Parallel Computing with GPUs

Sorting and Libraries
Part 1 – Sorting

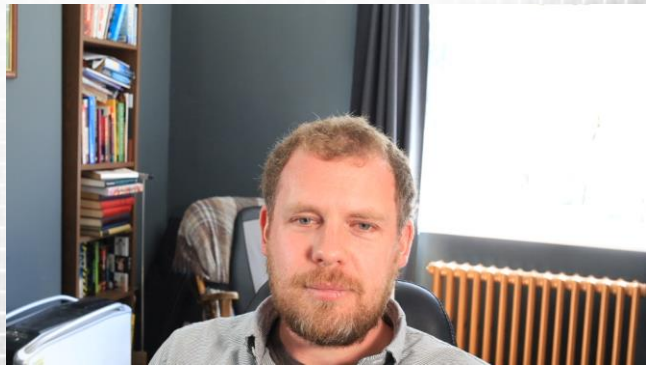
The University Of Sheffield.

Dr Paul Richmond

http://paulrichmond.shef.ac.uk/teaching/COM4521/

This Lecture (learning objectives)

- Sorting Networks
  - Demonstrate the use of a sorting network to achieve parallel sorting
  - Compare sorting networks with serial sorting approaches
- Merge and Bitonic Sort
  - Present the merge sort and considers its performance implications
  - Identify performacne features of bitonic sorting

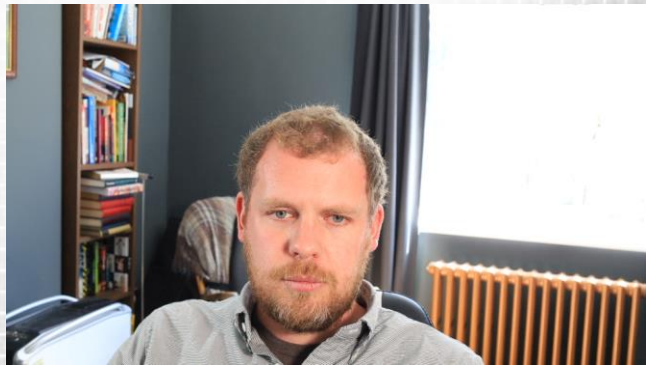


Serial Sorting Examples

- Insertion Sort
  - Insert a new element into a sorted list.
    - E.g. [1 6 3 4 2 5]
      - [1] -> [1 6] -> [1 3 6] -> [1 3 4 6] -> [1 2 3 4 6] -> [1 2 3 4 5 6]
- Bubble Sort
  - Exchange and Sweep to compare each pair of adjacent elements
  - *O(n²)* worst-case and average case, *O(n)* best case.
    - E.g. [1 6 3 4 2 5]
      - [1 6 3 4 2 5] -> [1 **3 6** 4 2 5] -> [1 3 **4 6** 2 5] -> [1 3 4 **2 6** 5] -> [1 3 4 2 **5 6**]
      - [1 3 **2 4** 5 6]
      - [1 **2 3** 4 5 6]

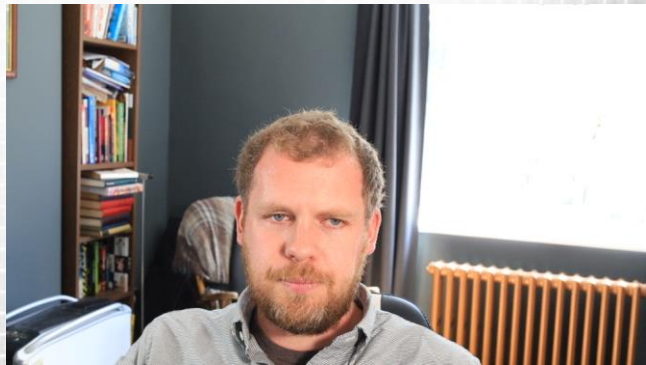
Serial Sorting Examples

- Insertion Sort
  - Insert a new element into a sorted list.
    - E.g. [1 6 3 4 2 5]
      - [1] -> [1 6] -> [1 3 6] -> [1 3 4 6] -> [1 2 3 4 6] -> [1 2 3 4 5 6]
- Bubble Sort
  - Exchange and Sweep to compare each pair of adjacent elements
  - *O(n²)* worst-case and average case, *O(n)* best case.
    - E.g. [1 6 3 4 2 5]
      - [1 6 3 4 2 5] -> [1 **3 6** 4 2 5] -> [1 3 **4 6** 2 5] -> [1 3 4 **2 6** 5] -> [1 3 4 2 **5 6**]
      - [1 3 **2 4** 5 6]
      - [1 **2 3** 4 5 6]



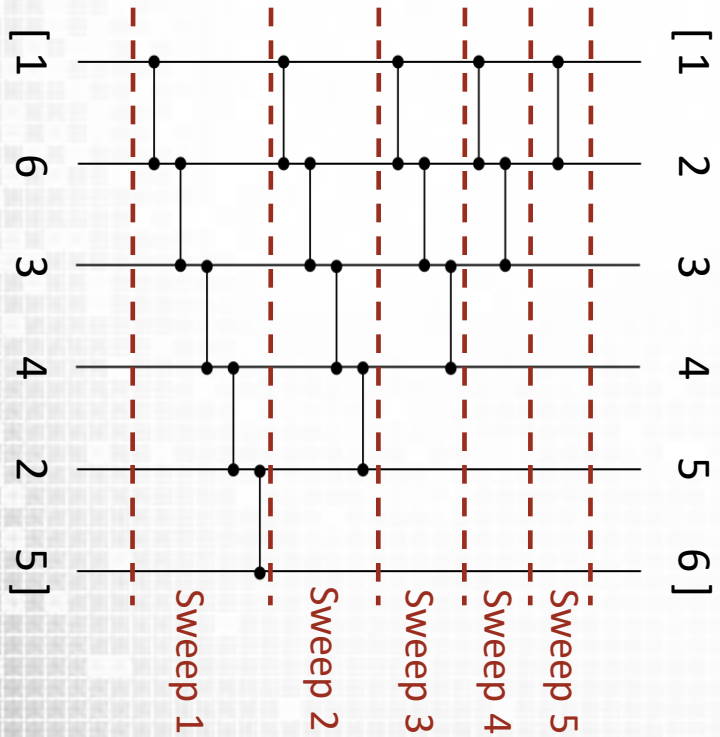
Classifying Sort Techniques/Implementations

- Data driven
  - Each step of the algorithm depends on the previous step version
  - Highly serial
- Data independent
  - The algorithms performs fixed steps and does not change its processing based on data
  - Well suited to parallel implementations
  - Can be expressed as a sorting network…



Sorting Networks

- A sorting network is a comparator network that sorts all input sequences
  - Following the same execution of stages
- Consider the previous Bubble Sort [1 6 3 4 2 5]



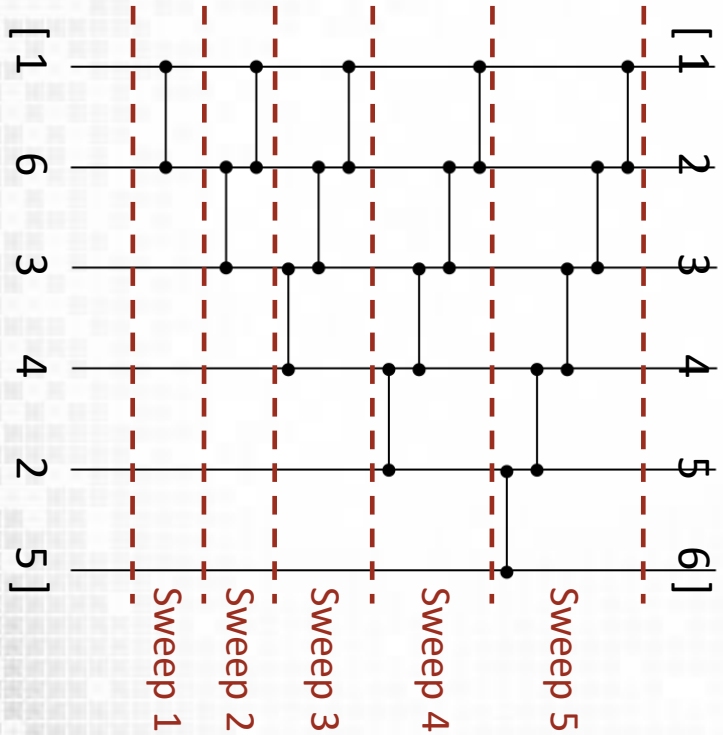
[1 6 3 4 2 5] -> [1 **3 6** 4 2 5] -> [1 3 **4 6** 2 5] -> [1 3 4 **2 6** 5] -> [1 3 4 2 **5 6**]
[1 3 4 2 5 6] -> [1 3 4 2 5 6] -> [1 3 **2 4** 5 6] -> [1 3 2 4 5 6]
[1 3 2 4 5 6] -> [1 2 3 4 5 6] -> [1 2 3 4 5 6]
[1 3 2 4 5 6] -> [1 **2 3** 4 5 6]
[1 2 3 4 5 6]

Sweeps

Not considered
Compared not swapped
Compared and swapped

Sorting Networks

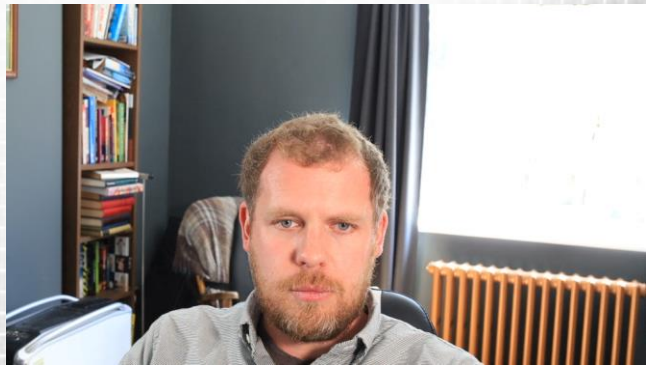
- And Insertion Sort…



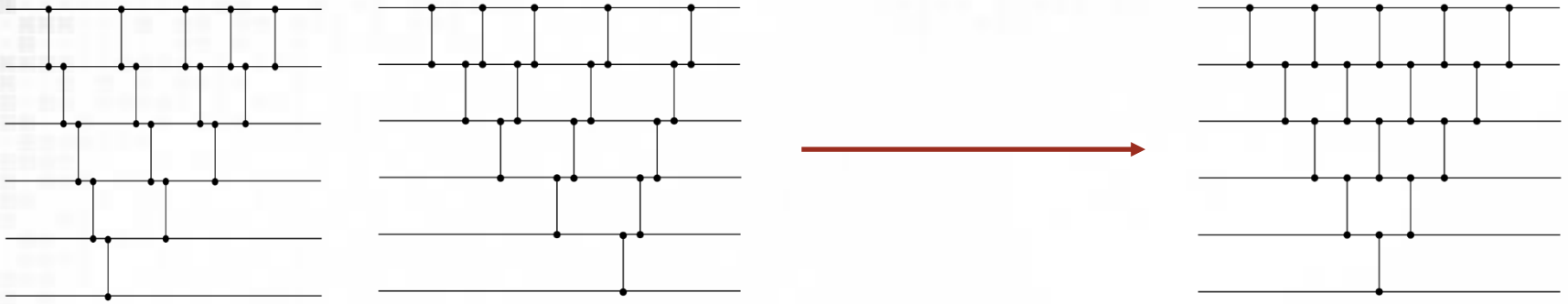
[**1 6** 3 4 2 5]
[1 **3 6** 4 2 5] -> [1 3 6 4 2 5]
[1 3 **4 6** 2 5] -> [1 3 4 6 2 5] -> [1 3 4 6 2 5]
[1 3 4 **2 6** 5] -> [1 3 **2 4** 6 5] -> [1 **2 3** 4 6 5] -> [1 2 3 4 6 5]
[1 2 3 4 **5 6**] -> [1 2 3 4 5 6] -> [1 2 3 4 5 6] -> [1 2 3 4 5 6] -> [1 2 3 4 5 6]

Sweeps

Not considered
Compared not swapped
Compared and swapped



Parallel Sorting Networks



Bubble Insertion

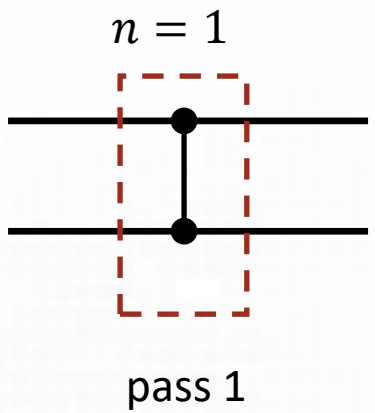
[1 6 3 4 2 5]
[1 3 6 4 2 5]
[1 3 4 6 2 5]
[1 3 4 2 6 5]
[1 3 2 4 5 6]
[1 2 3 4 5 6]
[1 2 3 4 5 6]
[1 2 3 4 5 6]
[1 2 3 4 5 6]

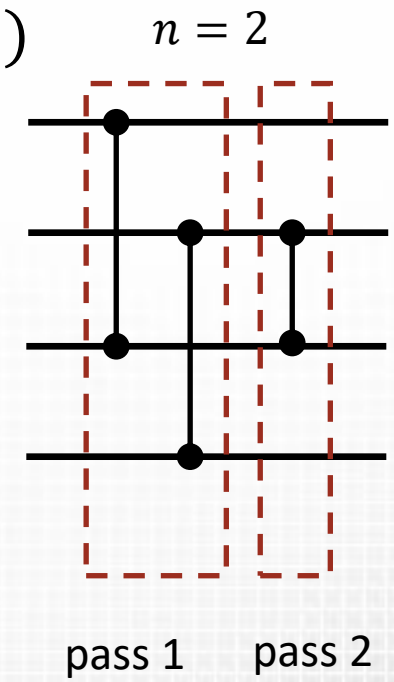
Sweeps = 9

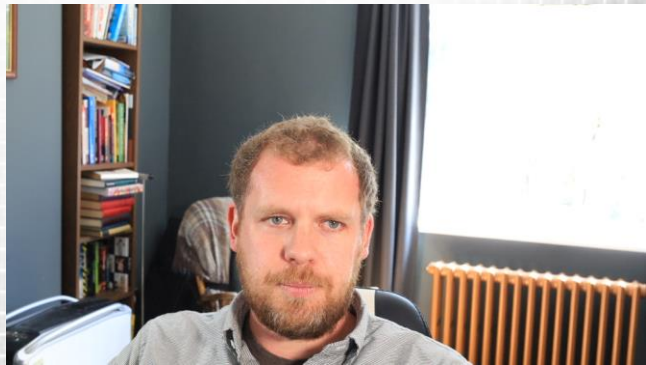
- Parallel Bubble and Insertion sorting network is still not very efficient
  - $2n - 3$ sweeps
  - $n(n-1)/2$ comparisons - $O(n^2)$ complexity

Merge Sort

- To reduce the $O(n^2)$ overhead we need a better sorting network
- The odd-even merge sort network (for power of 2 n)
  - Sort all odd and even keys separately and then merge m values of a stage
  - Merge a sorted sequence of elements on lines $< a_1, \ldots, an >$ with those on lines $< a_{n+1}, \ldots, a_{2n} >$
  - Each merge requires $\log(n)$ passes
  - Total complexity of $O(n \log(n^2) + \log(n))$



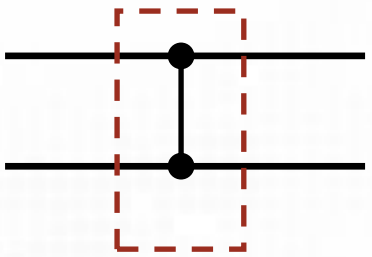




Merge Sort

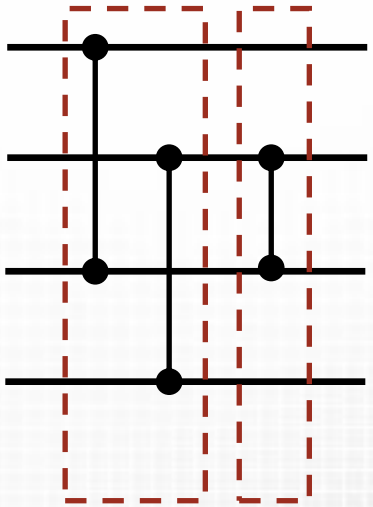
- To reduce the $O(n^2)$ overhead we need a better sorting network
- The odd-even merge sort network (for power of 2 n)
  - Sort all odd and even keys separately and then merge m values of a stage
  - Merge a sorted sequence of elements on lines $< a_1, \ldots, an >$ with those on lines $< a_{n+1}, \ldots, a_{2n} >$
  - Each merge requires $\log(n)$ passes
  - Total complexity of $O(n \log(n^2) + \log(n))$

$n = 1$

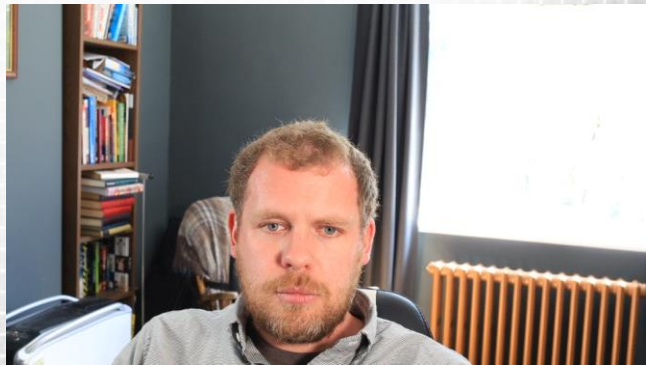


pass 1

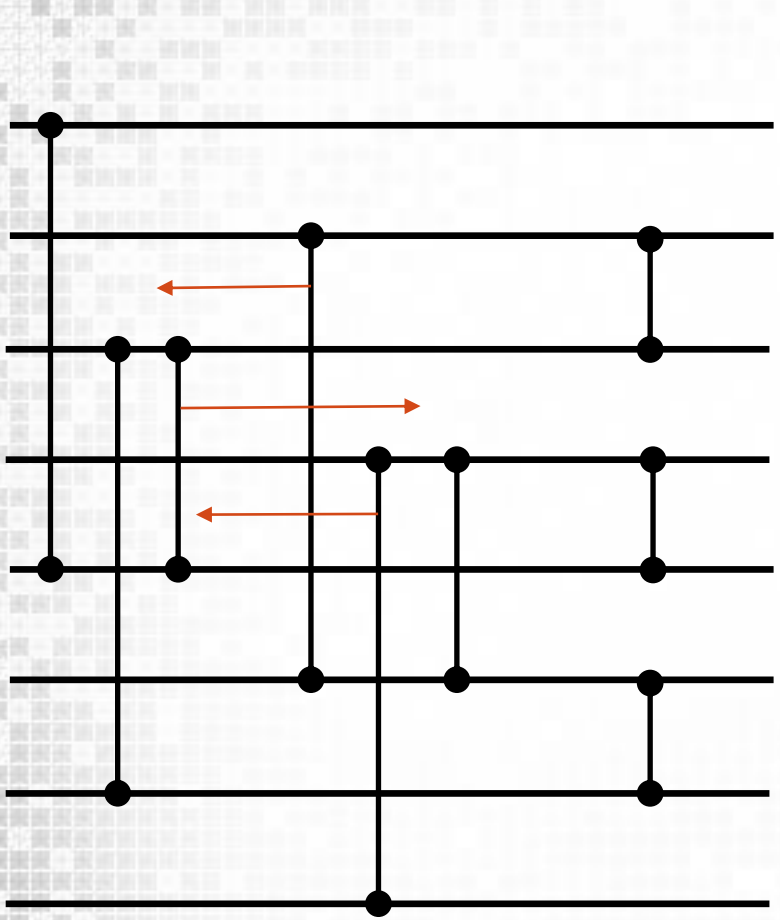
$n = 2$

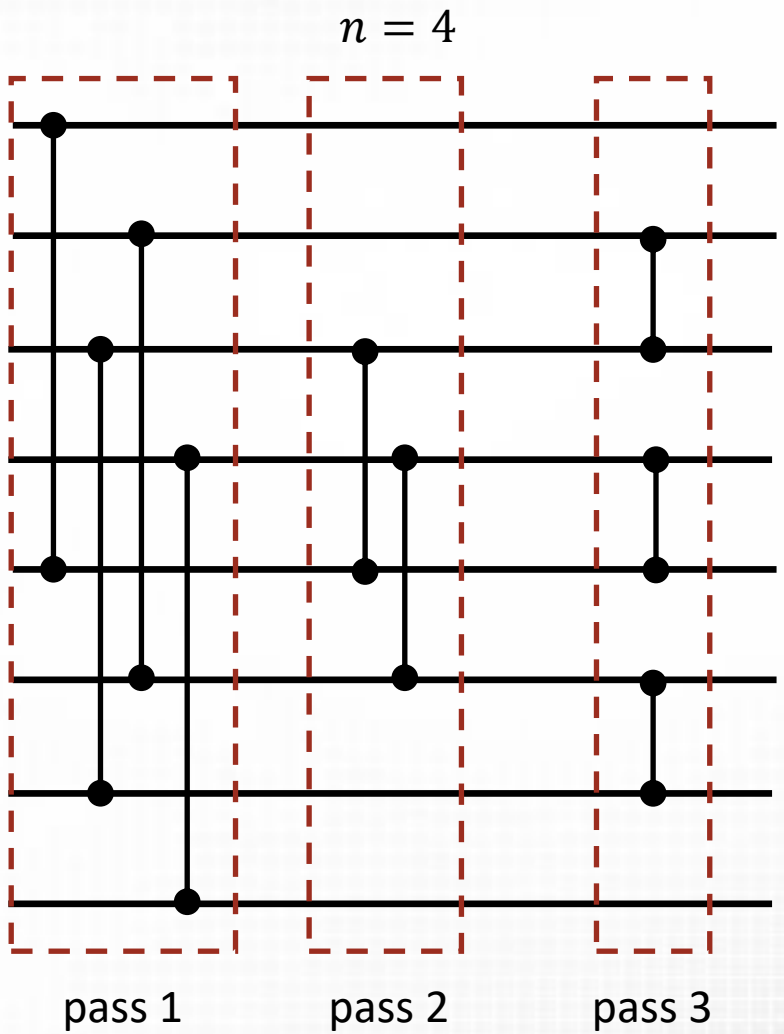


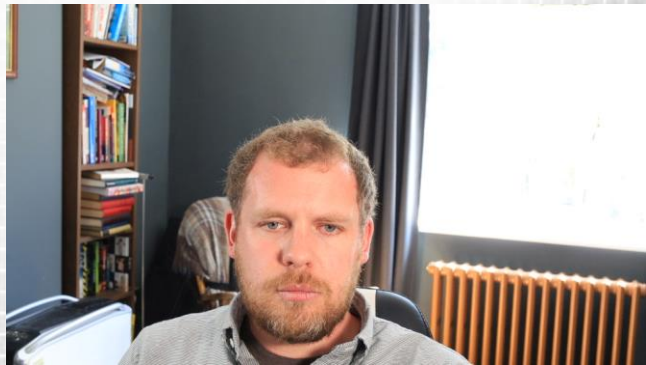
pass 1 pass 2



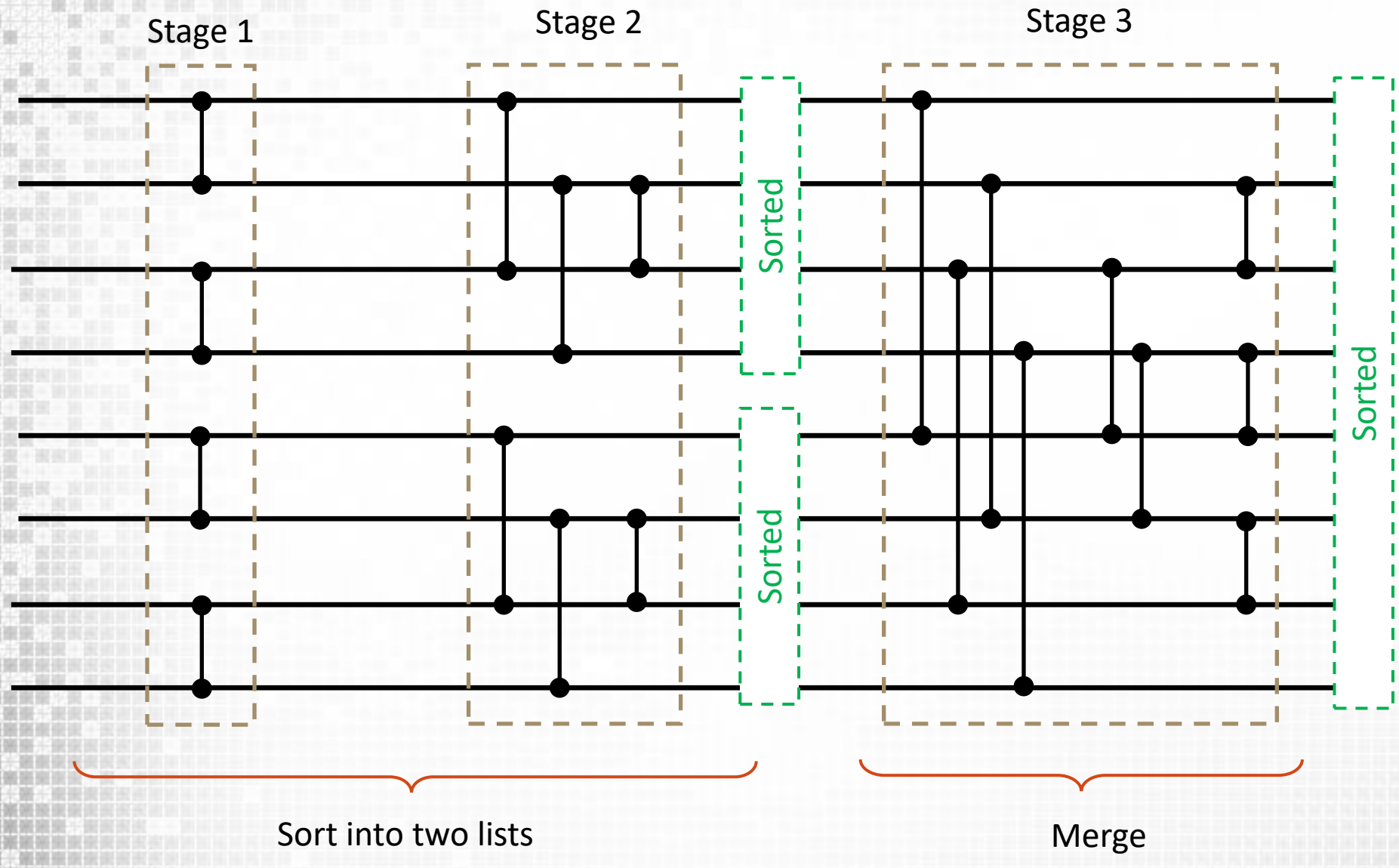
Merging of two sorted sequences (n=4)

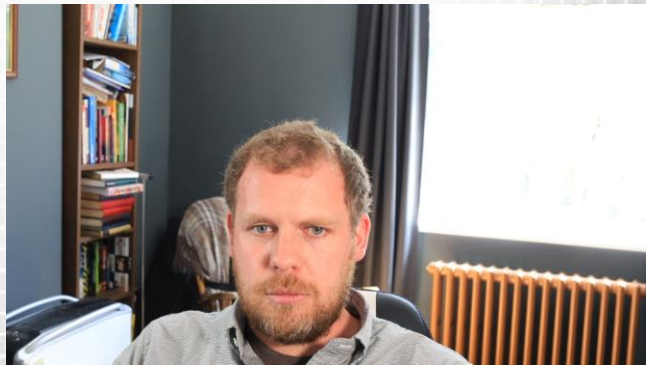




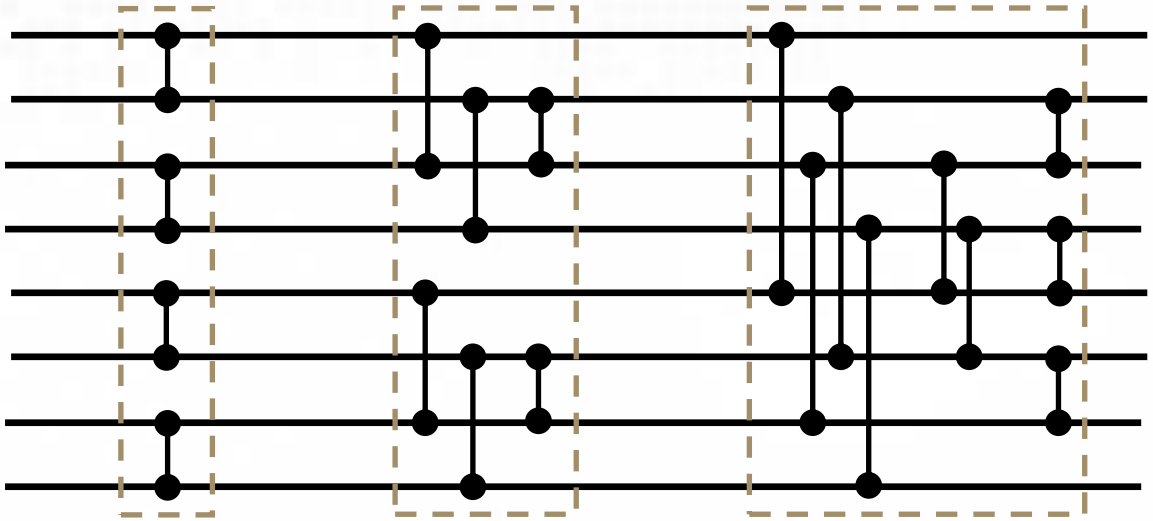


Full Merge Sorting (n=4)





Full Merge Sorting (n=4) example

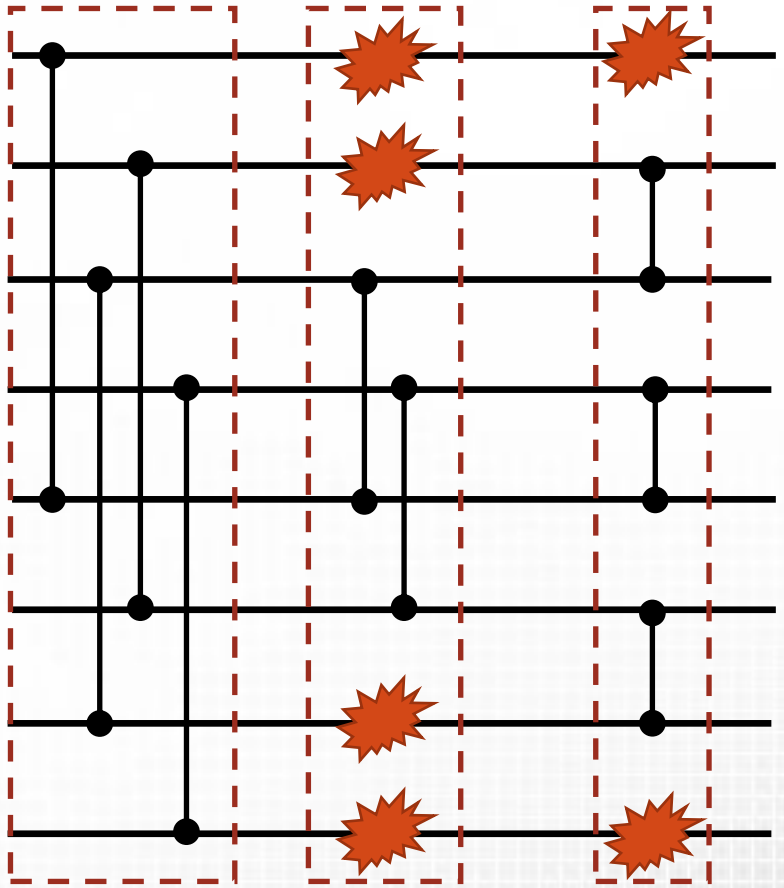


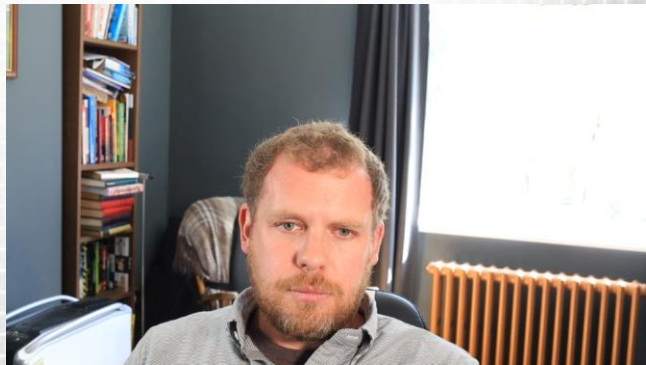
Input	Stage 1	Stage 2			Stage 3							Output
8	1	1			1							1
1	8		5	3			3				2	2
5	3	3		5		5			2		3	3
3	5		8					8		4	4	4
6	2	2			2				5		5	5
2	6		6	4			4			8	6	6
4	4	4		6		6					8	8
9	9		9					9				9

The University Of Sheffield.

Limitations of Merge Sort

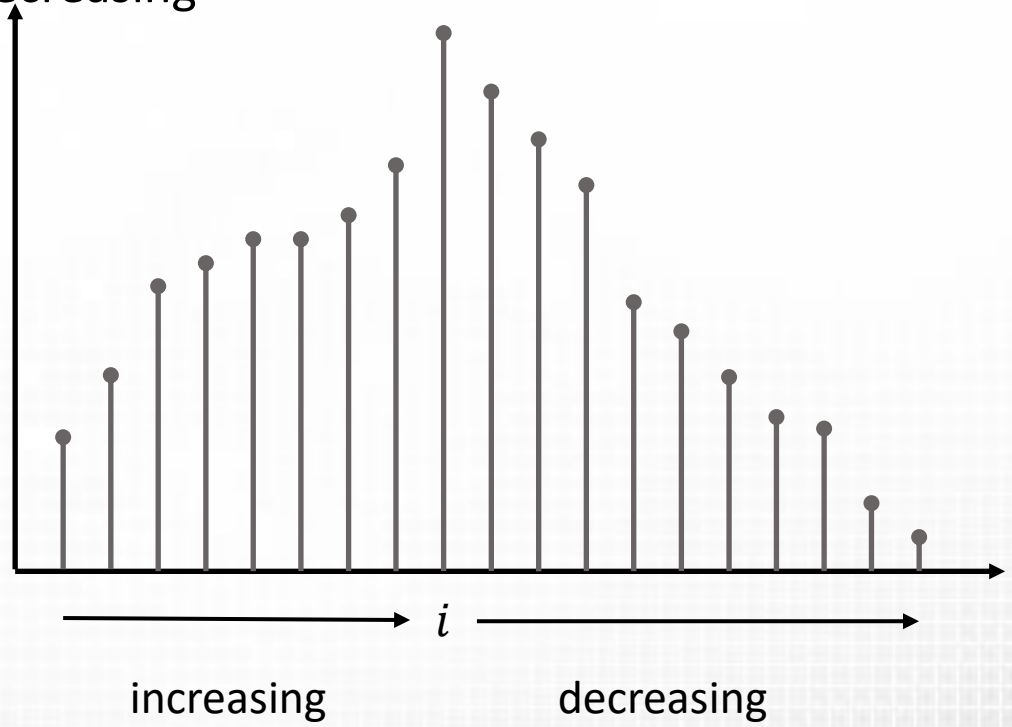
- What is potentially wrong with a merge sort GPU implementation?
  - Irregular memory accesses
  - Not all values are compared in each pass (uneven workload per thread)

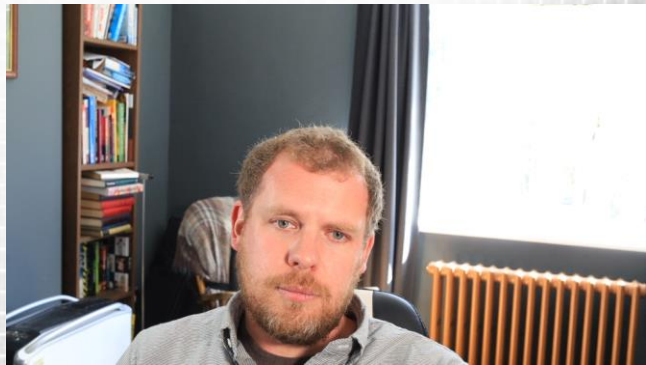




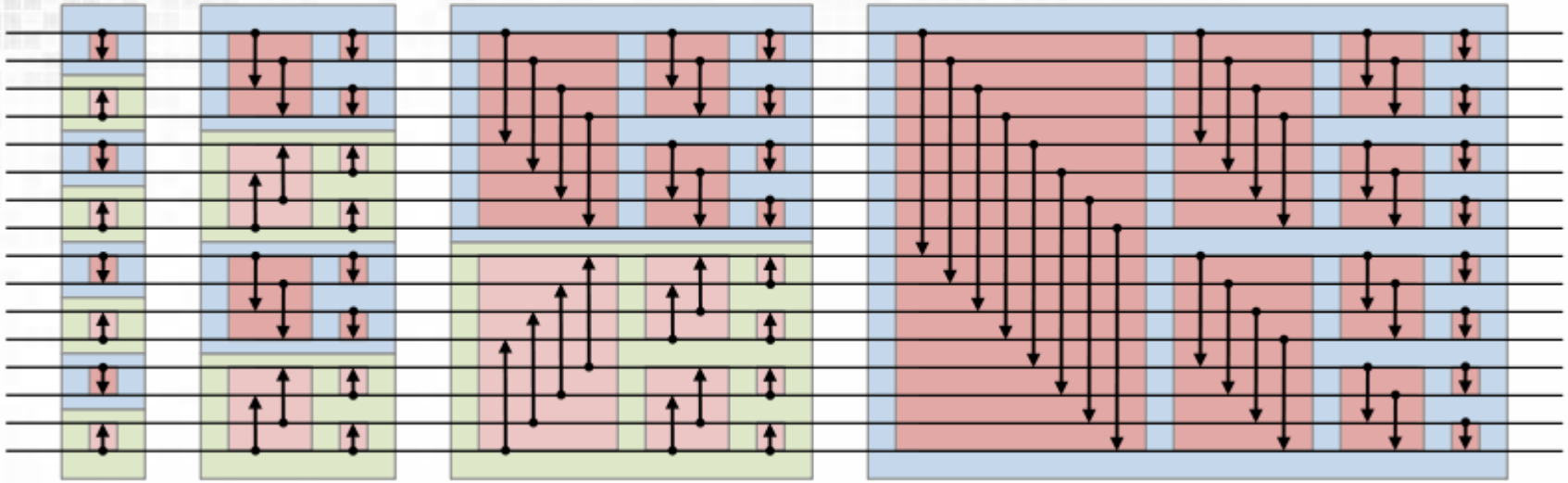
Solution: Bitonic Sort

- Bitonic sorting network
  - Iterative splitting and merging of inputs into increasing large bionic sequences
  - A sequence is bitonic if
    - There is an i, such that , $a_0 \ \ldots, a_i$ is monotonically increasing and $a_i \ \ldots, a_n$ is monotonically decreasing

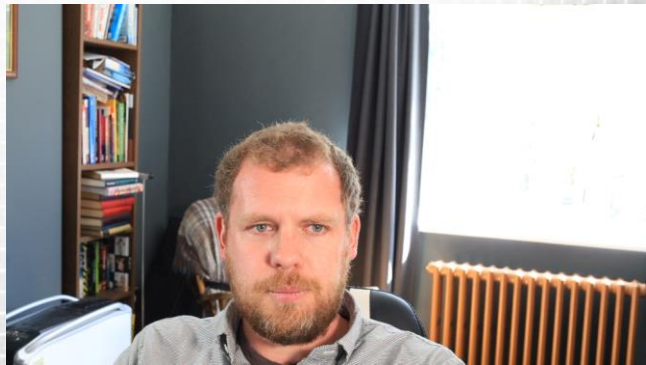




Bitonic Sorting Network



- Sorting and Merging increasing large bionic sequences
  - When $n = 2^k$ there are k levels with $\frac{n}{2}$ comparisons each
- GPU Implementation
  - Regular access strides :-)
  - Efficiently balanced workload :-)
  - Requires multiple kernel launches to merge over n > block size



Summary

- Sorting Networks
  - Demonstrate the use of a sorting network to achieve parallel sorting
  - Compare sorting networks with serial sorting approaches
- Merge and Bitonic Sort
  - Present the merge sort and considers its performance implications
  - Identify performance features of bitonic sorting

- Next: Libraries and Thrust