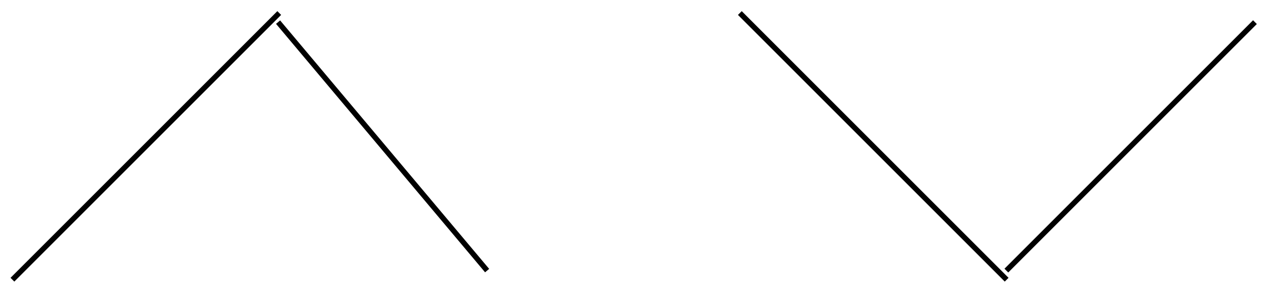
Asynchronous Bitonic Sorter

Algorithm and GasP Circuit Implementation

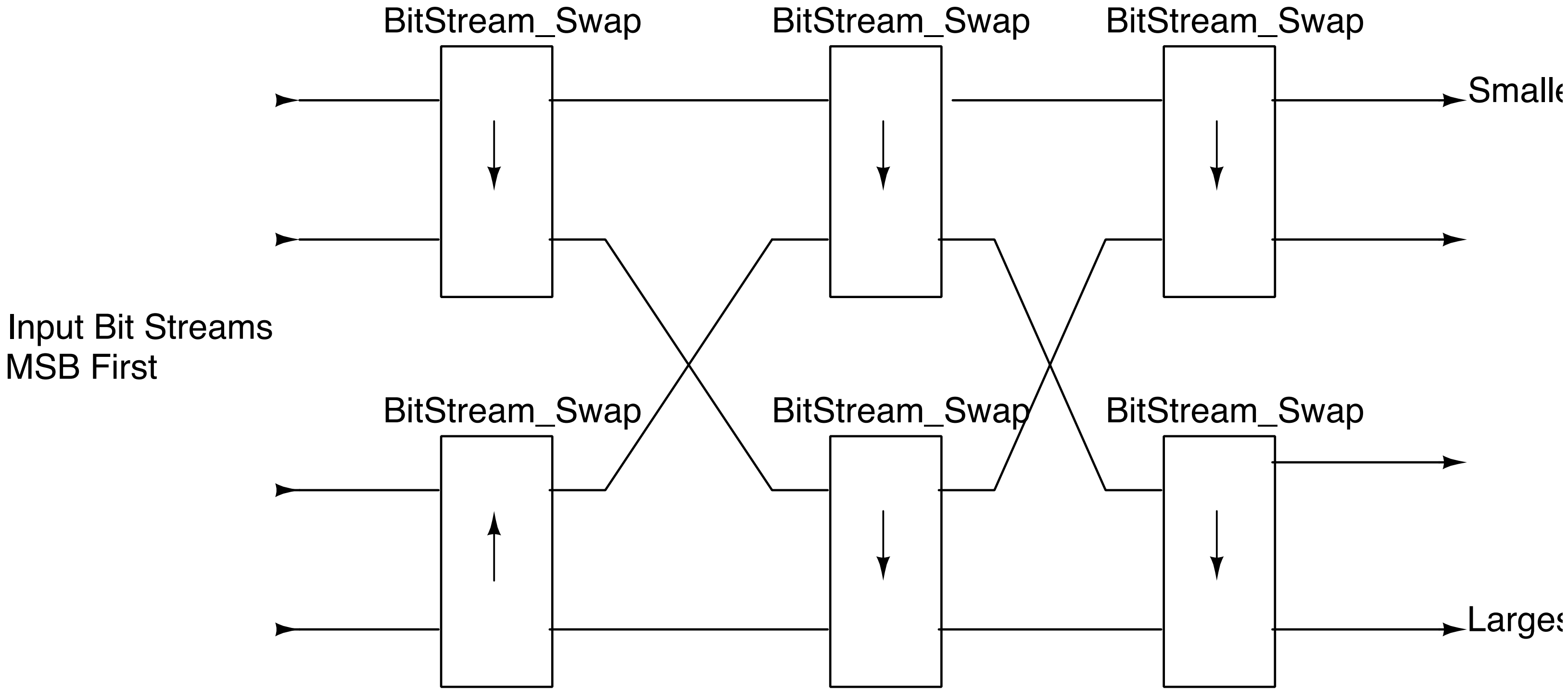
Bitonic Sort

- Bitonic Series
  - x_0 <= x_1 <= … <= x_n <= x_{n+1} <= … x_k
    - such that: 0 <= n <= k



Bitonic Sort Network

Bitonic Sorter - 4 streams X N bits



BitStream_Swap block takes two input bitstream and sorts them with the larger value pointed to by the arrow

Single Slide Summary

Asynchronous Bitonic Sorter

Bitonic Series: any cycle of a monotonic series

Algorithm

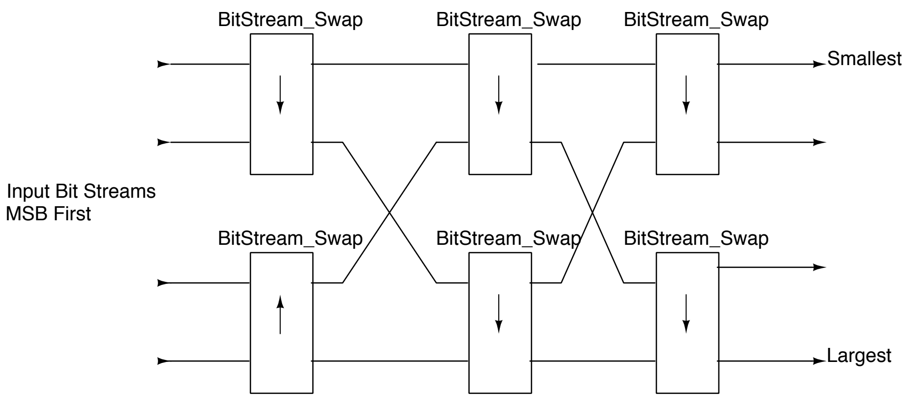
Given two sorted series, the number of comparisons is reduced by an effective binary search for each element in the other opposing series.

Asynchronous Performance

The worst-case performance is if each word is different by the LSBs. In this case, we need to run Word Width # comparisons per Word per stage..

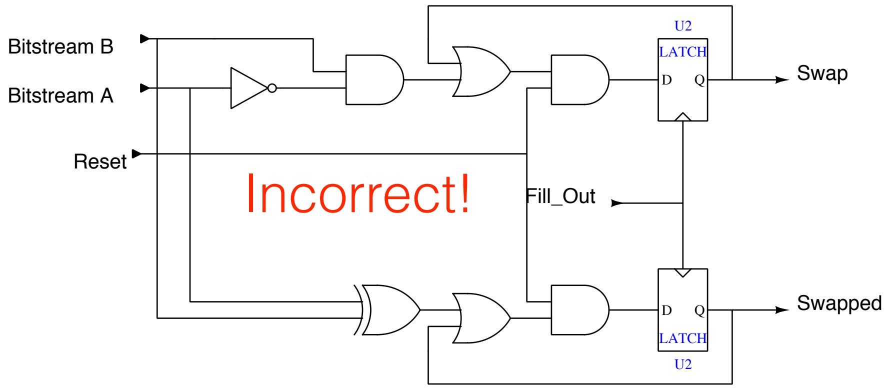
Best case is if the MSBs are different. In this case, a single comparison per word per stage is needed.

Bitonic Sorter - 4 streams X N bits

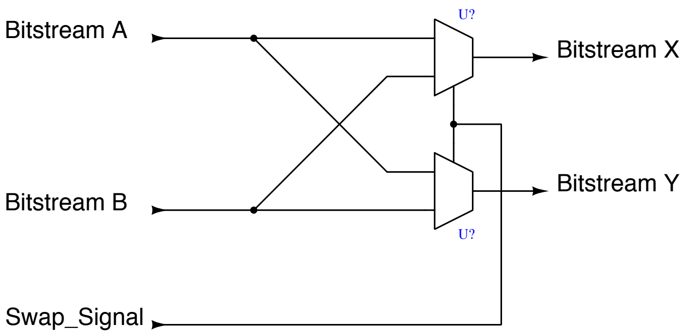


BitStream_Swap block takes two input bitstream and sorts them with the larger value pointed to by the arrow

Current Implementation



Bitstream Swap/Mux Joint



$$Depth = \sum_{i=1}^{log_2 N} i$$

$$\# \ of \ Comparators = (N \div 2) \times \sum_{i=1}^{log_2 N} i$$

Synchronous Performance

The synchronous latency through the network is equal to the depth of stages times the number of comparisons per stage. If we compare all N-bits, we have Word width * N * Depth.

Next Steps

1. Redesign logic to use Swapped signal to stop comparison and trigger next stage
2. Create RSYAMS Joints