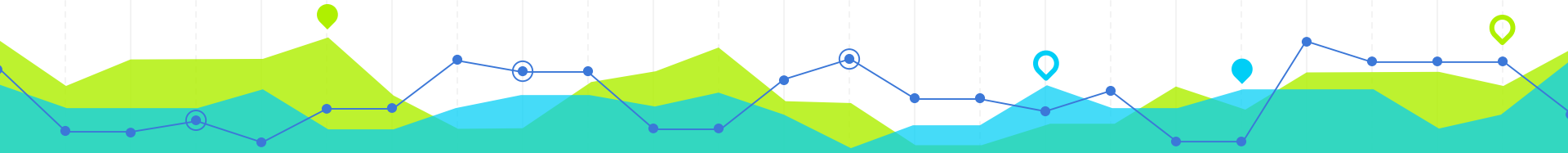
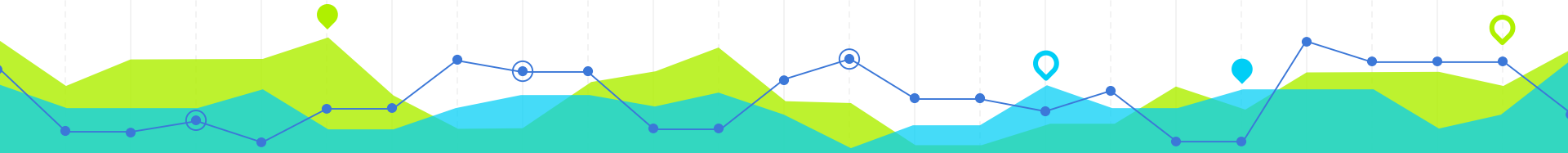


FAST PARALLEL SORTING ALGORITHMS ON GPUS

Introduction



Odd-Even Sort

1

Odd-Even Sort

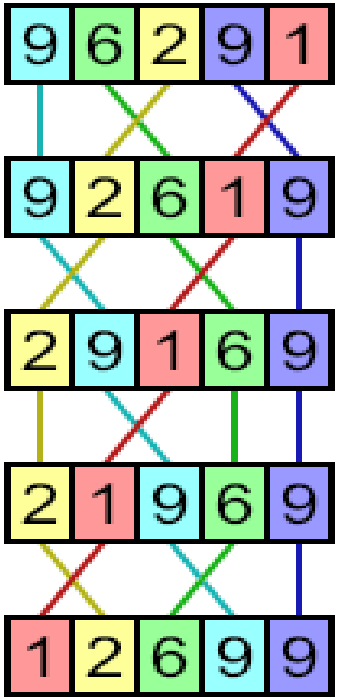
- The odd-even sort is a parallel sorting algorithm and is based on bubble-sort technique.
- Adjacent pairs of items in an array are exchanged if they are found to be out of order.
- Total running time fort this technique is O(log 2N).

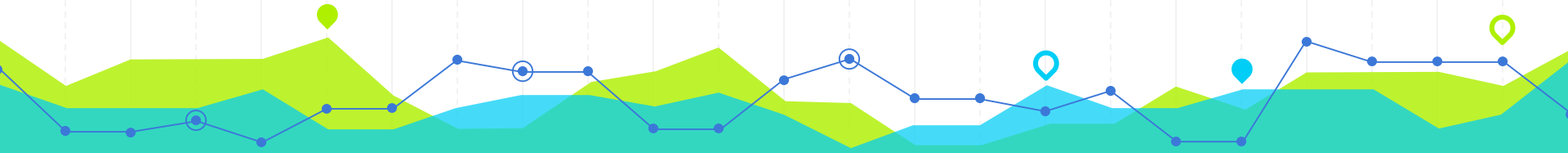
Odd-Even Sort Algorithm

Algorithm 1 Odd Even Sort

for $k = 1 \rightarrow N/2$ **do**
  do parallel
  if $i > i+1 \; \forall \; i\%2 \; != 0$ **then**
    swap $i, i+1$
  end if
  end parallel
  do parallel
  if $i > i+1 \; \forall \; i\%2 == 0$ **then**
    swap $i, i+1$
  end if
  end parallel
end for

Odd-Even Sort Algorithm





Bitonic Sort

3

Sorting Networks: Bitonic Sort

- A bitonic sorting network sorts n elements in $\Theta(\log^2 n)$ time.
- A bitonic sequence has two tones - increasing and decreasing, or vice versa. Any cyclic rotation of such networks is also considered bitonic.
- $\langle 1,2,4,7,6,0 \rangle$ is a bitonic sequence, because it first increases and then decreases. $\langle 8,9,2,1,0,4 \rangle$ is another bitonic sequence.
- The kernel of the network is the rearrangement of a bitonic sequence into a sorted sequence.

Sorting Networks: Bitonic Sort

- Let $s = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be a bitonic sequence such that $a_0 \leq a_1 \leq \cdots \leq a_{n/2-1}$ and $a_{n/2} \geq a_{n/2+1} \geq \cdots \geq a_{n-1}$.

- Consider the following subsequences of s:

$$s_1 = \langle \min\{a_0, a_{n/2}\}, \min\{a_1, a_{n/2+1}\}, \ldots, \min\{a_{n/2-1}, a_{n-1}\} \rangle$$
$$s_2 = \langle \max\{a_0, a_{n/2}\}, \max\{a_1, a_{n/2+1}\}, \ldots, \max\{a_{n/2-1}, a_{n-1}\} \rangle \quad (1)$$

  Note that s_1 and s_2 are both bitonic and each element of s_1 is less than every element in s_2.

- We can apply the procedure recursively on s_1 and s_2 to get the sorted sequence.

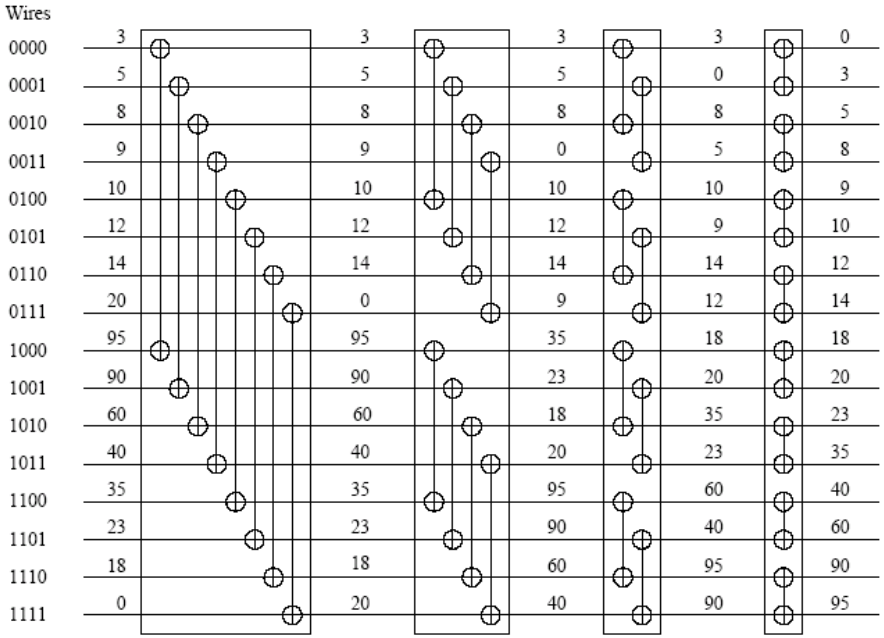
Original sequence	3	5	8	9	10	12	14	20	95	90	60	40	35	23	18	0
1st Split	3	5	8	9	10	12	14	0	95	90	60	40	35	23	18	20
2nd Split	3	5	8	0	10	12	14	9	35	23	18	20	95	90	60	40
3rd Split	3	0	8	5	10	9	14	12	18	20	35	23	60	40	95	90
4th Split	0	3	5	8	9	10	12	14	18	20	23	35	40	60	90	95

Merging a 16-element bitonic sequence through a series of log 16 bitonic splits.

Sorting Networks: Bitonic Sort

- We can easily build a sorting network to implement this bitonic merge algorithm.
- Such a network is called a *bitonic merging network*.
- The network contains $\log n$ columns. Each column contains $n/2$ comparators and performs one step of the bitonic merge.
- We denote a bitonic merging network with n inputs by $\oplus\mathrm{BM}[n]$.
- Replacing the $\oplus$ comparators by $\ominus$ comparators results in a decreasing output sequence; such a network is denoted by $\ominus\mathrm{BM}[n]$.

Sorting Networks: Bitonic Sort



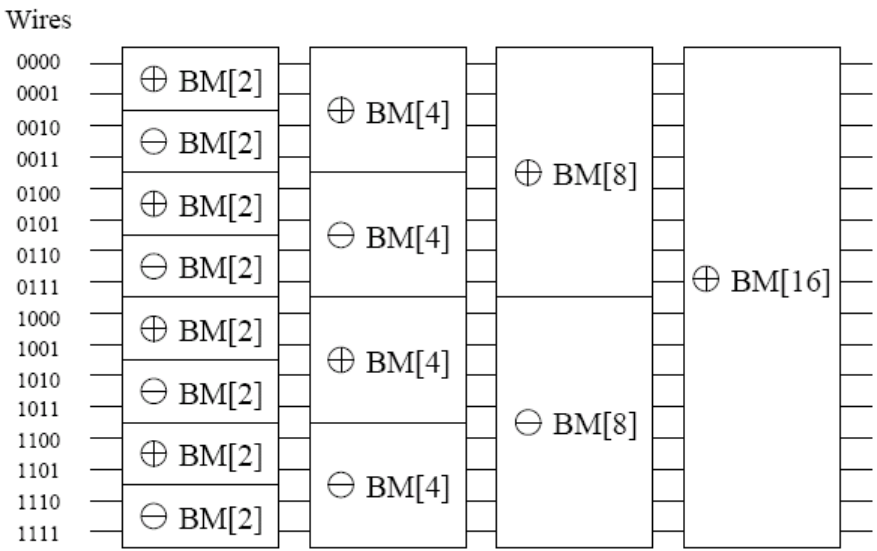
A bitonic merging network for $n = 16$. The entire figure represents a $\oplus$BM[16] bitonic merging network. The network takes a bitonic sequence and outputs it in sorted order.

Sorting Networks: Bitonic Sort

How do we sort an unsorted sequence using a bitonic merge?

We must first build a single bitonic sequence from the given sequence.

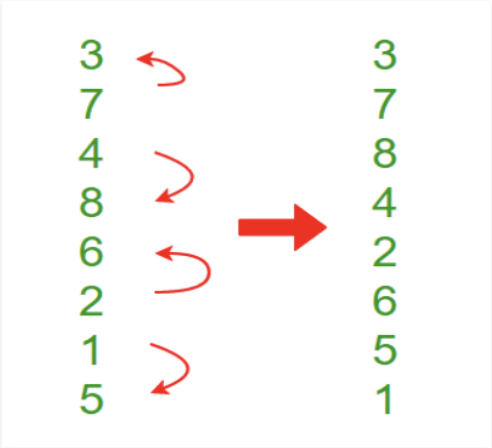
- A sequence of length 2 is a bitonic sequence.
- A bitonic sequence of length 4 can be built by sorting the first two elements using $\oplus BM[2]$ and next two, using $\ominus BM[2]$.
- This process can be repeated to generate larger bitonic sequences.



A schematic representation of a network that converts an input sequence into a bitonic sequence. In this example, ⊕BM[k] and ⊖BM[k] denote bitonic merging networks of input size k that use ⊕ and ⊖ comparators, respectively. The last merging network (⊕BM[16]) sorts the input. In this example, $n = 16$.

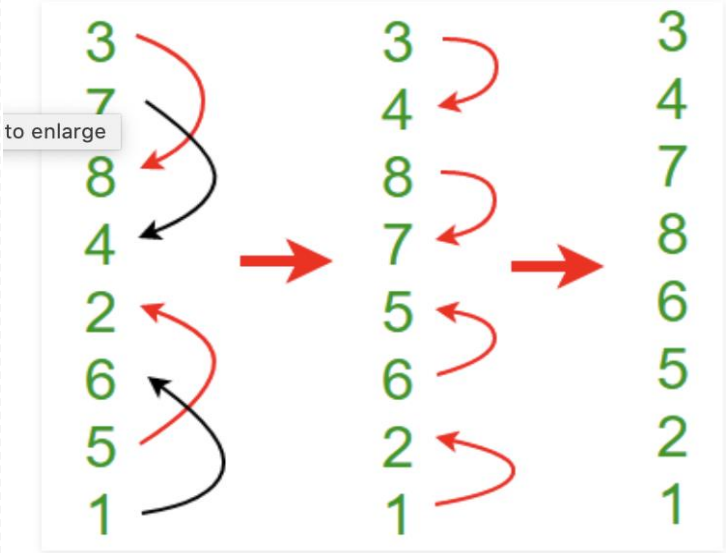
Bitonic Sequence

- Step #1

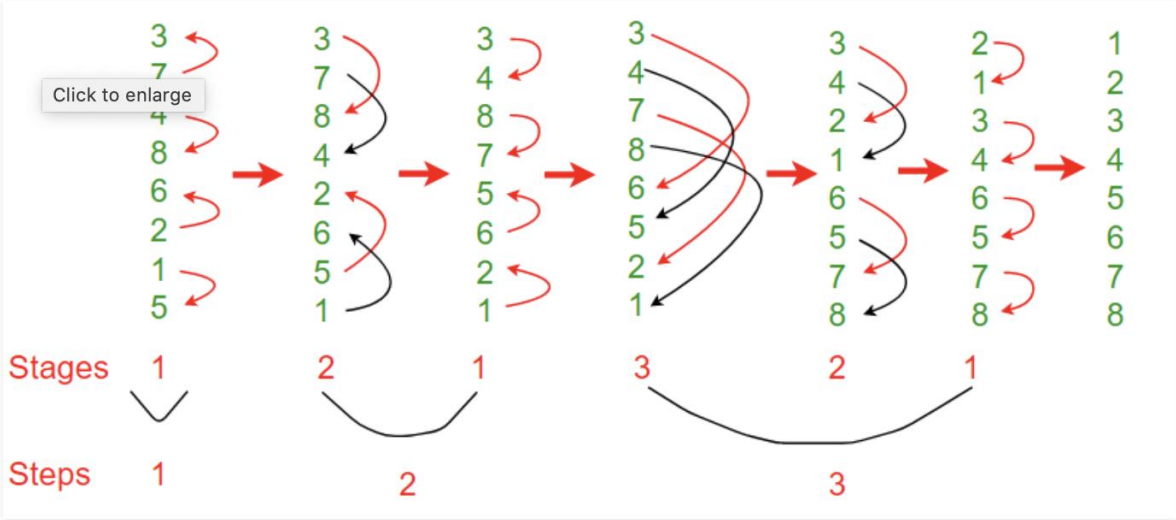


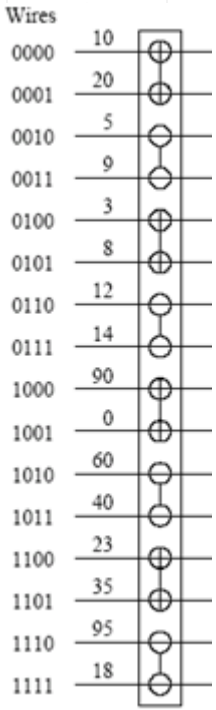
Bitonic Sequence

- Step #2

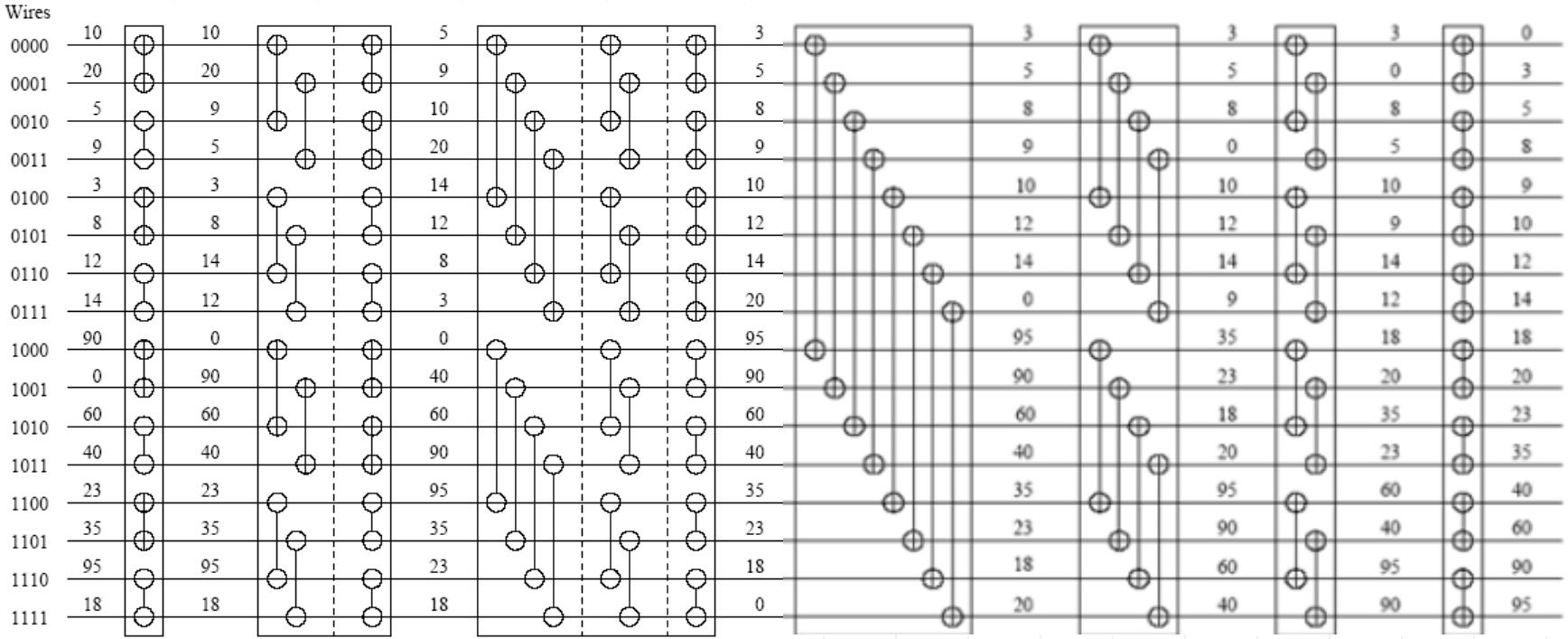


Bitonic Sequence




Wires
0000
0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111
10
20
5
9
3
8
12
14
90
0
60
40
23
35
95
18

Sorting Networks: Bitonic Sort



Bitonic Sort

- The sequence of comparisons is data-independent makes it one of the fastest and suitable parallel sorting algorithms.
- In the first step it makes the arbitrary sequence in to bitonic sequence.
- In the second step the bitonic sequence is sorted.

Conclusion

- Sorting is one of the most fundamental problems in computer science, as it is used in most software applications.
- This presentation shows differrent parallel sorting algorithms on GPUs.
- The performance is affected mainly by : nature of algorithm and hardware platforms.