Lesson 1-3 MergeSort Notes

MergeSort Animations

Comparator Networks

Sorting network: a fixed network that sorts it inputs using a comparator.
A plus comparator - puts the smaller input on the top output (min(x,y))
A minus comparator puts the larger input on the top output (max(x,y))

A circuit with 3 comparators will have a depth of 3.

Question: Suppose you're only allowed to use plus or minus comparators, is there a way to sort three elements using fewer comparators or that has a shorter critical path?

Sort 4 Values
Question: Based on the circuit shown, eliminate the last two comparators. Will the circuit sort any order of inputs?

Bitonic Sequences

The first step in the sorting circuit results in the first half increasing and the second half decreasing. This is called a BITONIC Sequence.

Do you have a bitonic sequence?

A sequence is bitonic if it goes up, then down.

$(a_0, a_1, \ldots, a_{n-1})$ is bitonic if ….

$a_0 \leq a_1 \leq \ldots \leq a_i$ and $a_{i+1} \geq \ldots \geq a_{n-1}$

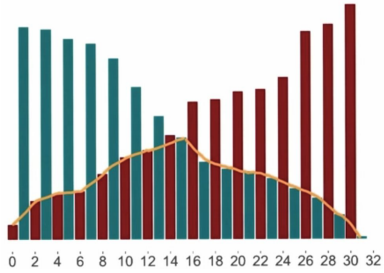
The bitonic sequence is initially non-decreasing, then non-increasing. This condition may hold for the initial sequence or for a circular shift of the sequence.

Bitonic Splits

Once a sequence is bitonic it is easy to sort.

To split:

1. split the sequence into two parts, one that increases and one that decreases.
2. pair the elements in each part. Begin with $(a_0,a_{n/2})$.
3. Now take the minimum of each pair. $\min(a_0,a_{n/2})$, etc.
4. These minimums will form a new bitonic subsequence.



5. Now look at the maximums of each pair. $\max(a_0,a_{n/2})$, etc.
6. The maximums will also form a bitonic subsequence.

This is a bitonic split.

In a bitonic split all elements of the max subsequence are greater than all elements of the min subsequence.

This will lead to a divide and conquer scheme.
The split can be done without extra storage.

So … the result of a bitonic split on a bitonic sequence is two bitonic sequences.

Bitonic Splits: A Parallel Scheme

A bitonic split can be viewed as a DAG of independent comparators.
This will lead to the following parallel scheme:

```
bitonicSplit (A[0:n-1])
  // assume 2|n
  parfor i ← 0 to n/2 - 1 do
    a ← A[i]
    b ← A[i + n/2]
    A[i] ← min(a, b)
    A[i + n/2] ← max(a, b)
```

A subtle point: the fixed size circuit has a constant depth or span.

Bitonic Merge

Given a bitonic sequence, if you perform the maximum number of bitonic splits on it, you will achieve a bitonic merge…. meaning the bitonic sequence is now sorted in order.

The bitonic merge pseudocode:

```
bitonicMerge (A[0:n-1])
    //assume A is bitonic
    if n ≥ 2 then
        //assume 2|n
        bitonicSplit (A[:])
spawn   bitonicMerge (A[0: n/2 - 1])
        bitonicMerge (A[n/2 : n-1])
```

Bitonic Merge Network

A bitonic merge is a sequence of splits. A sequence of splits is a set of min/max pairs. Compare the circuit with the pseudo code.

Generate a Bitonic Sequence

To create a more generic bitonic sequence network:

1. Use plus and minus comparators
2. The first set of comparators is creating a series of up-down pairs.
3. This will make the first four elements of an eight item sequence bitonic.
4. It will also make the second four elements bitonic.
5. Now - turn one into an increasing subsequence and the other into a decreasing subsequence.

In summary- to create a bitonic sequence:

1. start with an arbitrary input.
2. run plus/minus bitonic merges of size 2.
3. run plus/minus bitonic merges of size 4.
4. continue until done.

Conclusion:

The bitonic merge has a fixed regular structure that lends itself to a natural implementation with a programmable gate array, etc.

It also means it will map well to fixed data parallel hardware like SIMD, etc.

The downside is …. it is NOT work optimal. So trade-offs will have to be made.