3 How Cool is Generalized Race Logic! (20 points) :

One application of generalized race logic is to solve the 1-0 knapsack problem. The 1-0 knapsack problem states, suppose you have a bag with maximum capacity C. Each item i in the bag has weight w_i and value v_i. Place items into the bag such that the bag has the maximum value, constrained by the fact that the weight in the bag must be below C. The knapsack problem has many applications, such as download managers that divide up downloads into chunks (weight) and try to maximize the chunks in the packet, constrained by the maximum capacity.

According to MIT course recitation on https://courses.csail.mit.edu/6.006/fall11/rec/rec21_knapsack.pdf, the Knapsack problem can be reformulated as finding the shortest path in a directed acyclic graph, where the number of vertices and edges in the graph is n * C, where n is the number of items in the bag. The value of the edges is the value of the item i. The advantage of the GRL approach compared to the binary approach is that in the binary approach, you need to traverse all V + E edges in order to find the shortest path. However, in the GRL approach, the compute time is on the order of the length of the shortest path. Assuming that the values of the items do not grow too large, the GRL solution will have better runtime and lower energy cost because entire exploration of the DAG is not needed. Furthermore, a queue will not be needed, as usually required in shortest path algorithms. Therefore, additional memory will not be required in addition to the memory for the acyclic directed graph.

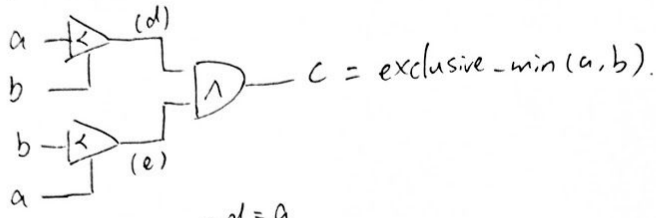
Furthermore, any variation of the graph traversal, such as Longest Increasing Subsequence, Longest Common Subsequence, and maximum value in a grid can be solved by representing the DP subproblems as an directed acyclic graph and running shortest/longest path traversal. Real applications of this are the diff algorithm which calculates the difference between two files (https://en.wikipedia.org/wiki/Diff), by using the longest increasing subsequence algorithm.

4 I'm Something of a Scientist Myself ! - Norman Osborn (20 points) :

4. (1)

exclusive_min(a, b) = $(a < b) \wedge (b < a)$

Diagram:



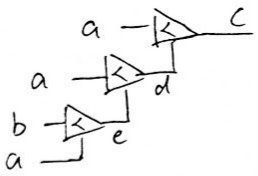
proof: if $a < b \Rightarrow \begin{cases} d = a \\ e = \infty \end{cases} \Rightarrow c = a.$

if $b < a \Rightarrow \begin{cases} d = \infty \\ e = b \end{cases} \Rightarrow c = b.$

if $a = b \Rightarrow \begin{cases} d = \infty \\ e = \infty \end{cases} \Rightarrow c = \infty.$

4. (2).

Greater_than(a, b) = $(a < (a < (b < a)))$.



proof: if $a < b \Rightarrow \begin{cases} e = \infty \\ d = a \\ c = \infty. \end{cases}$

if $b < a \Rightarrow \begin{cases} e = b \\ d = \infty \\ c = a. \end{cases}$

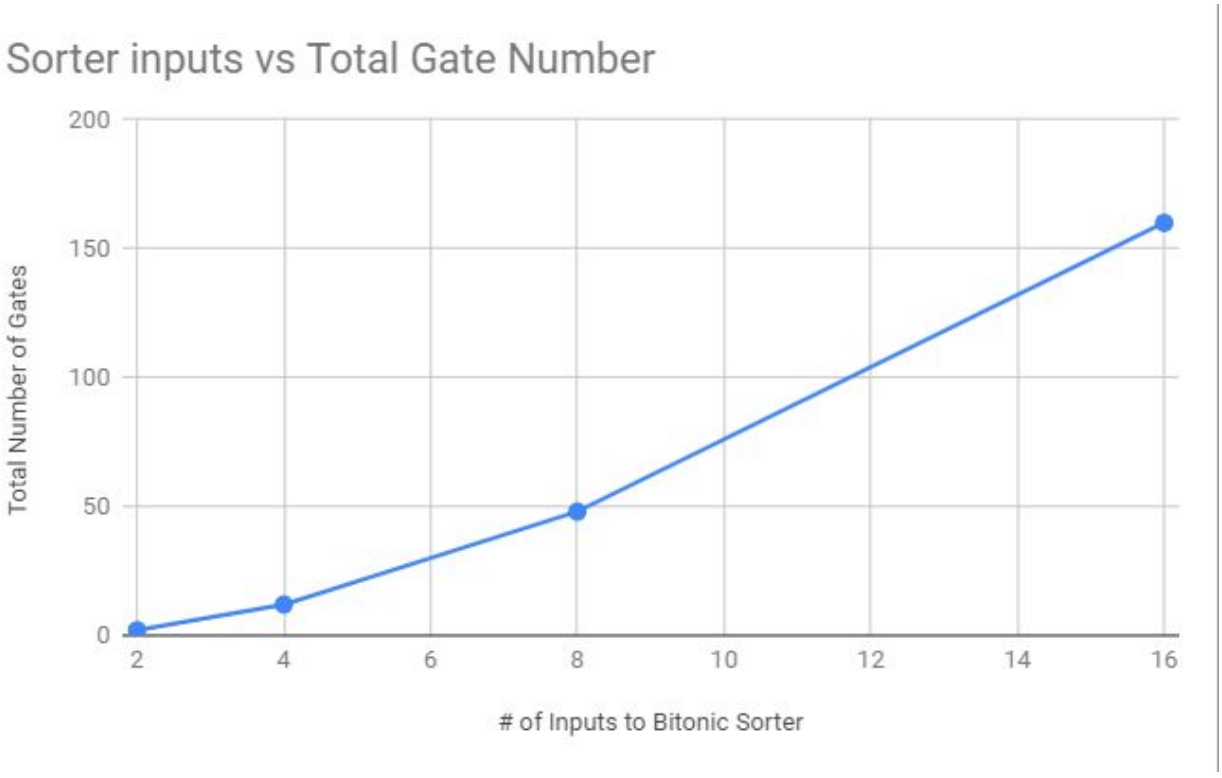
if $b = a = \begin{cases} e = \infty \\ d = a \\ c = \infty. \end{cases}$

6.1 Qualitative Analysis (20 points) :

A 2 input bitonic sorter is made up of 2 gates. When using HIGH to LOW transitions, an AND gate performs the MIN operations and an OR gate performs the MAX operation.

The number of gates associated with an N input sorter can be seen in the following table, and is shown in the chart below:

Inputs	Sorters	Gates
2	1	2
4	6	12
8	24	48
16	80	160

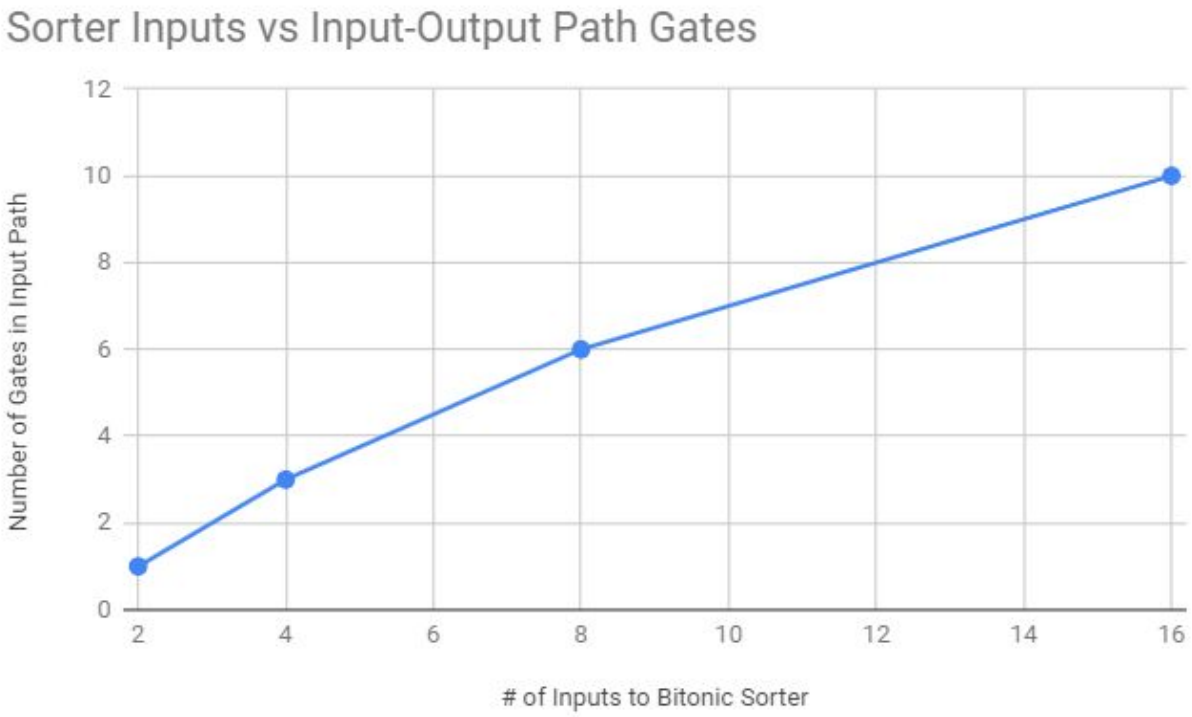


The following equation can be used to determine the total number of gates in an N-input sorter:

Gate Total = (N/2) * log2(N) * [log2(N) + 1]

The number of gates present between an input line and its corresponding output associated with a N input sorter can be seen in the following table, and is shown in the chart below:

Inputs	Input/Output Path Gates
2	1
4	3
8	6
16	10

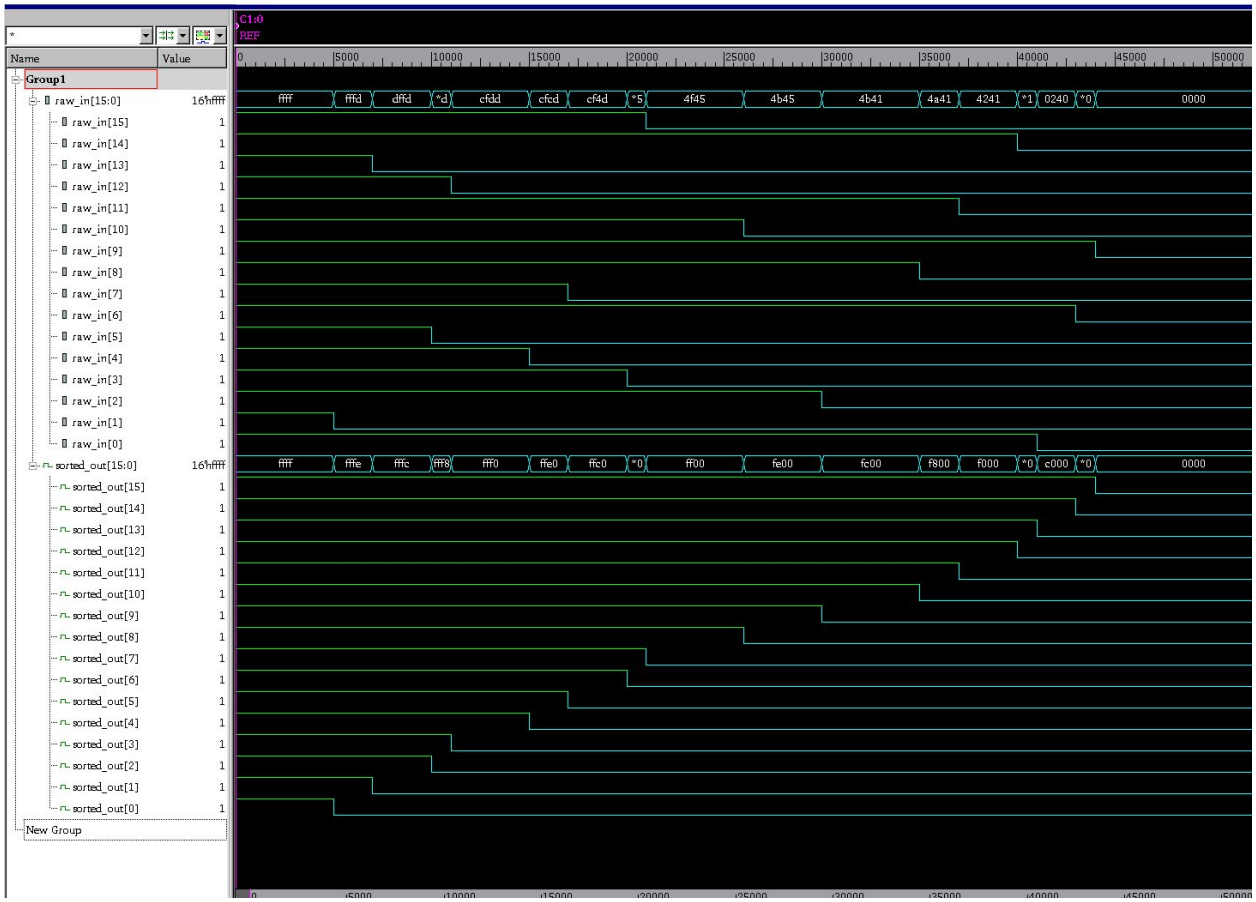


The following equation can be used to determine the number of gates present between an input line and its output for a N-input sorter:

Gates in Path = [(log2(N) - 1)*log2(N)]/2 + log2(N)

6.4 VCS Simulation (10 points) :

Once the 16-input sorter has been implemented, it sorts the input signals in order with earlier transitions being placed closer to the LSB of the bus. As shown in the image below:



7 Simulation Results (20 points) :

The table below shows the simulation results compared to the qualitative values calculated in 6.1

Metric	Qualitative Values	Simulation Values
Area	160 (gates)	212.8 (um^2)
Latency	10 (gate delays)	67 (ns)
Power	160 (gates)	5.5076 (uW)