctly then the cost and critical path length of behav_merge should be much larger than a Batcher merge. But the behavioral code in behav_merge has a run time of $O(2n)$ running as an ordinary program, and consumes $O(2n)$ memory, both of which are optimal for an algorithm that must operate on all of 2n items. In fact, recursively applied code based on behav_merge can sort a sequence in $O(n \log n)$ time, which is the best one can normally get in many cases.

What is it about the hardware realization of behav_merge that makes it so much less efficient than the software realization? Your answer should consider how much hardware is being used at each moment in time.