An 11-Step Sorting Network for 18 Elements

Sherenaz W. Al-Haj Baddar, Kenneth E. Batcher

Kent State University
Department of Computer Science
Kent, Ohio 44240
salhajba@cs.kent.edu
batcher@cs.kent.edu

***Abstract*— Sorting Networks are cost-effective multistage interconnection networks with sorting capabilities. These networks theoretically consume Θ(NlogN) comparisons. However, the fastest implementable sorting networks built so far consume Θ(Nlog^{2}N) comparisons, and generally, use the Merge-sorting strategy to sort the input. An 18-element network using the Merge-sorting strategy needs at least 12 steps-here we show a network that sorts 18 elements in only 11 steps.**

***Index Terms*—Sorting Networks, Partial Ordering, 0/1 cases.**

I. Introduction

Parallel processors are fast and powerful computing systems that have been developed to help undertake computationally challenging problems. Deploying parallelism for solving a given problem implies splitting the problem into subtasks. Each subtask is assigned to one of the numerous computing components constituting a parallel system. These components usually communicate in order to accomplish their designated subtasks, which introduces the problem of connecting them efficiently. Several interconnection networks schemes have been built to help solve this problem, among which are multistage interconnection networks. These widely used networks deploy a significantly smaller number of switching elements to achieve a relatively more efficient inter-processor communication. Many multistage interconnection networks were developed including sorting networks.

Van Voorhis [1] defines a sorting network as a circuit with N inputs and N outputs such that for any set of inputs $\{I_1,I_2,\ldots,I_N\}$, the resulting output is the set $\{O_1,O_2,\ldots,O_N\}$. The output set must be a permutation of the input set $\{I_1,I_2,\ldots,I_N\}$. Moreover, for every two elements of the output set O_j and O_k, O_j must be less than or equal to O_k whenever $j \leq k$.

Sorting networks are constructed using stages (steps) of basic cells called Comparator Exchange(CE) modules. A CE is a 2-element sorting circuit. It accepts two inputs via two input lines, compares them and outputs the larger element on its high output line, whereas the smaller is output on its low output line. It is assumed that two comparators with disjoint inputs can operate in parallel. A typical CE is depicted in Figure 1[2].

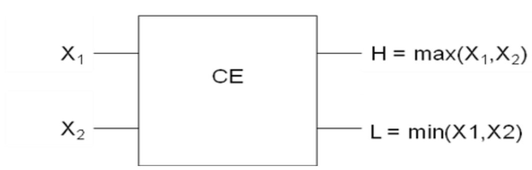


Figure 1. A comparator exchange module[2]

An optimal N-element sorting network requires θ(NlogN) comparisons to sort N elements[3]. However, no practical solution for building such networks has been developed so far. On the other hand, several practical θ(Nlog^{2}N) techniques for building sorting networks exist among which is the Merge-sorting developed by Batcher[2]. This technique has been generally deployed for building the best performing sorting networks known so far. Accordingly, the best known network for sorting 18 elements uses the Merge-sorting and consumes 12 steps. Nevertheless, an 11-step network for the same input size has been discovered and is illustrated here. This discovered network shows that further investigation can result in developing implementable networks that are closer to the optimal complexity of θ(NlogN).

Section 2 reviews some basic mathematical concepts necessary for discussing the 11-step network illustrated here. Section 3 depicts the 11 steps constituting the developed network using Sortnet, a software tool developed by Batcher to help build better sorting networks[4]. Finally, Section 4 concludes the technical report and highlights future work.

2. MATHEMATICAL BACKGROUND

This section illustrates some basic mathematical concepts necessary for discussing the developed sorting network. These concepts include: Knuth diagrams, 0/1-principle, partial ordering , Haase diagrams, and Shmoo charts.

2.1 Knuth diagrams

Knuth diagrams are pictorial representations of sorting networks that help distinguish the several steps constituting the investigated network[5]. In a typical Knuth diagram, each input element is represented by a horizontal line and each CE is represented by a vertical line connecting the two elements being compared. The elements being sorted are assumed to have numeric labels. Thus, an N-element network will have its input elements labeled from 0 through (N-1). Moreover, the network's top most element is assigned the highest label and the bottom most element is assigned the smallest. It is also assumed that the elements are fed into the left most end of the network and received sorted at the other end, with the maximum element being the top most element. Figure 2 illustrates a typical Knuth diagram for sorting 4 elements[5].

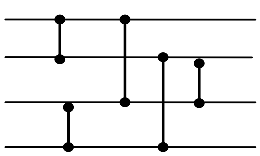


Figure 2. A Knuth diagram for a 4-element sorting network

2.2. The 0/1 principle

The 0/1 principle plays a vital role in building and verifying sorting networks. It states that a given N-element sorting network sorts N inputs correctly if it sorts all the 2^N binary strings of length N [5].

A 0/1 case is a sequence of length N where the value of each entry in the sequence is either 0 or 1. If a given 0/1 case, A, of length N, has j zeros then it has N-j ones. When a comparator element compares two entries in A, it might swap the values of the compared entries. However, the number of zero and one entries in A remains the same. Hence, a series of comparator elements sorts case A iff it rearranges A's entries such that the first j locations hold zeros and the next N-j locations hold ones.

The number of zeros(ones) in a given binary sequence of N bits ranges between 0 and N which implies that there are N+1 0/1 sorted cases. Hence, an N-element sorting network, that rearranges any of the 2^N input permutations into one of the N+1 possible sorted 0/1 cases, can sort any arbitrary sequence of N elements, and is said to be a valid N-element sorting network.

2.3. Partial ordering and Haase diagrams

A total order relation is imposed on the set of elements sorted by a valid sorting network. This total ordering is achieved by the end of the sorting process. However, only a partial order relation exists on this set at any arbitrary step in the sorting process prior to the last step, and such a set is called a **p**artial **o**rder **set** (poset)[6]. Having a partial ordering relation, denoted by R, on a set of elements implies that there may exist a pair of elements x and y, such that there is at least one 0/1 case where x=0 and y=1, and at least another 0/1 case, where x = 1 and y = 0.

Haase diagrams are used in order to visualize the progress of a sorting network [6]. In a conventional Haase diagram, elements are represented by vertices and relations among them are represented by edges. An edge running from vertex x to vertex y exists iff there does not exist a 0/1 case in which x=0 and y=1. It is noticed that the relative positioning of an edge's two endpoints implies its direction. More preciously, if vertex x appears above vertex y, then the direction of the edge running between both is assumed to be from x to y.

Figure 3 depicts the progress of the sorting network illustrated in Figure 2 using Haase diagrams. In the first step, two partial ordering relations are formed and such diagram is called a multi-segment poset. In the second step, the two segments are combined into a one-segment poset, or simply a poset. Finally, step 3 transforms this poset into a totally ordered chain of elements.

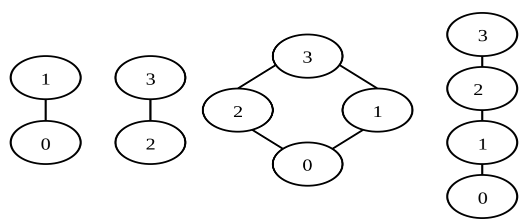


Figure 3. Haase diagrams tracking the sorting of 4 elements

2.5. Shmoo Charts

A Shmoo chart is a two-dimensional diagram where each column shows all 0/1 cases with the same number of zeros and each row shows an element[4]. Figure 4 illustrates the Shmoo

```
      Number of Zeroes in Case
      00000    No. of Cases
      43210    where key = 1

   3: 01111 :            5
   2: 00-11 :            3
   1: 00-11 :            3
   0: 00001 :            1
```

Figure 4. The Shmoo chart of the 4-element network after step 2

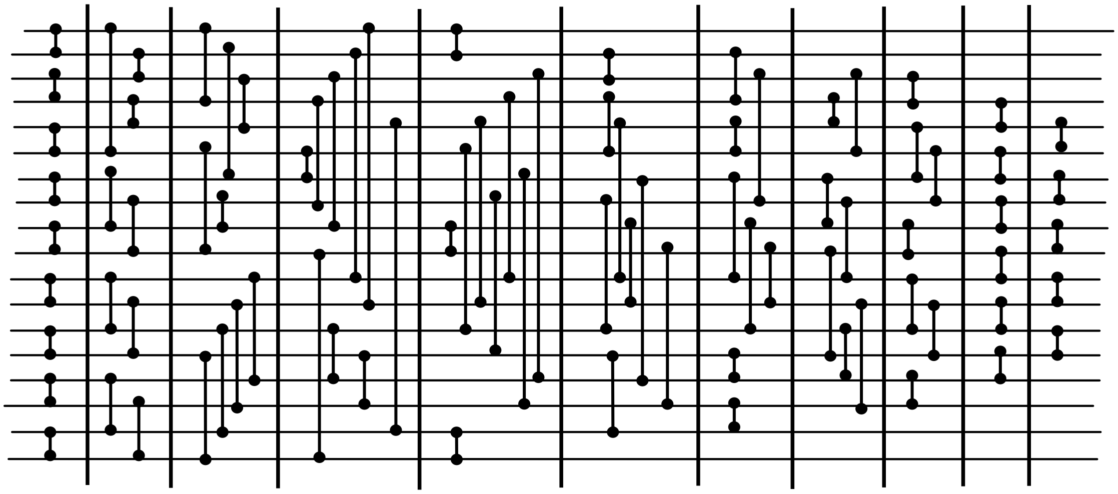


Figure 5. The Knuth diagram of the developed solution

chart generated after step 2 of the 4-element network depicted in Figure 3.

The columns of a typical Shmoo chart are ordered according to the number of zeros in the set of 0/1 cases with the case of all zeros at the left of the chart and the one with all ones at the right of it. The elements in the rows are ordered according to the number of the 0/1 cases in which that particular element is 1. The entry in the chart for a given row and a given column can be either:

- 0: if the value of the element is 0 for all 0/1 cases in a given column.
- 1: if the value of the element is 1 for all 0/1 cases in a given column.
- -: if the value of the element is equal to 0 at least for one 0/1 case and equal to 1 at least for another 0/1 case in a given column.

A shmoo chart becomes dash-free when all elements get sorted , i.e. all entries in the Shmoo chart are either zeroes or ones. Shmoo charts are generated by Sortnet, the software package developed by Batcher, in order to help monitor the progress of a typical sorting network[4].

3. An 11-step solution for sorting 18 elements

This section describes an 11-step solution for sorting 18 elements that outperforms the merge-based, 12-step solution which used to be the best known solution for this problem. Figure 5 illustrates the developed network.

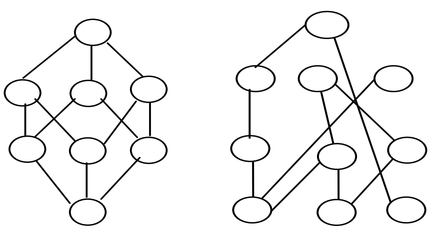


Figure 6. The two-segment poset obtained after applying step 3.

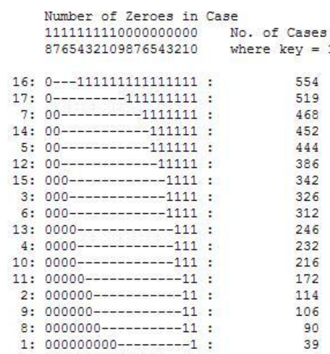
3.1. The first three steps

The first two steps construct a four-segment poset. Three out of the four segments contain 4 elements each and the last contains 6 elements. The third step connects two 4-element segments to get a segment of 8 elements and connects the two remaining segments to form a segment of 10 elements. Hence, step 3 results in a two-segment poset with one segment containing 8 elements and the other containing 10 elements. Figure 6 describes this two-segment poset.

3.2. The fourth step

In this step, a one-segment poset will be formed by doing nine comparisons. Four of these comparisons connect elements that belong to the two different segments, while the rest of them connect elements that belong to the same segment. The comparisons conducted in this step are described in Figure 5. Due to the fact that the poset becomes messy at this point, Shmoo charts will be used starting from this step to illustrate the effect of deploying additional CEs to the network. Hence, Figure 7 illustrates the Shmoo chart obtained after applying the CEs in step 4.

3.3. The fifth, sixth, and seventh steps



```
     Number of Zeroes in Case
     111111111000000000    No. of Cases
     876543210987654321O    where key = 1

16: 0---11111111111111 :          554
17: 0---------11111111 :          519
 7: 00---------1111111 :          468
14: 00----------111111 :          452
 5: 00----------111111 :          444
12: 00-----------11111 :          386
15: 000-----------1111 :          342
 3: 000-----------1111 :          326
 6: 000-----------1111 :          312
13: 0000-----------111 :          246
 4: 0000-----------111 :          232
10: 0000-----------111 :          216
11: 00000-----------11 :          172
 2: 000000----------11 :          114
 9: 000000----------11 :          106
 8: 0000000---------11 :           90
 1: 000000000--------1 :           39
 0: 00000000000000---1 :            4
```

Figure 7. The Shmoo chart of the first 4 steps

The strategy used for comparing elements in these steps is to compare each two adjacent elements in the Shmoo chart. Thus, each of these steps uses the Shmoo chart generated by the step before in order to pick the pairs of elements to be compared. Figure 8 depicts the Shmoo chart that is generated from applying the CEs of steps 5,6, and 7, as described in Figure 5.

```
     Number of Zeroes in Case
     1111111110000000000    No. of Cases
     8765432109876543210    where key = 1

17:  0111111111111111111 :            75
16:  0011111111111111111 :            74
14:  000---1111111111111 :            68
13:  000----111111111111 :            68
15:  000------1111111111 :            65
12:  0000------111111111 :            55
11:  00000-----111111111 :            54
10:  00000------11111111 :            47
 9:  000000------1111111 :            44
 7:  0000000------111111 :            32
 4:  00000000------11111 :            29
 5:  000000000-----11111 :            22
 8:  000000000------1111 :            21
 3:  00000000000-----111 :            11
 6:  000000000000----111 :             8
 2:  0000000000000---111 :             8
 1:  0000000000000000011 :             2
 0:  0000000000000000001 :             1
```

Figure 8. The Shmoo chart after step 7

3.4. The eighth step

Several different ways do exist for picking the pairs of elements to form step 8. However, experimentation showed that doing step 8 the way depicted in Figure 5 helped reduce the overall number of steps required to complete the sorting task. Figure 9 illustrates the Shmoo chart obtained by applying the CEs in step 8 as depicted in Figure 5.

```
     Number of Zeroes in Case
     1111111110000000000    No. of Cases
     8765432109876543210    where key = 1

17:  0111111111111111111 :            52
16:  0011111111111111111 :            51
14:  000-111111111111111 :            49
15:  000---1111111111111 :            47
13:  0000---111111111111 :            44
11:  00000---11111111111 :            40
12:  00000-----111111111 :            36
10:  00000-----111111111 :            36
 9:  0000000-----1111111 :            28
 8:  00000000----1111111 :            25
 7:  000000000----111111 :            20
 5:  000000000----111111 :            18
 4:  00000000000----1111 :            11
 6:  000000000000---1111 :             9
 3:  0000000000000---111 :             7
 2:  000000000000000-111 :             4
 1:  0000000000000000011 :             2
 0:  0000000000000000001 :             1
```

Figure 9. The Shmoo chart after step 8

3.5. The ninth step

In this step, the same pattern used for steps 5, 6, and 7 is used again, i.e., each two adjacent elements in the Shmoo chart generated by step 8 are going to be compared in this step. This implies that only seven pairs of comparisons are needed, since the two smallest as well as the two largest elements were found by the steps before. Applying the seven CEs of step 9 as described in Figure 5 results in the chart depicted in Figure 10.

```
     Number of Zeroes in Case
     1111111110000000000    No. of Cases
     8765432109876543210    where key = 1

17:  0111111111111111111 :            33
16:  0011111111111111111 :            32
15:  0001111111111111111 :            31
14:  0000--1111111111111 :            28
13:  0000--1111111111111 :            28
12:  00000---11111111111 :            24
11:  00000---11111111111 :            24
10:  0000000---111111111 :            19
 9:  0000000---111111111 :            19
 8:  000000000---1111111 :            14
 7:  000000000---1111111 :            14
 5:  00000000000--111111 :            10
 6:  00000000000---11111 :             9
 3:  0000000000000--1111 :             6
 4:  00000000000000-1111 :             5
 2:  0000000000000000111 :             3
 1:  0000000000000000011 :             2
 0:  0000000000000000001 :             1
```

Figure 10. The Shmoo chart after step 9

3.6. The tenth step

It is quite obvious that the Shmoo chart illustrated in Figure 10 has 6 groups each of which has 2 elements. Consequently, step 10 compares each group's two elements. The CEs used in step 10 are depicted in Figure 5 and the resulting Shmoo chart is described in Figure 11.

```
     Number of Zeroes in Case
     1111111110000000000    No. of Cases
     8765432109876543210    where key = 1

17:  0111111111111111111 :            23
16:  0011111111111111111 :            22
15:  0001111111111111111 :            21
14:  0000111111111111111 :            20
13:  00000-1111111111111 :            18
12:  00000-1111111111111 :            18
11:  0000000-11111111111 :            15
10:  0000000-11111111111 :            15
 9:  000000000-111111111 :            12
 8:  000000000-111111111 :            12
 7:  00000000000-1111111 :             9
 6:  00000000000-1111111 :             9
 5:  0000000000000-11111 :             6
 4:  0000000000000-11111 :             6
 3:  0000000000000001111 :             4
 2:  0000000000000000111 :             3
 1:  0000000000000000011 :             2
 0:  0000000000000000001 :             1
```

Figure 11. The Shmoo chart after step 10

3.7. The last step

The Shmoo chart that results from applying step 10 has 5 groups, and again, each of these groups has two elements. Obviously, step 11 compares each two elements within a group resulting in the last 5 pairs of comparisons described in Figure 5 and the resulting, dash-free Shmoo chart is illustrated in Figure 12.

```
     Number of Zeroes in Case
     1111111110000000000    No. of Cases
     8765432109876543210    where key = 1

17:  0111111111111111111 :            18
16:  0011111111111111111 :            17
15:  0001111111111111111 :            16
14:  0000111111111111111 :            15
13:  0000011111111111111 :            14
12:  0000001111111111111 :            13
11:  0000000111111111111 :            12
10:  0000000011111111111 :            11
 9:  0000000001111111111 :            10
 8:  0000000000111111111 :             9
 7:  0000000000011111111 :             8
 6:  0000000000001111111 :             7
 5:  0000000000000111111 :             6
 4:  0000000000000011111 :             5
 3:  0000000000000001111 :             4
 2:  0000000000000000111 :             3
 1:  0000000000000000011 :             2
 0:  0000000000000000001 :             1
```

Figure 12. The Shmoo chart after step 11

4. CONCLUSION AND FUTURE WORK

Sorting networks are cost-effective multistage interconnection networks with sorting capabilities. The fastest practical sorting networks built so far generally deploy the Merge-sorting developed by Batcher and consume $\theta(N\log^2 N)$ comparisons. Here we describe an 18-element sorting network that is one step less than the fastest known network for the same input size. This network was developed using Sortnet, a software package developed by Batcher to help build better sorting networks.

The 18-element sorting network illustarted here proves that networks that are faster that the best known solutions can be developed. Furthermore, the developed network enables improving the best known networks with input sizes that are multiples of 18. Nevertheless, such a network requires further analysis so that it can be generalized for larger input values that are not multiples of 18 including 24 and 32.

5. REFERENCES

[1] Van Voorhis, D. C., Efficient Sorting Networks, Stanford University, CA, USA, Doctoral Thesis, 1972.

[2] K. E.Batcher, “Sorting Networks and their Applications”, *Spring Joint Computer Conference*, AFIPS Proc., vol. 32, 1968, pp. 307-314.

[3] M. Ajtai, J. Komlos, and E. Szemerdi, “Sorting in nlogn Steps”, *Combinatorica*, Vol. 3, 1983, pp. 1-19.

[4] Batcher, K. E., Course Notes on Sorting Networks, Special Topics: Sorting Networks, Department of Computer Science, Kent State University, 2005.

[5] Knuth, D. E., The Art of Computer Programming: Volume 3: Sorting and Searching, Addison-Wesley Longman, USA, 2nd Edition, 1998, Chapter 5, pp. 225-228.

[6] Rosen, K. H., Discrete Mathematics and its Applications, McGraw-Hill Companies, USA, 5th Edition, 2003, Chapter 7, pp. 520-525.