MANCHESTER 1824
The University of Manchester

High-Throughput Sorting by Dynamically Merging Multiple Hardware Sequential Sorters

Wei Song

03/04/2014

Advanced Processor Technologies Group
The School of Computer Science

Motivation

- Hardware sorter is important.
- Parallel sorters have size limit.
  - Sorting N numbers need a network sized $N\log^2(N)$
- Sequential sorters have throughput limit.
  - Sorting throughput is limited to 1 number per cycle.
- Is there a way to sort N (N>1M) numbers with a throughput larger than 1 number per cycle?

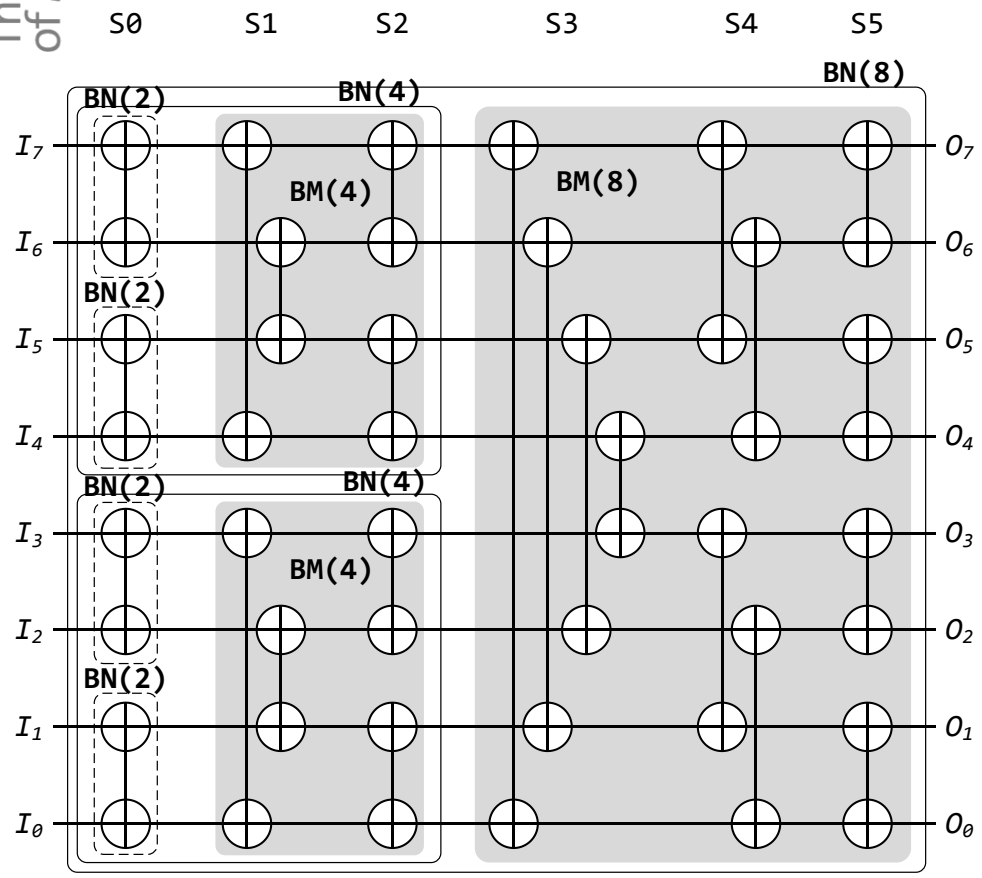
Content

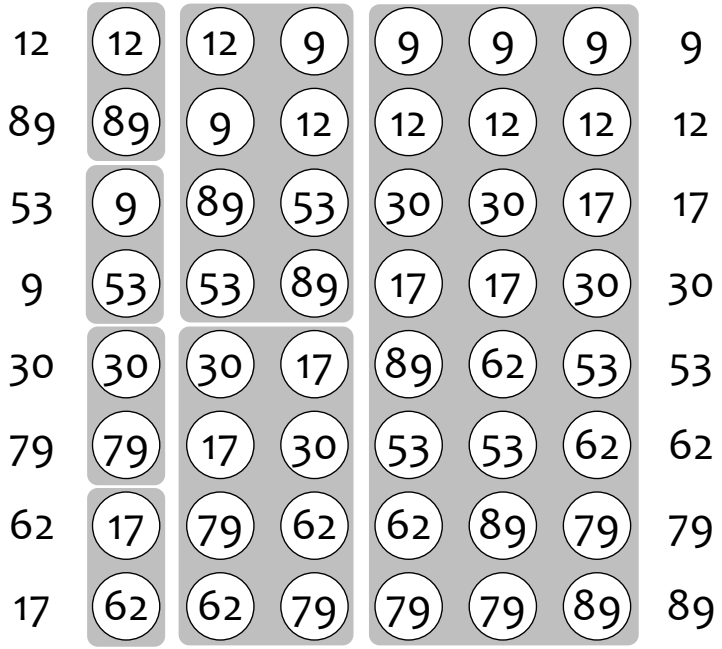
- Review of existing sorters
  - Parallel sorters
  - Sequential sorters
- Parallel merge-tree sorter
  - Key ideas
  - Hardware structure
  - Performance

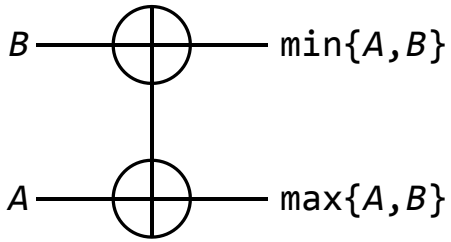
MANCHESTER 1824

The University of Manchester

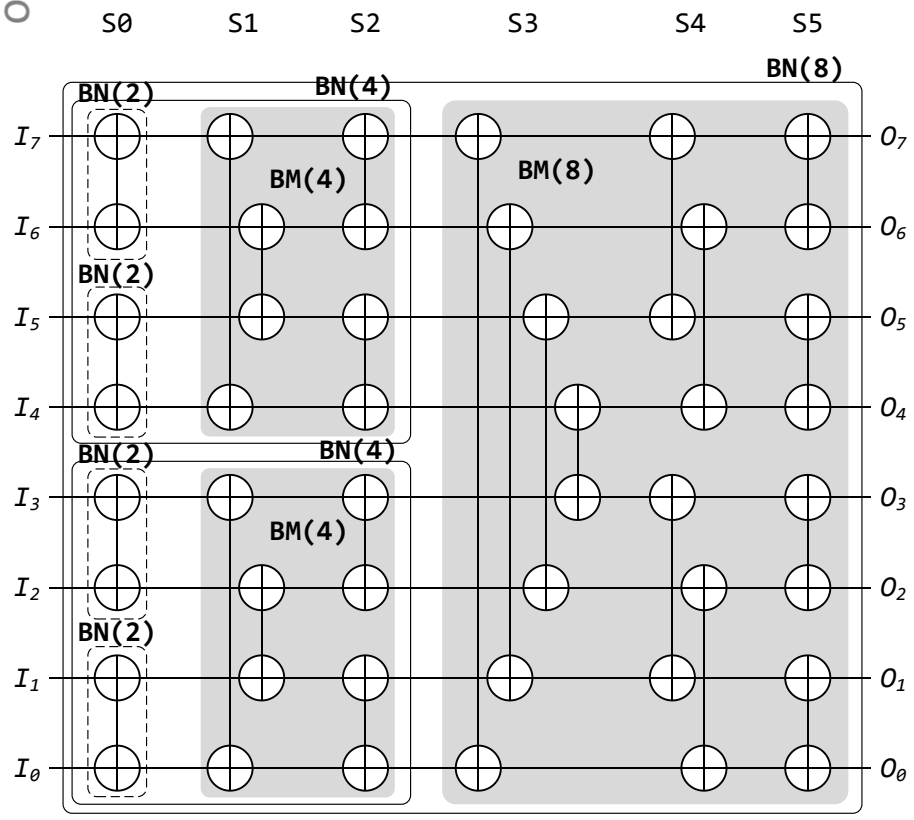
Parallel Sorters (Bitonic Sorting Network)

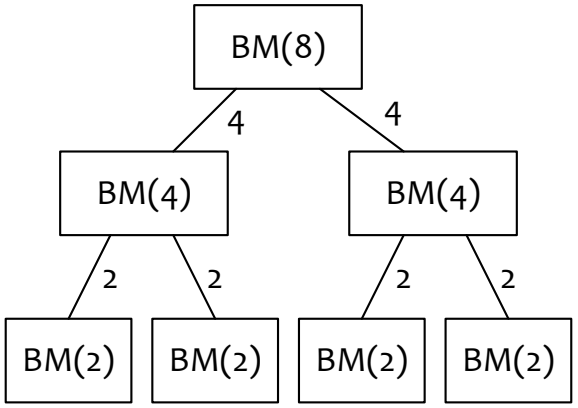






Parallel Sorters (Bitonic Sorting Network)





Bitonic Network (BN)
Bitonic Merger (BM)

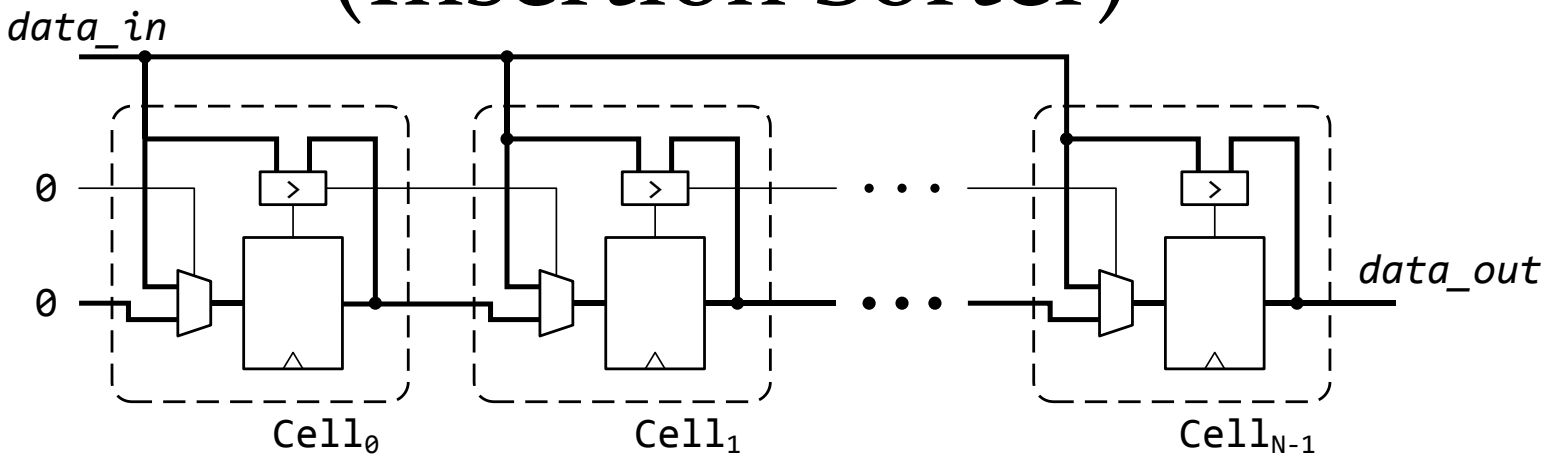
Data Set Size: P

Throughput: P

Size(Compare): $P\log^2(P)$

Delay: $\log^2(P)$

Sequential Sorters (Insertion Sorter)



3						
12	3					
7	3	12				
1	3	7	12			
9	1	3	7	12		
20	1	3	7	9	12	
	1	3	7	9	12	20

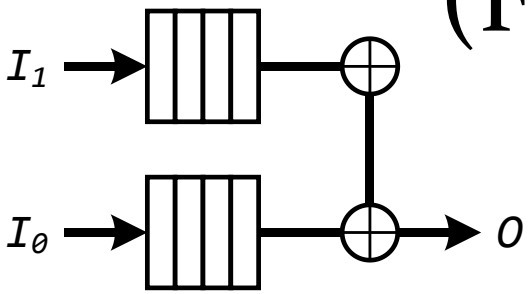
Data Set Size: N

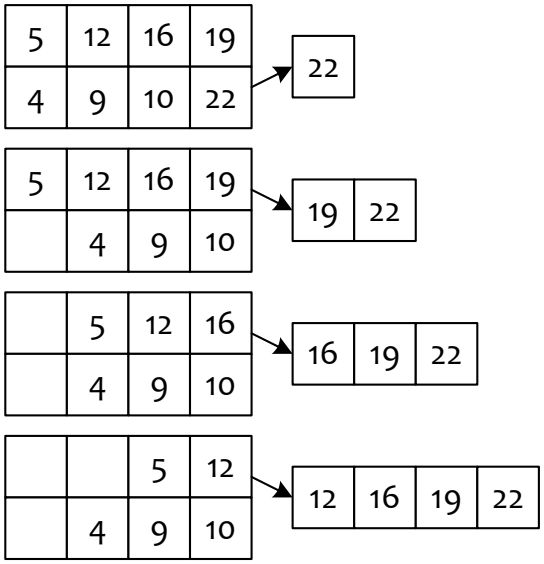
Throughput: **1**

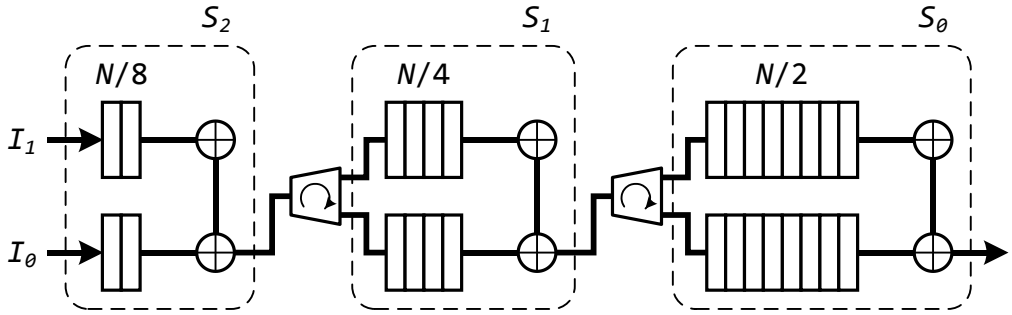
Size(cells): N

Delay: N

Sequential Sorter (FIFO-merge)







Data Set Size: N

Throughput: **1**

Size(Memory): $2N$

Delay: N

D. Koch and J. Torresen, "**FPGASort: a high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting**," in *Proc. of FPGA*, February 2011, pp. 45–54.

Summarise Existing Sorters

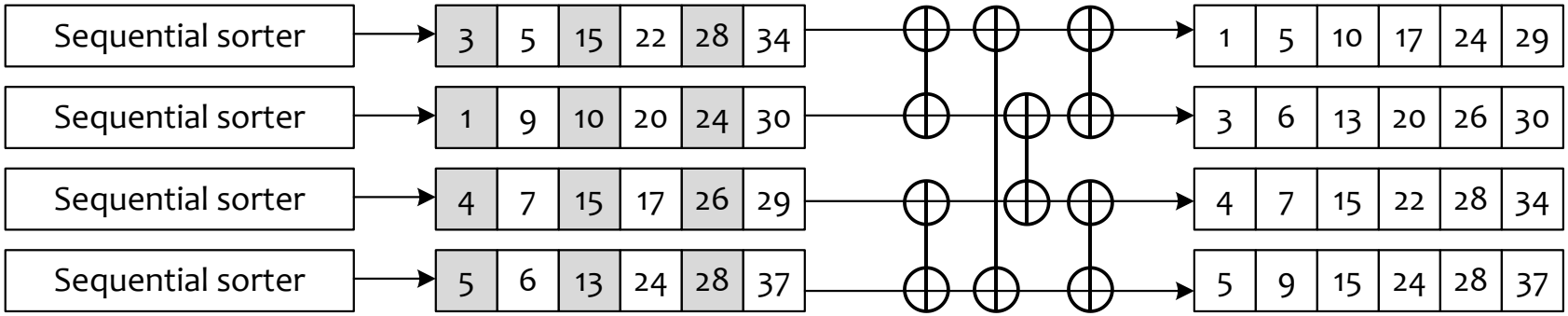
- Parallel Sorters
  - High throughput
  - Area increases significantly with the quantity of data
  - Sorting a small quantity of numbers
- Sequential Sorters
  - Linear area overhead
  - Feasible for large data sets
  - Low throughput

Can we dynamically merge multiple sequential sorters?

Parallel Merging

Merge multiple sequential sorters using a Bitonic network?

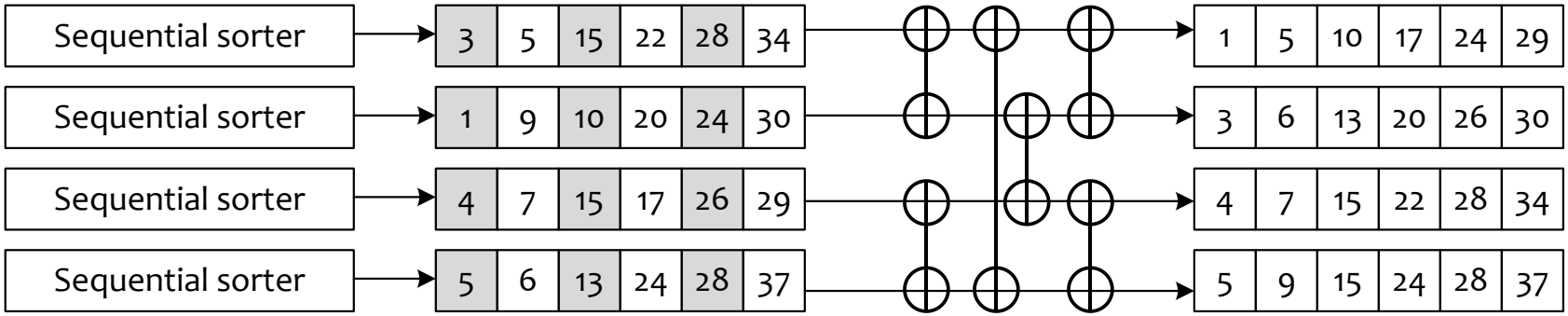
YES



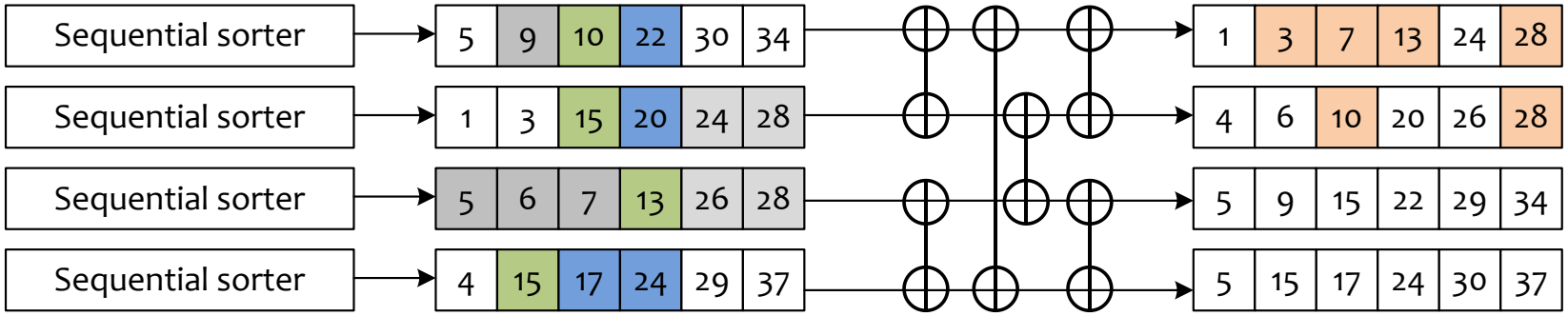
Parallel Merging

Merge multiple sequential sorters using a Bitonic network?

YES



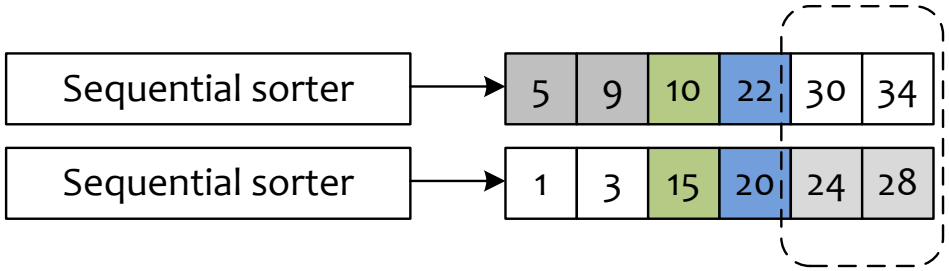
NO!



Numbers may not be distributed evenly among sequences.

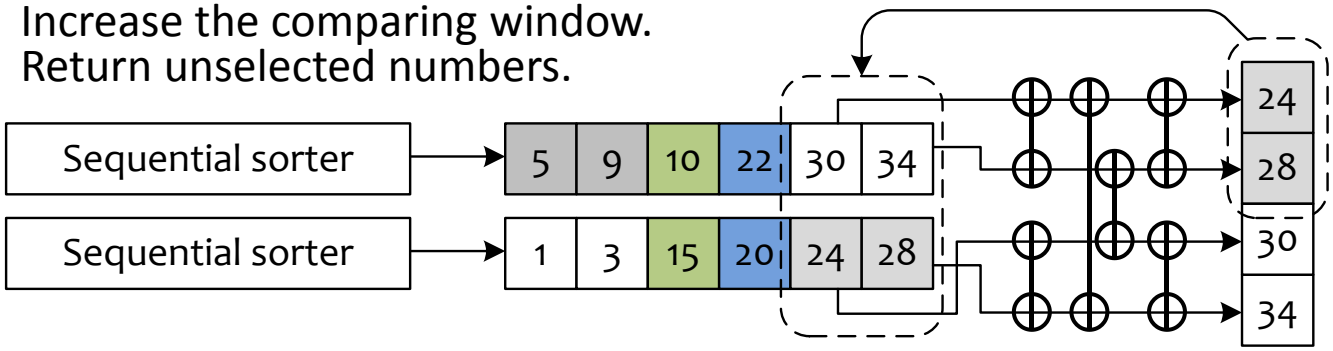
Parallel Merging

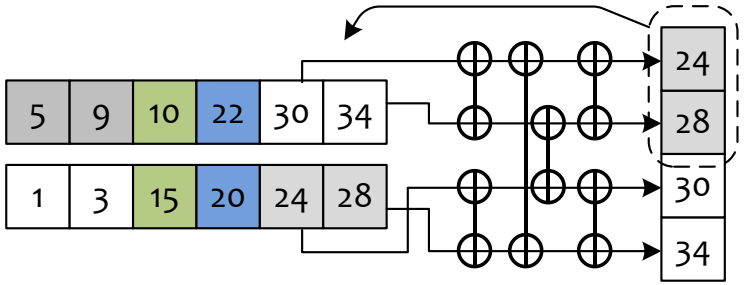
Increase the comparing window.

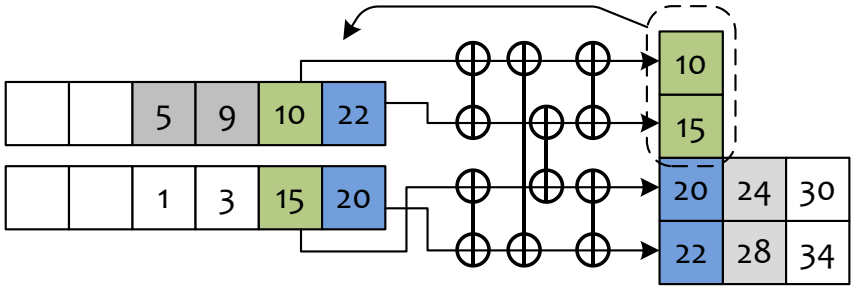


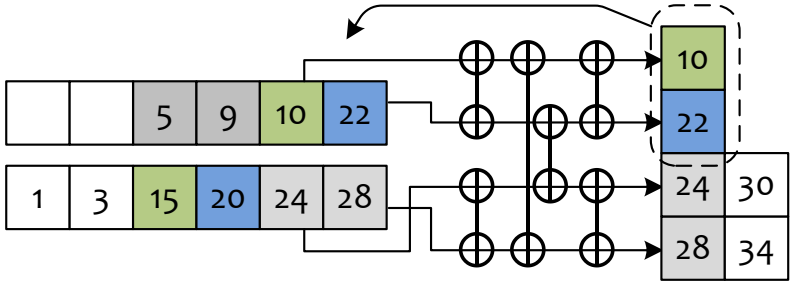
Parallel Merging

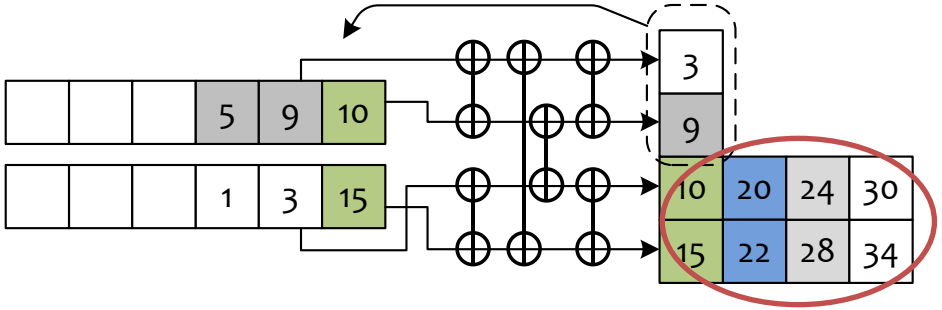
Increase the comparing window.
Return unselected numbers.





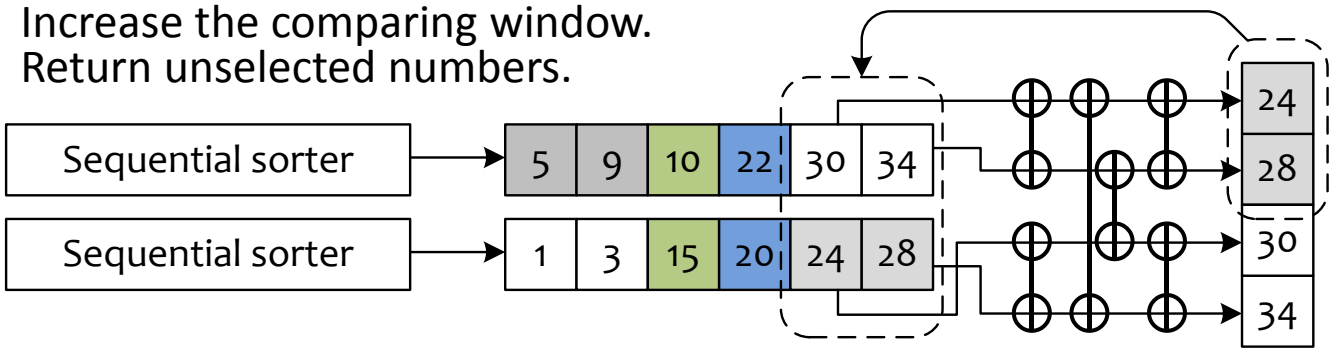






YES!

Parallel Merging

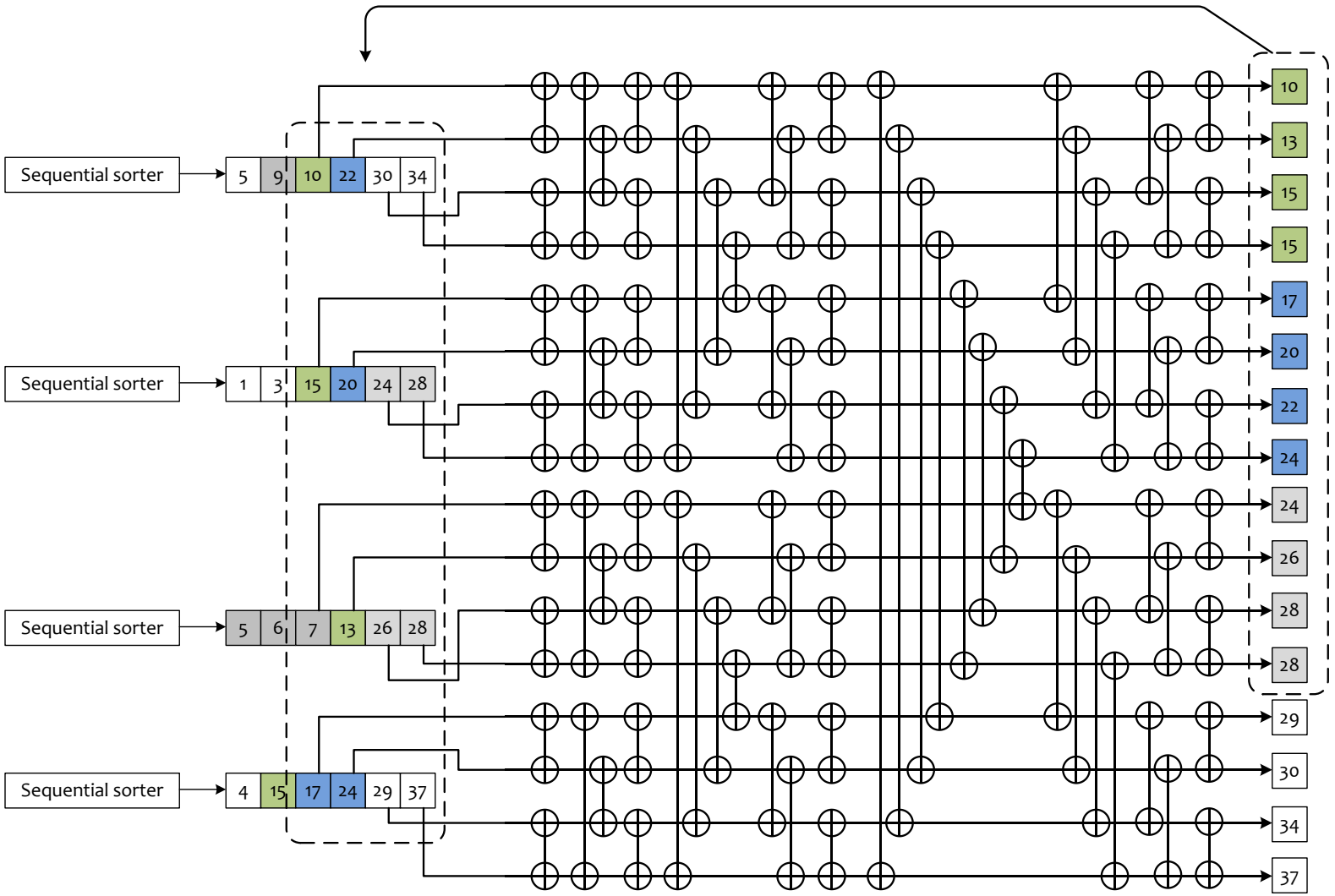


Requirement:

To merge ***S*** pre-sorted sequence and at a speed of ***S*** numbers per cycle,

1. Increase the comparing window to ***S*** x ***S***;
2. Using an ***S*** x ***S*** -input Bitonic sorting network; [Area overhead]
3. Return the ***S*** x (***S*** - 1) unselected numbers; [Control overhead]
4. Unselected numbers should be returned in one cycle. [Slow clock]
5. Maximal shifting rate of ***S*** numbers per cycle. [Speed mismatch]

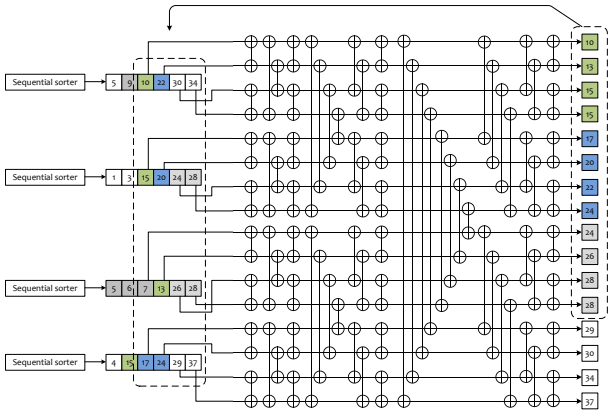
Parallel Merging



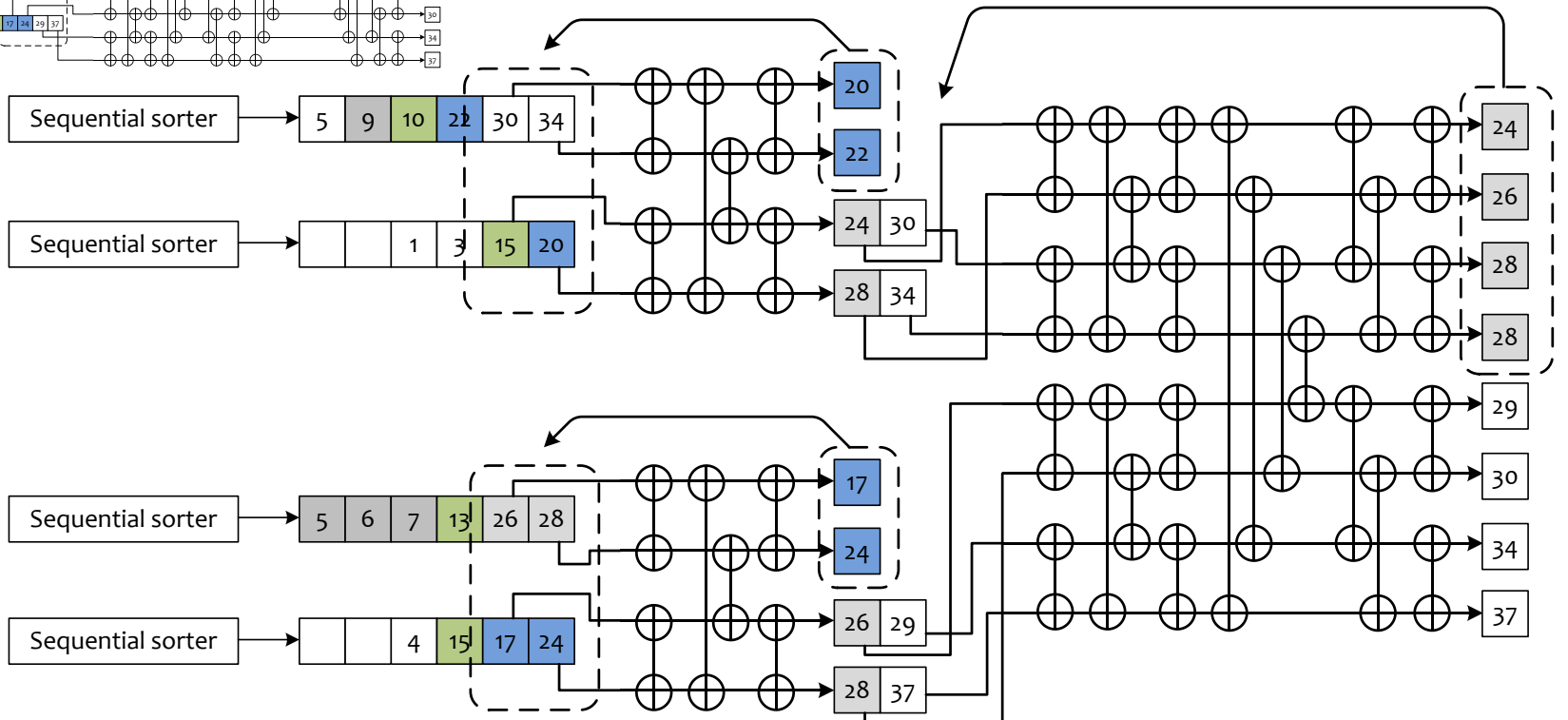
MANCHESTER 1824

The University of Manchester

Optimising the Parallel Merging



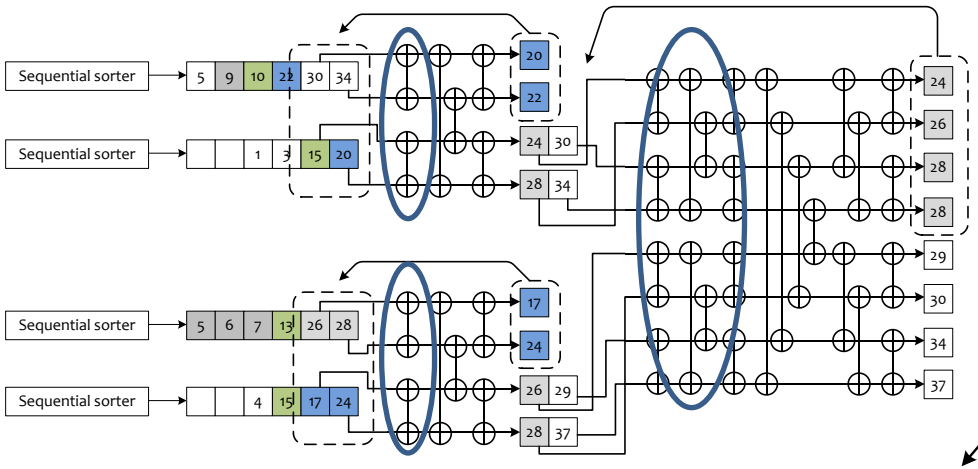
Using a tree structure reduces the number of comparators by > 50%.



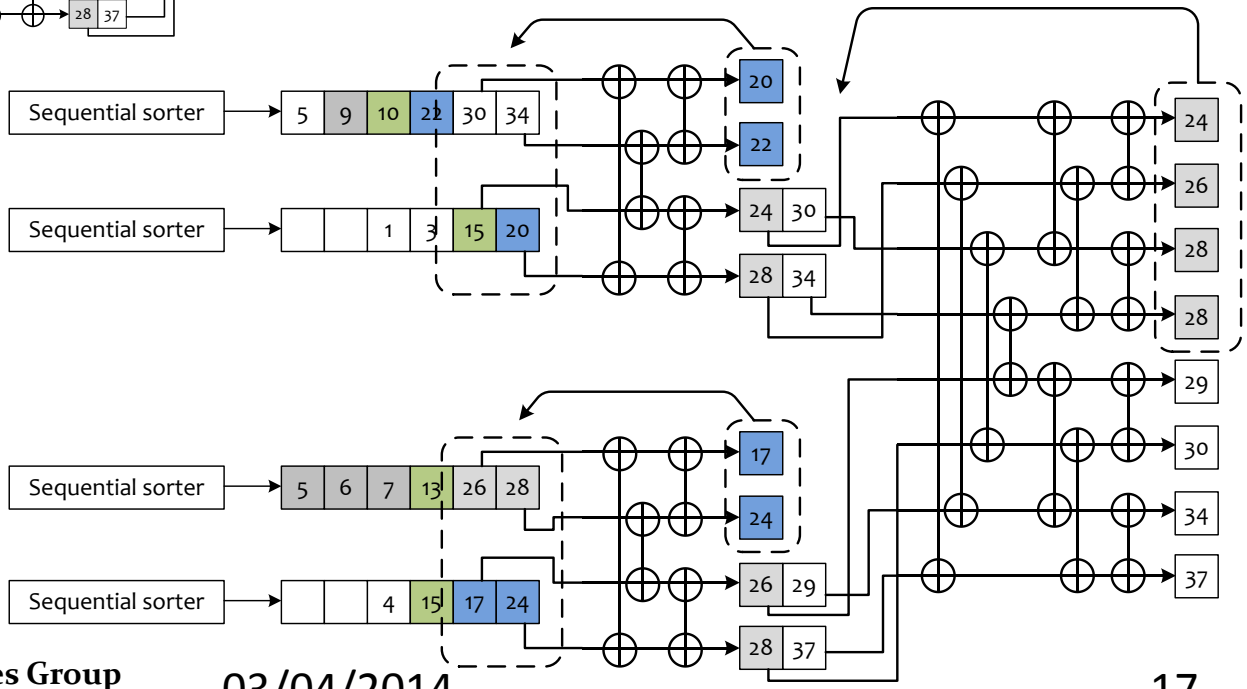
MANCHESTER 1824

The University of Manchester

Optimising the Parallel Merging

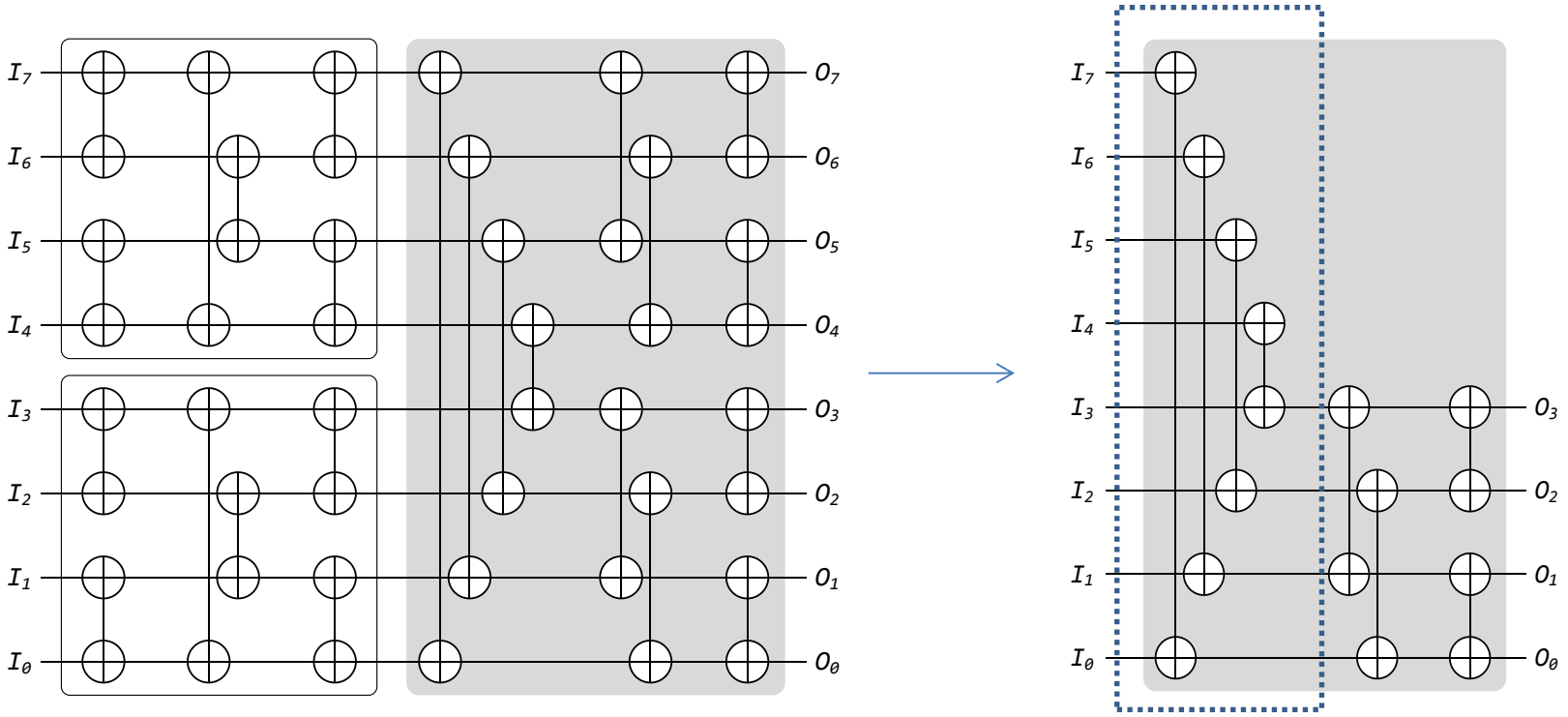


Replace the Bitonic sorting networks with Bitonic mergers because the sequences are pre-sorted.



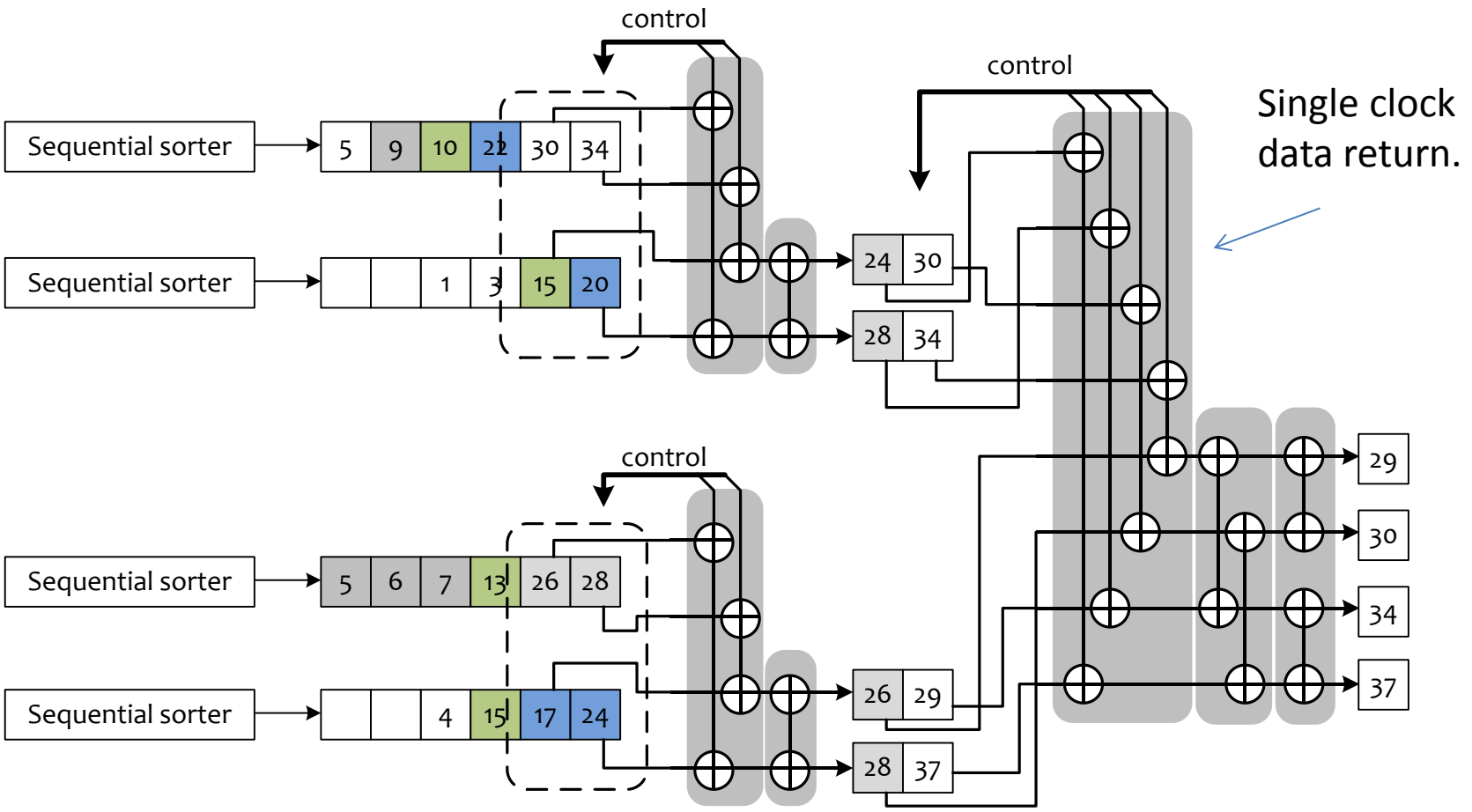
1. Reduce the comparing window.
2. Reduce the size of soring networks.
3. Reduce the numbers being returned.

Bitonic Partial Merger

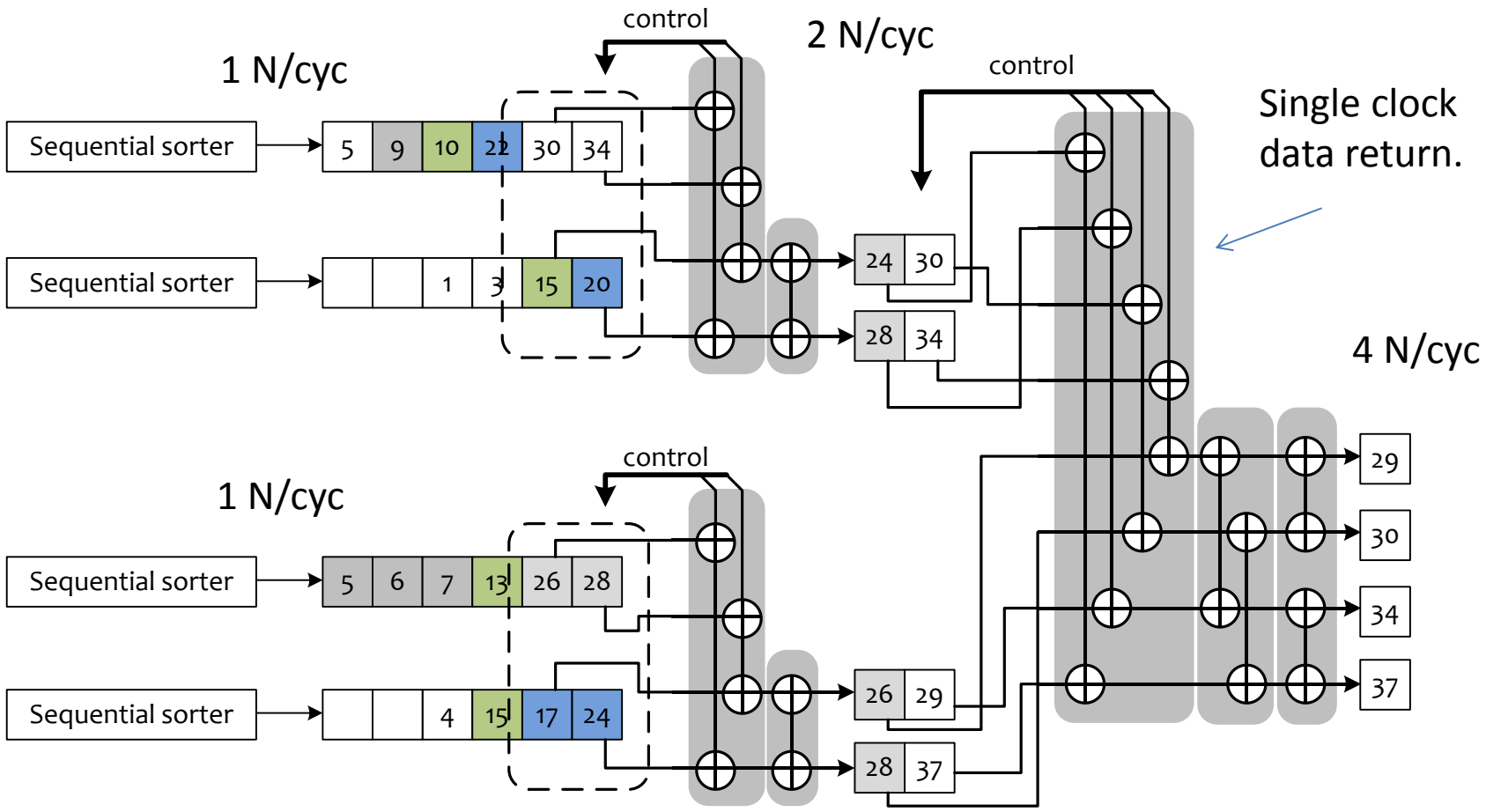


- A. Farmahini-Farahani, H. J. Duwe, III, M. J. Schulte, and K. Compton, "**Modular design of high-throughput, low-latency sorting units**," *IEEE Transactions on Computers*, vol. 62, no. 7, pp. 1389–1402, July 2013.

Optimising the Parallel Merging

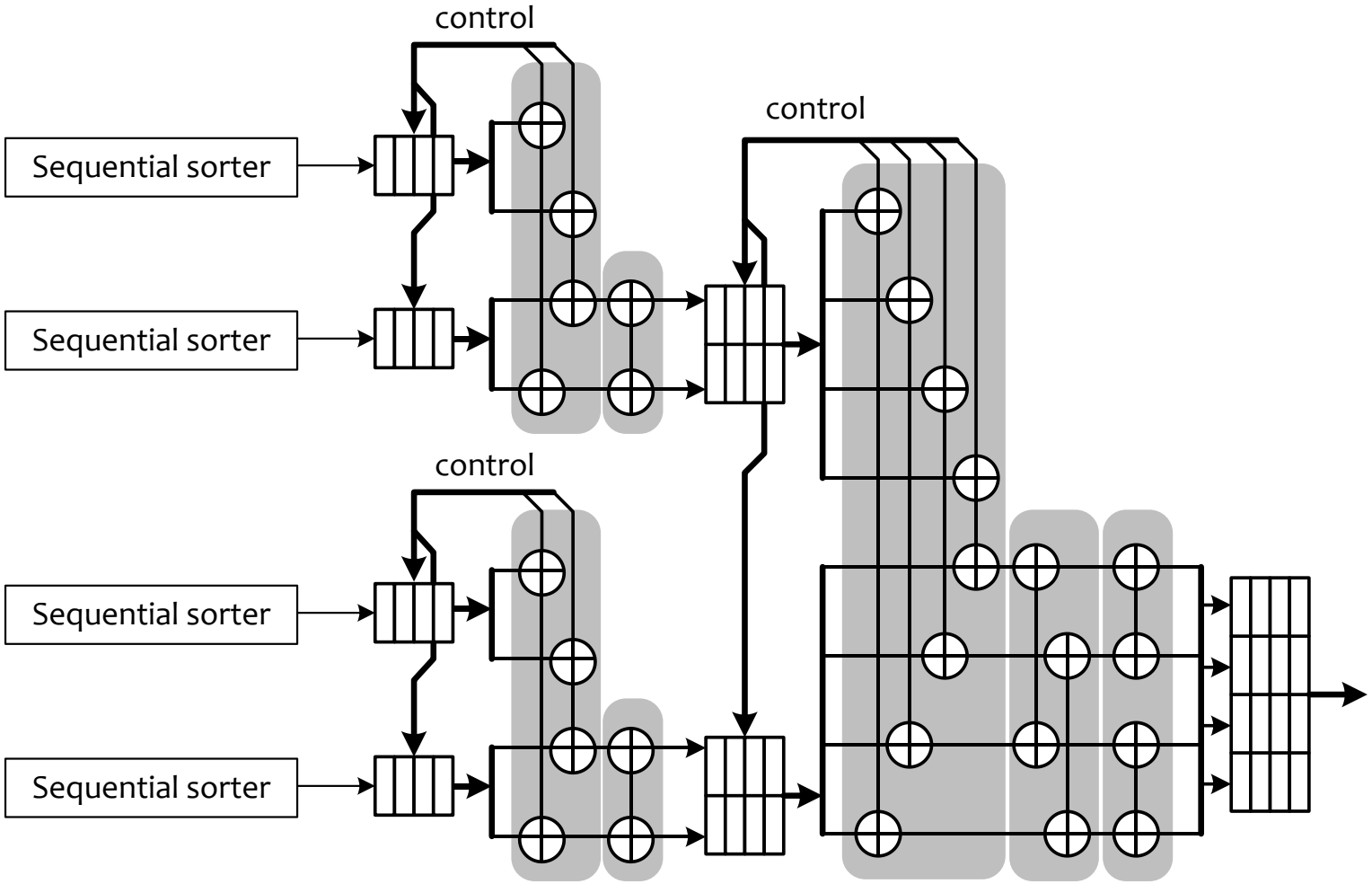


Optimising the Parallel Merging

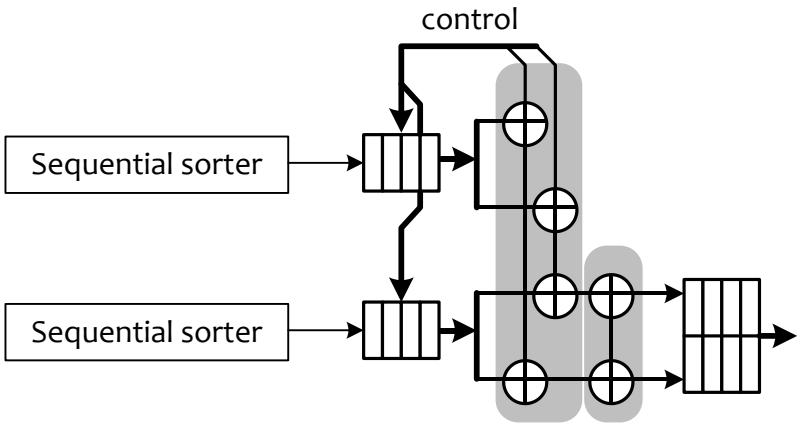


The last issue: Speed mismatch between inputs and outputs.

Speed Mismatch: Using FIFO and Allow Stalls



How Stalls Occur



Even distribution has 0 stall.

Original Sequences

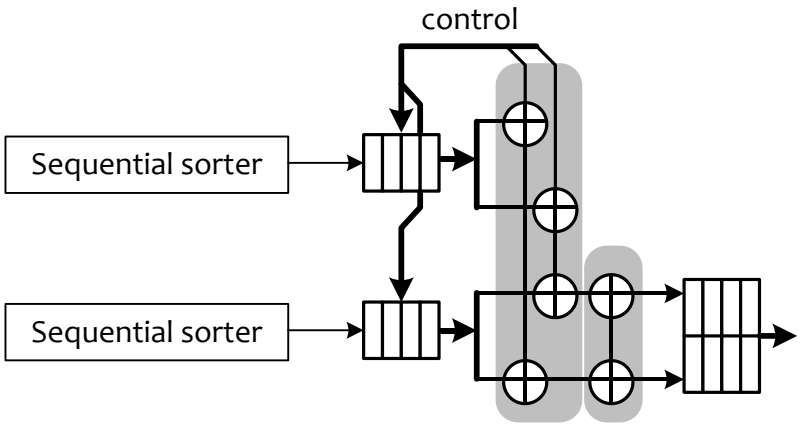
16	4	20	6	10	12	18	2	14	8
11	3	9	5	13	1	19	7	17	15

Pre-sorted

2	4	6	8	10	12	14	16	18	20
1	3	5	7	9	11	13	15	17	19

0 stall $R = 0\%$ $\alpha = 0$

How Stalls Occur



Uneven distribution may reduce the speed to the minimum of 1 N/Cyc.

Original Sequences

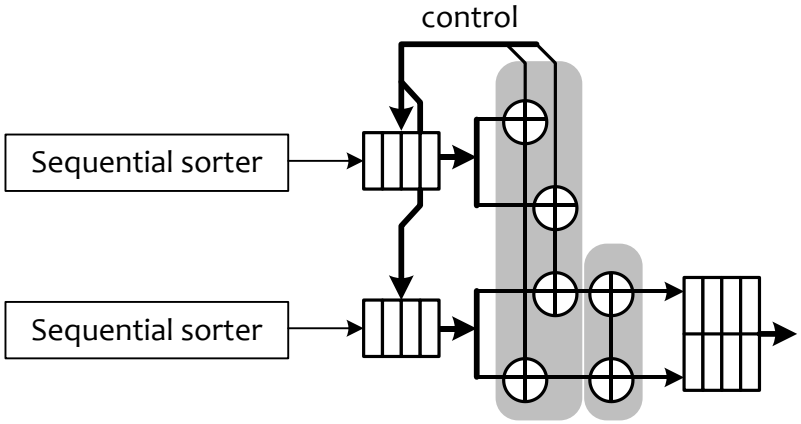
18	12	20	13	15	16	19	11	17	14
6	3	9	2	7	1	10	4	5	8

Pre-sorted

11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10

10 stalls $R = 50\%$ $\alpha = 1.0$

How Stalls Occur



Random distribution has average stall rates which can be reduced using long FIFO.

Original Sequences

6	12	9	13	7	1	19	4	17	14
18	3	20	2	15	16	10	11	5	8

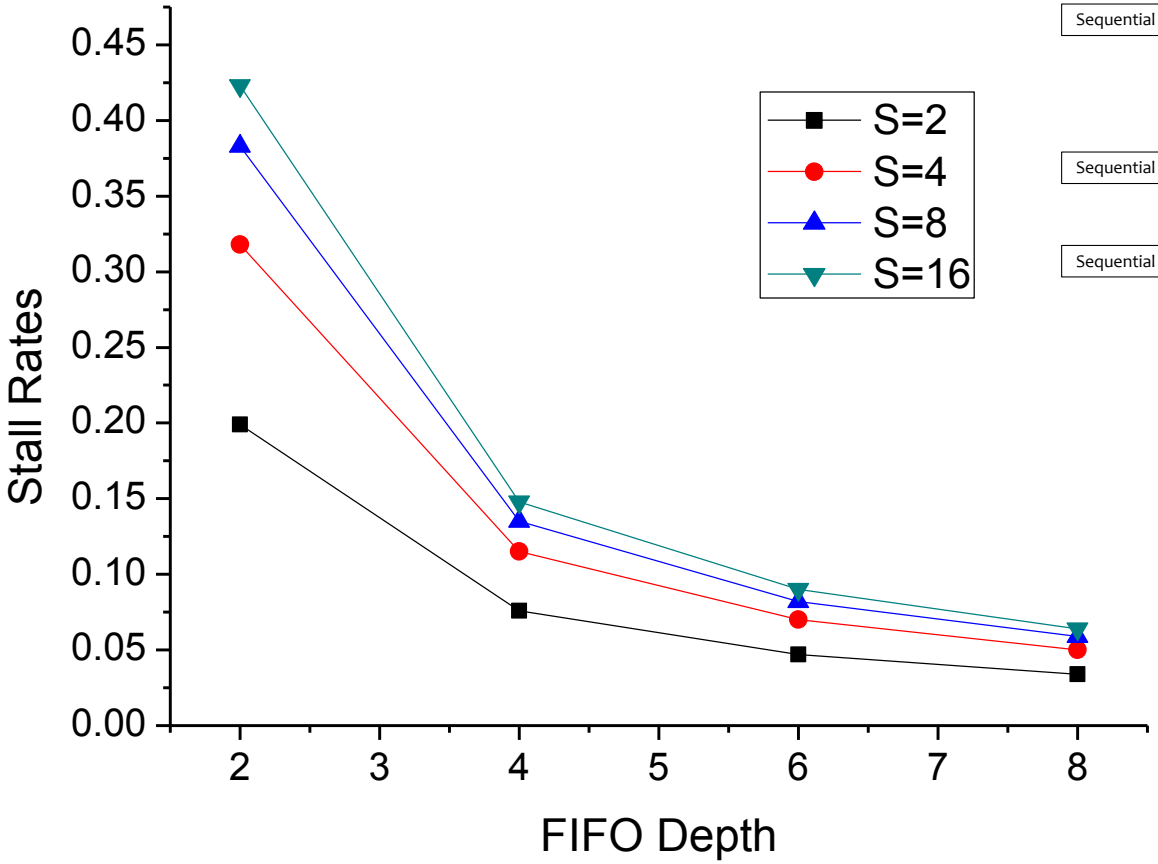
Pre-sorted

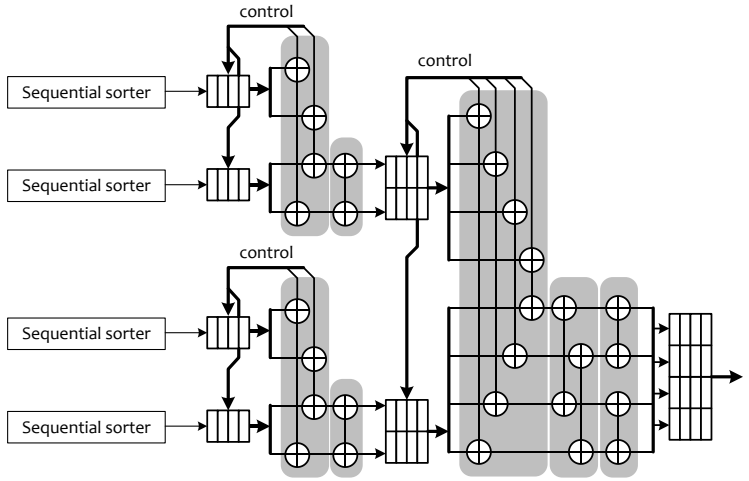
1	4	6	7	9	12	13	14	17	19
2	3	5	8	10	11	15	16	18	20

2 stalls $R = 17\%$ $\alpha = 0.2$

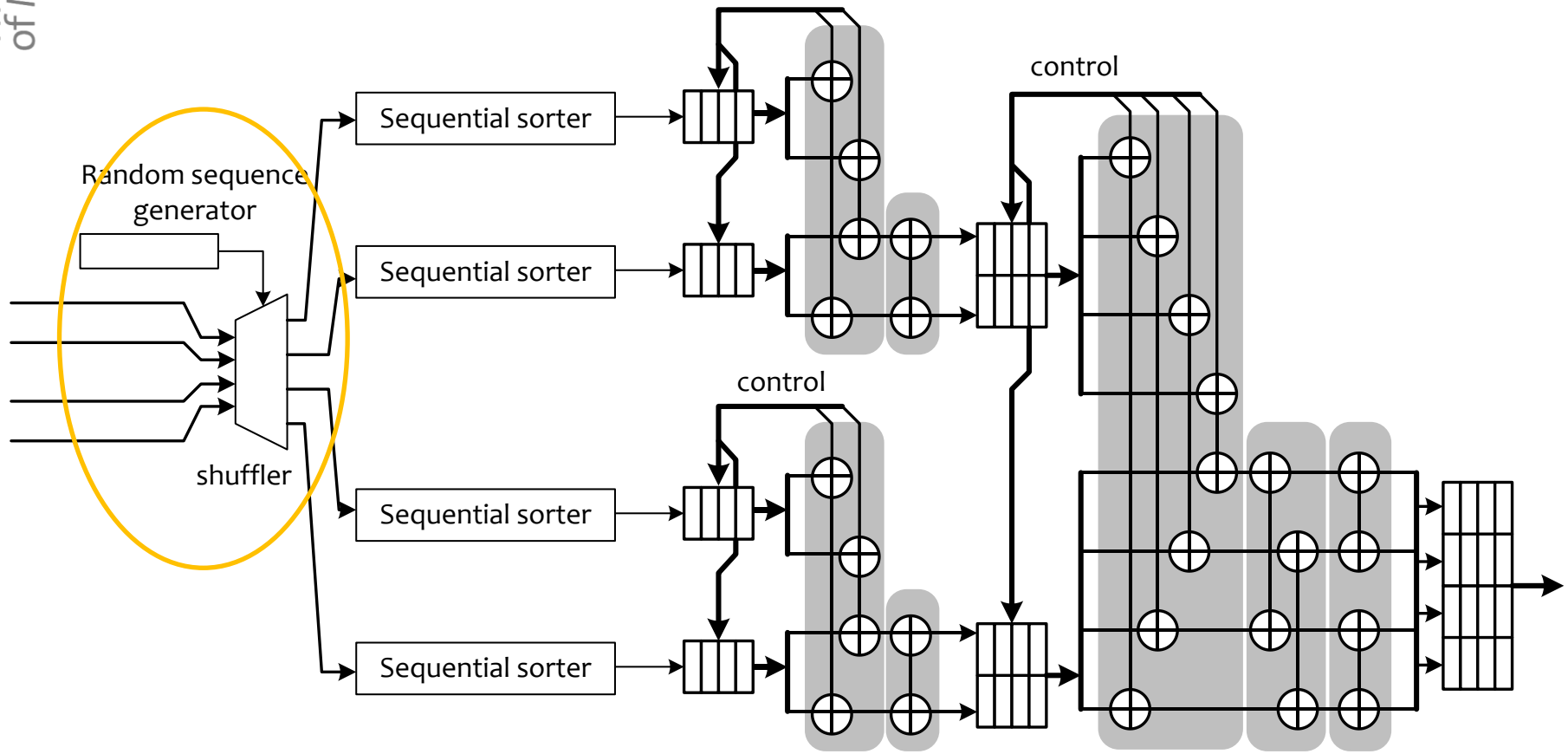
0 stall if using an FIFO with its depth > 2.

Reducing Stalls using Long FIFOs

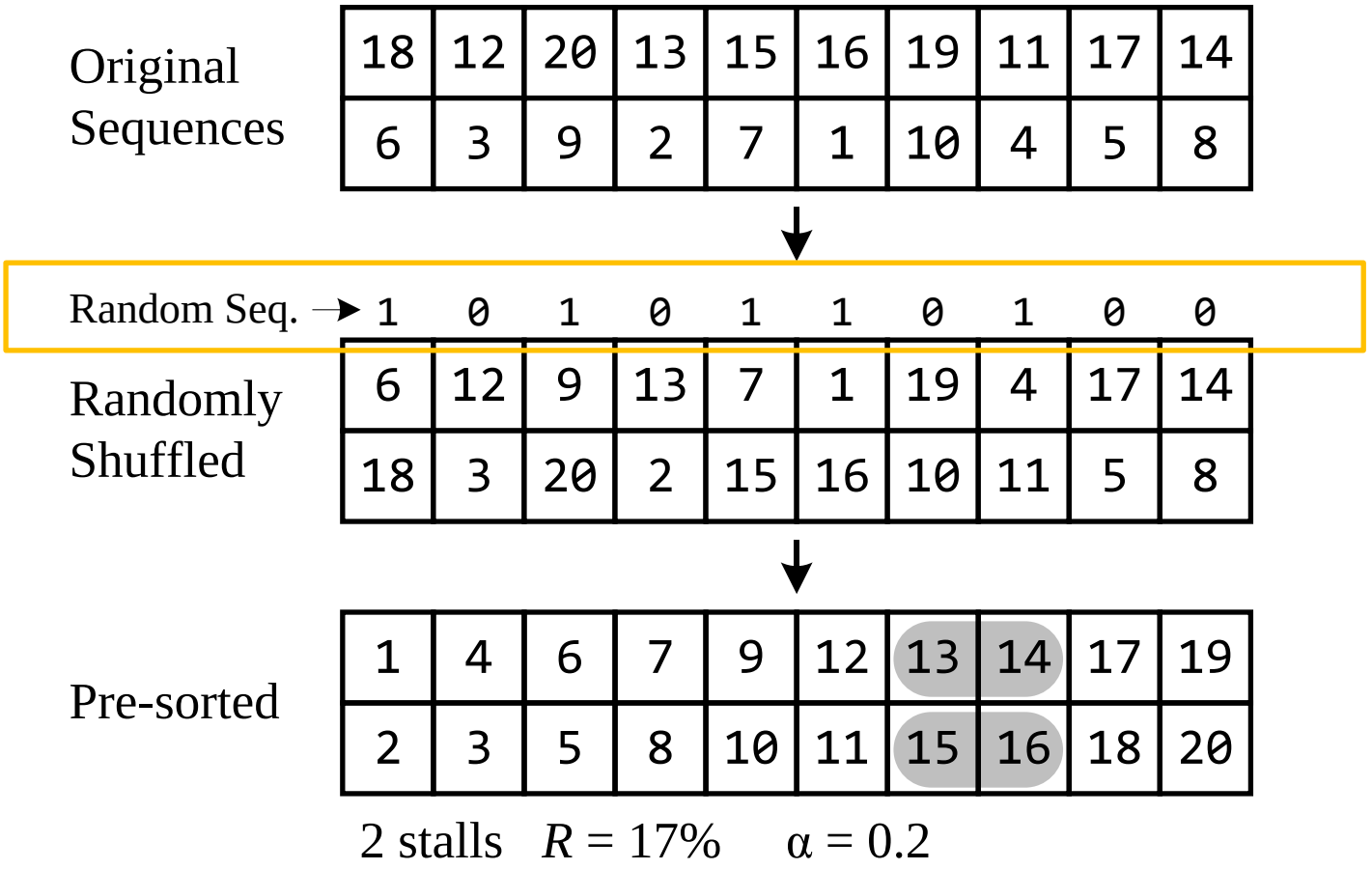




Handle Uneven Distribution



Handle Uneven Distribution



2 stalls R = 17% α = 0.2

Some results

Sorters	Frequency	Slices	RAM	No. of Records	Throughput
PMT(4)[Virtex 7]	226MHz	70%	100%	459K	51.5 Gb/s
PMT(8) [Virtex 7]	206MHz	96%	26%	115K	91.4 Gb/s
PMT(16) No seq. [Virtex 7]	202MHz	96%	0%		193.6 Gb/s
Seq. (Dirk Koch*) [Virtex 5]	252MHz	74%	98%	43K	16 Gb/s
PCIe 2.1 x8 [Vietex 7]	250Mhz				32 Gb/s
PCIe 3.0 x8 [Virtex 7]	250MHz				64 Gb/s

* D. Koch and J. Torresen, "**FPGASort: a high performance sorting architecture exploiting run-time reconfiguration on FPGAs for large problem sorting**," in *Proc. of International Symposium on Field Programmable Gate Arrays*, February 2011, pp. 45–54.

Summary

- The results of two sequential sorters can be dynamically merged using a Bitonic partial merger.
- Multiple sequential sorters can be merged using a tree of Bitonic partial mergers.
- Allowing stalls, the speed mismatch issue can be alleviated using long FIFOs and random shufflers.
- **The throughput limit of 1 Number / cycle is scrapped.**

MANCHESTER 1824

The University of Manchester

THANKS!