On Fast Bitonic Sorting Networks

Tamir Levi∗ Ami Litman†

January 1, 2009

Abstract

This paper studies fast Bitonic sorters of arbitrary width. It constructs such a sorter of width $n$ and depth $\lceil \log(n) \rceil + 3$, for any $n$ (not necessarily a power of two).

Keywords: Bitonic Sorting, Merging, Comparator Networks, Zero-One Principle, Min Max Networks

∗Faculty of Computer Science, Technion, Haifa 32000, Israel. levyt@cs.technion.ac.il
†Faculty of Computer Science, Technion, Haifa 32000, Israel. litman@cs.technion.ac.il
1 Introduction

This paper studies fast Bitonic sorters\(^\dagger\) of arbitrary width and our main result is such a sorter of width \(n\) and depth \([\log(n)] + 3\), for any \(n\) (not necessarily a power of two).

Most of the work on key processing networks is under the well-known comparator model. This paper focuses on a different model of computation – the Min-Max model, presented in [5]. This model is derived from Monotone Boolean Circuits [2]; the latter processes binary values while the former processes keys - members of some ordered set. To this end, AND-gates are replaced by \textit{MIN-gates} and OR-gates are replaced by \textit{MAX-gates}. The Min-Max model has an additional restriction – the indegree of all gates is exactly two. By [5], the Min-Max model sustains all the known variants of the 0-1 Principle. Furthermore, for some variants it is the strongest model of computation that sustains them. Therefore, we find the Min-Max model more natural than the comparator model.

A survey of related work is presented shortly. Efficient Bitonic sorters for certain special cases are already known. The famous examples are Batcher’s Bitonic sorters [1] whose width are powers of two. However, prior to this work, the only known technique to construct Bitonic sorters of arbitrary width is due to Liszka and Batcher [9]. For odd \(n\), their technique produces Bitonic sorters of width \(n\) and of depth \(2 \left\lceil \log(n) \right\rceil - 1\) comparators. As said, the depth of our construction is \([\log(n)] + 3\) gates. Namely, our networks are almost twice as fast compared to prior ones.

On the face of it, a Min-Max network may have much more computational elements than a comparator network of the same depth and width. (In the extreme case, a Min-Max network may have a disjoint sub-network for each of its outputs.) However, this does not occur in our fast Bitonic sorters – their depth is bounded by \((\lceil \log(n) \rceil + 3)\) and they have at most \(n \cdot (\lceil \log(n) \rceil + 3)\) gates.

We believe that this work is interesting, not only due to its results, but also due to its techniques. This paper is built on the technique of Nakatani et al. [13] as follows. Let \(B'\) and \(B''\) be two Bitonic sorters of width \(n'\) and \(n''\) and of depth \(d'\) and \(d''\), respectively. Nakatani et al. show how to combine several copies of \(B'\) and \(B''\) into a Bitonic sorter of width \(n' \cdot n''\) and of depth \(d' + d''\). We refer to their elegant technique as \textit{Bitonic-Multiplication}.

We present a similar technique called \textit{Bitonic-Addition}. In contrast to Bitonic-Multiplication, which is applicable both in the comparator model and in the Min-Max model, the Bitonic-Addition technique is applicable only in the Min-Max model. For the above \(B'\) and \(B''\), we combine a single copy of \(B'\), a single copy of \(B''\) and a network of depth at most 2 into a Bitonic sorter of width \(n' + n''\) and of depth \(\max(d', d'') + 2\) or smaller. The family of \textit{Add-Multiply} Bitonic sorters, presented in Section 4, is generated by the above two construction techniques starting with the two trivial Bitonic sorters of width 1 and 2. Our main result of fast Bitonic sorters is due to members of this family.

The Bitonic-Addition technique is based on the ‘divide and conquer’ paradigm. Namely, the input Bitonic sequence of length \(n\) is statically (without any gates) partitioned into two Bitonic sequences of length \(n'\) and \(n''\) which are “smoothly interleaved” (as defined in Section 3). The concept of “smoothly interleaved” sequences is based on the idea of “Smooth sets” which capture the idea of an “evenly distribute set of integers”. These sets and their relevance to scheduling were investigated by Litman and Moran [10, 11, 12]. This paper demonstrates that this concept is useful also to oblivious algorithms.

Another technique, demonstrated in this paper, is the idea of \textit{conclusive sets} [3]. Our conclusive set is a small and elegant set of vectors such that every network that sorts all the members of this set is a Bitonic sorter. Traditionally, key processing networks are analyzed via (some variant of) the 0-1 Principle. However, as this paper demonstrates, it is sometimes better to use non-binary conclusive sets.

Other results of this paper concern lower bounds on the depth of Min-Max Bitonic sorters. In this context we observe the following abnormal phenomena. For many key processing problems (such as sorting, merging or insertion) the smaller the input width the easier the problem. Surprisingly, this is not

\(^\dagger\)A Bitonic sequence is a rotation of a concatenation of two sequences – an ascending sequence followed by a descending one. A Bitonic sorter of width \(n\) is an acyclic network that sort any Bitonic sequence of length \(n\).
the case in Bitonic sorting; sometimes, reducing the width of the input sequence may force an increase on the depth of the Bitonic sorter. Specifically, we prove that, for infinitely many \( n \), there is an \( n' \) smaller than \( n \) such that sorting Bitonic sequences of length \( n' \) requires a Min-Max network of a greater depth compared to sorting Bitonic sequences of length \( n \).

The above abnormality w.r.t. Min-Max networks implies the same abnormality w.r.t. comparator networks. In fact, this phenomena is even more widespread in the comparator model. As shown in [7], for any \( n \) that is not a power of two, the minimal depth of a Bitonic sorter in the comparator model is at least \( \lceil \log(n) \rceil + 1 \). This last result does not hold in the Min-Max model. We show, in section 5, that for infinitely many \( n \), which are not powers of two, there are Min-Max Bitonic sorters of width \( n \) and of depth \( \lceil \log(n) \rceil \). (These networks are not Add-Multiply Bitonic sorters.). These networks demonstrate that some tasks\(^2\) can be solved in the Min-Max model faster than in the comparator model.

1.1 Preliminaries

Our networks process keys which are members of some ordered set, denoted by \( K \). The exact nature of keys is usually not important but for definiteness we choose \( K = \mathbb{Q} \). A sequence \( v \) of \( n \) keys, is denoted by \( v = (v_1, v_2, \ldots, v_n) \). Note that the first element of \( v \) is \( v_1 \) rather than \( v_0 \). We refer to \( n \) as the width of \( v \) and denote it by \( |v| \). Such a sequence is ascending-descending if it is a concatenation of an ascending sequence followed by a descending sequence. A sequence is Bitonic\(^3\) if it is a cyclic rotation of an ascending-descending sequence.

As said, this paper discusses two classes of networks that process keys. The in-degree of a computational element in both classes is exactly two. However, these classes differ w.r.t. the out-degree of a computational element; in one class the out-degree is restricted to be exactly two while in the other class the out-degree is unrestricted. We next present these two models.

The main class addressed by this paper are the Min-Max networks [5] which are henceforth referred to as \( m\text{-networks} \). Their basic computational elements are MIN-gates and MAX-gates. These gates compute the minimum and maximum of two keys, respectively. An \( m\text{-network} \) is an acyclic network of

\[ b_1 \quad b_2 \quad b_3 \quad b_4 \quad b_5 \]

\[ o_1 \quad o_2 \quad o_3 \quad o_4 \quad o_5 \]

Figure 1: A Bitonic m-sorter of width 5 and depth 4.

\(^2\)Another such task, due to Knuth [4], is insertion of a single key into a sorted sequence. Furthermore, as shown in [5], there are some tasks that can be solved only in the Min-Max model.

\(^3\)The term ‘Bitonic’ was coined by Batcher [1] and we follow his terminology. We caution the reader that some authors use the same term with other meanings.
such gates\footnote{As said, m-networks are a natural generalization of Monotone Boolean Circuits \cite{2}. However, Monotone Boolean Circuits may have gates that produce the constant values 0 and 1 while m-networks have no such gates.}; see Figure 1. In this figure solid triangles denote MAX-gates and hollow triangles denote MIN-gates. Keys enter an m-network via \textit{input ports} and exit the network via \textit{output ports}. These ports are depicted by solid circles. Usually, the input and the output of such a network are sequences. (These two sequences are not necessarily of the same width.) The network specifies, in some form, how the input sequence is fed to the input ports and how the output sequence is assembled from the output ports.

A \textit{Bitonic m-sorter of width $n$} is an m-network that sorts every Bitonic sequence of width $n$; the m-network of Figure 1 is a Bitonic m-sorter of width 5. Establishing the last statement via the 0-1 Principle requires verifying all the 20 non-constant binary Bitonic vectors of width 5. As shown in Section 3, the same can be done via only 5 Bitonic vectors.

The second class of networks addressed by this paper are the well-known comparator networks which are henceforth referred to as \textit{c-networks}. A \textit{comparator} is a combinational device that sorts two keys. It sends the minimal key on exactly one of its outgoing edges, called the \textit{MIN-edge}, and sends the maximal key on the other outgoing edge, called the \textit{MAX-edge}. A \textit{comparator network} is an acyclic network of comparators that satisfies the above fanout restriction. See Figure 2. In this figure, a solid arrowhead denotes a MAX-edge and a hollow arrowhead denotes a Min-edge. As usual, keys enter (exit) an c-network via \textit{input ports (output ports)}. The fanout of an input port is exactly one. (There is no such restriction in m-networks.) A \textit{Bitonic c-sorter of width $n$} is a c-network that sorts every Bitonic sequence of width $n$; the c-network of Figure 2 is a Bitonic c-sorter of width 5. Again, as shown in Section 3, 5 vectors suffice to verify that this network is, indeed, a Bitonic c-sorter.

![Figure 2: A Bitonic c-sorter of width 5 and depth 4.](image)

Clearly, every c-network can be translated into an m-network by replacing every comparator with two gates – a MIN-gate and a MAX-gate. The opposite translation from an m-network to a c-network does not work. A gate can be replaced by a comparator. However, one of the values produced by this comparator is discarded while the other may be transmitted to several destinations. The resulting network is not a valid c-network.

Let $N$ be a network (either a c-network or an m-network). The \textit{input width} of $N$, denoted $|N|^I$, is the number of input ports of $N$. Similarly, the \textit{output width} of $N$, denoted $|N|^O$, is the number of output ports of $N$. If $|N|^I = |N|^O$ then we call this number the \textit{width} of $N$ and denote it by $|N|$. By the fanout restriction of c-networks, the above equality holds for every c-network. On the other hand, m-networks do not necessarily satisfy this constraint; still, this paper focus on m-networks which do.

Let $e$ be an edge of a c-network (an m-network) $N$. The \textit{input depth} of $e$, denoted $d^I(e)$, is the maximal number of comparators (gates) along a path from an input port to $e$, not including the comparator (gate) at the end of $e$ (if there is such a comparator). Similarly, $d^O(e)$ denotes the maximal number of comparators (gates) along a path from $e$ to some output port, not including a comparator (gate) at the beginning of $e$. The \textit{depth} of $N$, denoted $d(N)$, is the maximal depth of its edges. Note that the depth of $N$ may be zero; in this case $N$ has no comparators or gates.
1.2 Related work

All prior works on Bitonic sorters address comparator networks. We now summarize what is known about the minimal depth, let call it $D^C(n)$, of a Bitonic c-sorter of width $n$. We first consider lower bounds on $D^C(n)$. By a straightforward reachability argument, for every $n$,

$$D^C(n) \geq \lceil \log(n) \rceil$$  \hspace{1cm} (1)

Inequality (1) is usually not tight. As shown in [7], for any $n$ that is not a power of two, the following holds.

$$D^C(n) \geq \lceil \log n \rceil + 1$$  \hspace{1cm} (2)

We next consider upper bounds on $D^C(n)$. Batcher’s [1] well-known Bitonic sorter is based on a network of depth one that splits every Bitonic sequence of width $2n$ into two Bitonic sequences of width $n$. The first contains the lower keys and the second contains the higher keys; therefore,

$$D^C(2 \cdot j) \leq D^C(j) + 1$$  \hspace{1cm} (3)

This, the fact that $D^C(2) = 1$, and Inequality (1) imply:

$$D^C(2^j) = j$$  \hspace{1cm} (4)

Our technique for fast Bitonic sorting is built upon the work of Nakatani et al. [13] which is detailed in Section 2. They show that for every $i$ and $j$

$$D^C(i \cdot j) \leq D^C(i) + D^C(j)$$  \hspace{1cm} (5)

Batcher and Liszka [9] introduced a recursive construction of Bitonic c-sorters of general width. Their construction partitions a Bitonic sequence of length $n$ into two Bitonic subsequences of length $\lceil \frac{n}{2} \rceil$ and $\lfloor \frac{n}{2} \rfloor$ and sorts each of them recursively. The two resulting sorted sequences are then merged into a single sorted sequence. This merge requires a depth one network when $n$ is even and a depth two network when $n$ is odd. Therefore,

$$D^C(2n + 1) \leq \max(D^C(n), D^C(n + 1)) + 2$$  \hspace{1cm} (6)

This implies that for every $n$,

$$D^C(n) \leq 2 \lceil \log(n) \rceil - 1$$  \hspace{1cm} (7)

We next address m-networks. Let $D^M(n)$ denote the minimal depth of a Bitonic m-sorter of width $n$. By a straightforward reachability argument, for every $n$,

$$D^M(n) \geq \lceil \log(n) \rceil$$  \hspace{1cm} (8)

As said, Inequality (2) holds for every $n$ that is not a power of two. The same does not hold for m-networks. Namely, Lemma 17 shows that $D^M(n) = \lceil \log n \rceil$ holds for infinitely many $n$ that are not powers of two.

The previous works that yield Inequalities (3),(4) and (5) address comparator networks. However, the same constructions hold also for m-networks. Therefore, we have:

$$D^M(2 \cdot j) \leq D^M(j) + 1$$  \hspace{1cm} (9)

$$D^M(2^j) = j$$  \hspace{1cm} (10)

$$D^M(i \cdot j) \leq D^M(i) + D^M(j)$$  \hspace{1cm} (11)

For arbitrary $n$, the exact values of $D^M(n)$ and $D^C(n)$ are unknown. Moreover, as discussed in Section 5, all prior techniques, including our Add-Multiply technique, fail to produce Bitonic sorters of minimal depth for infinitely many widths. This holds even when these techniques are mixed during the recursion process.
2 Multiplying Bitonic sorters

Our construction of Bitonic sorters is based on two binary operators on Bitonic sorters. The first operator, referred to as ‘multiplication’ and denoted by ‘∗’, was introduced by Nakatani et al. [13]. For any two Bitonic c-sorters (m-sorters), \( B' \) and \( B'' \), this operator generates a Bitonic c-sorter (m-sorter) of width \(|B'| \cdot |B''|\). The name ‘multiplication’ refers to the width the networks in question. This operator is based on the following lemma.

**Lemma 1 (Nakatani et al. [13])** Let \( b \) be a Bitonic sequence of \( j \cdot k \) keys arranged in a \( j \times k \) matrix \( M \) in a row major fashion. Then:
1) Every column in \( M \) is a Bitonic sequence.
Let \( M' \) the matrix derived from \( M \) by sorting each column separately. Then:
2) Every row in \( M' \) is a Bitonic sequence.
3) Every element of a row of \( M' \) is smaller or equal to any element of the next row.

In order to describe networks generated by the above lemma, we use the following notation. Let \( A, B \) and \( C \) be three c-networks (m-networks).

- We say that \( C \) is of the form \( \begin{pmatrix} A \\ B \end{pmatrix} \) if \( C \) is composed of two parallel (disjoint and unconnected) networks; one is a copy of \( A \) and the other is a copy of \( B \).
- We say that \( C \) is of the form \((nA)\) if \( C \) is composed of \( n \) parallel (disjoint and unconnected) copies of the network \( A \).
- We say that \( C \) is of the form \((AB)\) if \( C \) is a composed of two networks connected in serial, a copy of \( A \) followed by a copy of \( B \). An output port of the first network is merged with a unique input port of the second network. This form implies that \(|A|^O = |B|^I\).

Let \( A, B \) and \( C \) be three c-networks (m-networks). Then the following statements trivially hold:

1. If \( C \) is of the form \( \begin{pmatrix} A \\ B \end{pmatrix} \) then \(|C|^O = |A|^O + |B|^O, |C|^I = |A|^I + |B|^I\) and \(d(C) = \max(d(A), d(B))\).
2. If \( C \) is of the form \((nA)\) then \(|C|^O = n \cdot |A|^O, |C|^I = n \cdot |A|^I\) and \(d(C) = d(A)\).
3. If \( C \) of the form \((AB)\) then \(|C|^O = |B|^O, |C|^I = |A|^I\), and \(d(C) \leq d(A) + d(B)\)

Lemma 1 has the following corollary

**Lemma 2 (The Bitonic multiplying lemma [13])** Let \( B' \) and \( B'' \) be two Bitonic c-sorters (m-sorters) and let \( n' = |B'| \) and \( n'' = |B''| \). Then there is a Bitonic c-sorter (m-sorter) \( B \) of width \( n' \cdot n'' \) and depth \( d(B) = d(B') + d(B'') \) of the form \((n''B')(n'B'')\).

The binary operator ‘∗’ is defined via Lemmas 1 and 2. Namely, \( B' \ast B'' \) is the c-network (m-network), \( B \), constructed according to these lemmas.

3 Adding Bitonic Sorters

This section presents the Bitonic-Addition Operator, denoted ‘+’. This operator is defined only for m-networks and satisfies \(|B' + B''| = |B'| + |B''|\) and \(d(B' + B'') \leq \max(d(B'), d(B'')) + 2\).
3.1 Circular vectors

To present the `+' operator we utilize a new manner to organize keys. Traditionally, networks receive and produce sequences of keys. In many cases it is preferable to consider the keys to have a different structure. For example, in Lemma 1, keys are considered to be arranged in a matrix. To this end, we pick a finite set, \( X \), called the index set. A vector \( v \) over \( X \) is a function \( v : X \rightarrow K \). We denote the elements of \( v \) by \( v_x \), rather than \( v(x) \), and the term `index' is due to this notation. For example, for \( X = \{1, 2, \ldots, n\} \), a vector over \( X \) is the familiar sequence of \( n \) keys.

![Figure 3: Three circular vectors.](image)

To enable a network to receive (to produce) vectors over an index set \( X \), the network has some (usually implicit) bijection from the input ports (output ports) onto \( X \); this bijection specifies how the input is fed into (output is collected from) the network.

A sequential key structure is, by nature, unsymmetrical; it has a first member and a last member. In the context of Bitonic sequences, this lack of symmetry is artificial, since Bitonic sequences (by definition) are closed under rotations. Hence, it is preferred that the keys are organized in a structure that is of a circular nature – a structure that has no first element or last element. To formalize this idea we pick, for every \( n \), some directed graph \( G^n = (C^n, E^n) \) which is a cycle of \( n \) vertices. The exact nature of the vertices is not important. The set \( C^n \) serves as the index set for Bitonic vectors. Namely, a circular vector of width \( n \) is a function \( v : C^n \rightarrow K \). In contrast to a sequence, a circular vector has no first or last elements – all elements play the same role in this structure. Figure 3 depicts three such circular vectors.

A linearization of a circular vector \( v \) is a sequence of \(|v|\) keys as they appear along a spanning path of \( C^{|v|} \). A circular vector is Bitonic if it has a linearization that is ascending-descending. All three vectors of Figure 3 are Bitonic. By the same token, a circular vector is Unitonic if it has a linearization that is ascending. Unitonic vectors play a significant role in this paper which is summarized in Lemma 9. Vectors (b) and (c) of Figure 3 are Unitonic.

Assume we are given two Bitonic sorters, \( B' \) and \( B'' \) of width \( i' \) and \( i'' \), respectively. We construct a Bitonic m-sorter, denoted \( B' + B'' \), which operates as follows. The Bitonic input vector, of width \( i' + i'' \), is partitioned in a special manner into two Bitonic vectors, \( b' \) of width \( i' \) and \( b'' \) of width \( i'' \). This partition is static – it is done without gates. The vectors \( b' \) and \( b'' \) are sorted separately by \( B' \) and \( B'' \). The two resulting sorted vectors are merged by a combining network of width at most two.

The above partition is based on the notion of smooth sets, introduced by Even, Lincoln and Cohn [8] and further investigated by Litman and Moran [10]. A smooth set of integers has the property that its quantity within any interval is proportional to the size of the interval, up to a bounded additive deviation. Namely, a set of integer \( A \) is \((\rho, \Delta)\)-smooth if \( |I| : \rho - |I \cap A| < \Delta \), for any interval \( I \). In this work we focus on \((\rho, 1)\)-smooth sets of integers. Such sets capture the idea of a set of integers which is “as evenly distributed as possible”. For example, for \( j \in \mathbb{N} \), \( A \) is a \((\frac{1}{j}, 1)\)-smooth if and only if \( A = j \cdot \mathbb{Z} + i \) for some \( i \in \mathbb{Z} \).

This paper demonstrates that the idea of ‘smoothness’ is relevant to oblivious sorting algorithms. Let \( A \), \( B \) and \( C \) be finite sets and \( B \subseteq C \). We say that \( A \) is smooth w.r.t. \( B \) and \( C \) if the proportion of
$A \cap B$ in $B$ is very close to the proportion of $A \cap C$ in $C$. Namely, we say that $A$ is smooth w.r.t. $(C, B)$ if $|A \cap C| \cdot \frac{|B|}{|C|} - |A \cap B| < 1$. In the context of $C^n$, the concept of interval is naturally replaced with the concept of segment – a set of consecutive vertices of $C^n$. This leads to the following definition.

**Definition 3** A partition $(C', C'')$ of $C^n$ is smooth if the following conditions hold.

1. The set $C'$ is smooth w.r.t. $(C^n, S)$, for every segment $S$ of $C^n$.
2. The set $C''$ is smooth w.r.t. $(C^n, S)$, for every segment $S$ of $C^n$.

Actually, the above two conditions of Defn. (3) are equivalent – each of them implies the other.

Smooth partitions of $C^n$ were not addressed in [10] but it is not hard to translate their results into the context of partitions of $C^n$. Such a translation provides the following lemma.

**Lemma 4** Let $i', i'' \in \mathbb{N} + 1$ and let $n = i' + i''$. Then there is a smooth partition $(C', C'')$ of $C^n$ such that $|C'| = i'$, $|C''| = i''$.

In fact, by [10], the smooth partition as per Lemma 4 is unique, up to an automorphism of $G^n$. Figure 4 depicts a smooth partition $(C', C'')$ of $C^{16}$ such that $|C'| = 10$ and $|C''| = 6$. The set $C'$ is composed of the solid bullets and the set $C''$ is composed of the hollow bullets.

![Figure 4: A smooth partition of $C^{16}$](image)

The next lemma shows that certain Bitonic vectors can be partitioned into two vectors that are ‘interleaved in a smooth manner’, as defined shortly. To formalize this discussion, we use the following notation. For a vector $v$, let $\{v\}$ denote the range of $v$; namely, the set of keys that appear in $v$. A permutation of width $n$ is a vector $v$ of width $n$ such that $\{v\} = \{1, 2, \ldots, n\}$. Namely, every key in this interval appears exactly once in $v$.

**Definition 5** Two vectors, $v'$ and $v''$, are smoothly interleaved if the followings hold:

1. The vector $v = (v', v'')$ is a permutation.
2. The set $\{v'\}$ is smooth w.r.t. $(\{v\}, I)$ for every $I$, sub-interval of $\{v\}$.
3. The set $\{v''\}$ is smooth w.r.t. $(\{v\}, I)$ for every $I$, sub-interval of $\{v\}$.

For example, the vectors $v' = \langle 1, 3, 4, 6, 7, 9, 11, 12, 14, 15 \rangle$ and $v'' = \langle 2, 5, 8, 10, 13, 16 \rangle$ are smoothly interleaved. Actually, Conditions (2) and (3) of Defn. (5) are equivalent – each of them implies the other one.

**Lemma 6** Let $(C', C'')$ be a smooth partition of $C^n$ and let $u$ be a Unitonic permutation of width $n$. Let $u'$ and $u''$ be the partition of $u$ induced by $(C', C'')$. Then $u'$ and $u''$ are smoothly interleaved.
Proof: Conditions (1) of Defn. (5) clearly hold. As said, Conditions (2) and (3) are equivalent, so it suffices to prove only one of them. We prove Condition (2); namely, that \( \{ u^t \} \) is smooth w.r.t. \( \{ u \}, I \), for every sub-interval \( I \) of \( \{ u \} \).

Clearly, \( |\{ u^t \}| = |C'| \). Let \( I \) be a sub-interval of \( \{ u \} \) and let \( S = \{ c \in C^n | u_c \in I \} \). Since \( u \) is a Unitonic permutation, \( S \) is a segment of \( C^n \) and \( |S| = |I| \). Clearly, \( |C' \cap S| = |\{ u^t \} \cap I| \). By definition, \( C' \) is smooth w.r.t. \( (C^n, S) \). Hence, \( |C'| \cdot \frac{|S|}{n} - |C' \cap S| < 1 \). Applying the above equations yields, \( |\{ u^t \}| \cdot \frac{|I|}{n} - |\{ u^t \} \cap I| < 1 \). This holds for every sub-interval \( I \) of \( \{ u \} \), establishing Condition (2).

Note that Lemma 6 refers only to Unitonic permutations and it does not hold for general Bitonic vectors. We will later overcome this weakness.

For a sequence \( v \) and two indexes \( i, j \in [1, |v|] \), let \( v[i, j] \) denote the subsequence of \( v \) composed of all elements from position \( i \) until position \( j \), inclusively. Since the first element of \( v \) is \( v_1 \) (rather than \( v_0 \)), \( v[1, |v|] = v \). The following lemma provides substantial information on the relative order of keys in a pair of sorted sequences that are smoothly interleaved.

**Lemma 7** Let \( v' \) and \( v'' \) be two sorted sequences that are smoothly interleaved. Let \( \rho' = \frac{|v'|}{|v'| + |v''|} \) and \( \rho'' = \frac{|v''|}{|v'| + |v''|} \) and let \( 1 \leq k \leq |v'| + |v''| \). Then:

\[
\{ v'[1, [k \cdot \rho']} \} \cup \{ v''[1, [k \cdot \rho'']] \} \subset [1, k] \subset \{ v'[1, [k \cdot \rho']} \} \cup \{ v''[1, [k \cdot \rho'']] \}
\]

**Proof:** Let \( I = [1, k] \). By Condition (2) of Defn. (3), \(|I| \cdot \rho' - |I \cap \{ v' \}| < 1 \). This and \(|I| = k\) imply that,

\[
[k \cdot \rho'] \leq |I \cap \{ v' \}| \leq [k \cdot \rho']
\]

Since \( v' \) is sorted, it follows that

\[
\{ v'[1, [k \cdot \rho']] \} \subset I \cap \{ v' \} \subset \{ v'[1, [k \cdot \rho']] \}
\]

Our construction is symmetric w.r.t. to \( v' \) and \( v'' \); hence, converting the last expression from \( v' \) to \( v'' \) yields:

\[
\{ v''[1, [k \cdot \rho'']] \} \subset I \cap \{ v'' \} \subset \{ v''[1, [k \cdot \rho'']] \}
\]

The set union of the above two expressions satisfies:

\[
\{ v'[1, [k \cdot \rho']] \} \cup \{ v''[1, [k \cdot \rho'']] \} \subset (I \cap \{ v' \}) \cup (I \cap \{ v'' \}) = I = [1, k] \subset \{ v'[1, [k \cdot \rho']] \} \cup \{ v''[1, [k \cdot \rho'']] \}
\]

and the lemma follows.

The following lemma enables us to construct the desired combining network of depth two. For this discussion we consider bissequenced vectors – vectors of the form \( v = \langle v', v'' \rangle \) where \( v' \) and \( v'' \) are two sequences. We construct an index set for bissequenced vectors as follows; we pick two infinite sequences whose elements will serve as indexes. The nature of these elements is not important as long as each element appears in the two sequences exactly once. The first sequence is denoted \( \langle 1', 2', 3', \ldots \rangle \) and the second sequence is denoted \( \langle 1'', 2'', 3'', \ldots \rangle \). For every \( s, t > 0 \), define \( X^{s, t} = \{ 1', 2', \ldots, s', 1'', 2'', \ldots, t'' \} \). The set \( X^{s, t} \) serves as an index set for bissequenced vectors composed of two sequences of width \( s \) and \( t \), in the natural manner.
We also add two fictive indexes, denoted by $\sim -\infty$ and $\sim +\infty$, which are not members of any index set. We extend every vector $v$ to be defined over these fictive indexes by $v_{-\infty} = -\infty$ and $v_{+\infty} = +\infty$. The two values $-\infty$ and $+\infty$ are fictive keys, the former is smaller and the latter is greater than all real keys.

**Lemma 8** Let $k, n', n'' \in \mathbb{N}^+1$ and $1 \leq k \leq n' + n''$. Then there are three indexes $x, y, z \in X^{n'n''} \cup \{-\infty, +\infty\}$ such that the following holds. For every Bisequenced vector $v = (v', v'')$, if:

- $|v'| = n'$, $|v''| = n''$.
- $v'$ and $v''$ are sorted.
- $v'$ and $v''$ are smoothly interleaved

then $k$ is the median of the triple $(v_x, v_y, v_z)$.

Table 1 presents the three indexes associated with each $k$ as per Lemma 8 for $n' = 10$ and $n'' = 6$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$x, y, z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$\sim -\infty, 1', 1''$</td>
</tr>
<tr>
<td>2</td>
<td>$1', 2', 1''$</td>
</tr>
<tr>
<td>3</td>
<td>$2', 1'', 2''$</td>
</tr>
<tr>
<td>4</td>
<td>$2', 3', 2''$</td>
</tr>
<tr>
<td>5</td>
<td>$3', 4', 2''$</td>
</tr>
<tr>
<td>6</td>
<td>$4', 2'', 3''$</td>
</tr>
<tr>
<td>7</td>
<td>$4', 5', 3''$</td>
</tr>
<tr>
<td>8</td>
<td>$\sim +\infty, 5', 3''$</td>
</tr>
<tr>
<td>9</td>
<td>$\sim -\infty, 6', 4''$</td>
</tr>
<tr>
<td>10</td>
<td>$6', 7', 4''$</td>
</tr>
<tr>
<td>11</td>
<td>$7', 4'', 5''$</td>
</tr>
<tr>
<td>12</td>
<td>$7', 8', 5''$</td>
</tr>
<tr>
<td>13</td>
<td>$8', 9', 5''$</td>
</tr>
<tr>
<td>14</td>
<td>$9', 5'', 6''$</td>
</tr>
<tr>
<td>15</td>
<td>$9', 10', 6''$</td>
</tr>
<tr>
<td>16</td>
<td>$\sim +\infty, 10', 6''$</td>
</tr>
</tbody>
</table>

Table 1: The three indexes as per Lemma 8 for the case of $n' = 10$ and $n'' = 6$.

**Proof:** Let $\rho' = \frac{n'}{n' + n''}$, $\rho'' = \frac{n''}{n' + n''}$. We consider four disjoint and conclusive cases:

**Case 1** $\rho' \cdot k \in \mathbb{Z}$:

This condition is equivalent to $\rho'' \cdot (k - 1) \in \mathbb{Z}$. Let $v'$ and $v''$ be two sorted vectors that are smoothly interleaved. By Lemma 7, $\{v'[1, k \cdot \rho'] \} \cup \{v''[1, k \cdot \rho''] \} = [1, k]$. Therefore, $k = \max(\{v'[1, k \cdot \rho'] \} \cup \{v''[1, k \cdot \rho''] \})$. Since $v'$ and $v''$ are sorted, $k = \max(v'_k, v''_k)$; therefore, $k$ is the median of $\{v'_k, v''_k, v_{\sim +\infty}\}$.

**Case 2** $\rho' \cdot (k - 1) \in \mathbb{Z}$:

This condition is equivalent to $\rho'' \cdot (k - 1) \in \mathbb{Z}$. Let $v'$ and $v''$ be two sorted vectors that are smoothly interleaved. By the proof of the previous case, $[1, k - 1] = \{v'[1, (k - 1) \cdot \rho'] \} \cup \{v''[1, (k - 1) \cdot \rho''] \}$. Hence, $[k, n' + n''] = \{v'[[k \cdot \rho'], n'] \} \cup \{v''[[k \cdot \rho''], n''] \}$; that is, $k = \min(\{v'[[k \cdot \rho'], n'] \} \cup \{v''[[k \cdot \rho''], n''] \})$. Since $v'$ and $v''$ are sorted, $k = \min(v'_{[\rho', k]}, v''_{[\rho', k]})$; therefore, $k$ is the median of $\{v_{\sim -\infty}, v'_{[\rho', k]}, v''_{[\rho', k]} \}$

**Case 3** $\rho' \cdot k < \lceil \rho' \cdot (k - 1) \rceil$:

Clearly, $\rho' + \rho'' = 1$. This and $\rho' \cdot k \notin \mathbb{Z}$ imply

$$k = \rho' \cdot k + \rho'' \cdot k < \lceil \rho' \cdot k \rceil + \lfloor \rho'' \cdot k \rfloor < k + 2$$

And since $\lfloor k \cdot \rho' \rfloor$ and $\lceil k \cdot \rho'' \rceil$ are integers,

$$\lfloor \rho' \cdot k \rfloor + \lceil \rho'' \cdot k \rceil = k + 1 \quad (12)$$

Let $v = (v', v'')$ where $v'$ and $v''$ are two sorted sequences that are smoothly interleaved. We next show that $k \in \{v'_{[\rho', k]}, v''_{[\rho', k]}, v'_{[\rho'' \cdot k]}, v''_{[\rho'' \cdot k]} \}$. By Lemma 7,

$$[1, k] \subset \{v'[1, \lfloor k \cdot \rho' \rfloor]\} \cup \{v''[1, \lceil k \cdot \rho'' \rceil]\} \quad (13)$$
By Equation 12, the set in the right side of expression 13 has exactly \( k + 1 \) keys; therefore, this set contains the interval \([1, k]\) and exactly one additional key and this key is greater than \( k \). Hence, \( k \) is the second largest key in this set. Moreover, since \( v' \) and \( v'' \) are sorted, \( k \) is the second largest key out of the four keys \( \{v'_{[\rho', k]}, v'_{[\rho', k]}, v''_{[\rho''', k]}, v''_{[\rho''', k]}\} \).

Next, we show that \( k \neq v'_{[\rho', k]} \). The condition of this case implies that \( [\rho' \cdot (k - 1)] = [\rho' \cdot k] \); therefore, by Lemma 7,

\[
\{ v'[1, [\rho' \cdot k]] \} = \{ v'[1, [\rho' \cdot (k - 1)]] \} \subset [1, k - 1]
\]

This establishes that \( v'_{[\rho', k]} \neq k \) as promised. Hence, \( k \) is the second largest key in the remaining three keys. In other words, \( k \) is the median of \( \{v'_{[\rho', k]}, v''_{[\rho'', k]}, v''_{[\rho''', k]}\} \).

**Case 4** \( \rho'' \cdot k < [\rho'' \cdot (k - 1)] \):

Our construction is symmetric w.r.t. \( v' \) and \( v'' \); hence, this case is symmetric to case (3).

We next show that the above four cases are conclusive. If Case (2) does not hold then the following variant of Equation 12, for \( k - 1 \) instead of \( k \), holds.

\[
[\rho' \cdot (k - 1)] + [\rho'' \cdot (k - 1)] = k
\]

Since \( \rho' + \rho'' = 1, \rho' \cdot k + \rho'' \cdot k = k \); hence,

\[
\rho' \cdot k + \rho'' \cdot k = [\rho' \cdot (k - 1)] + [\rho'' \cdot (k - 1)]
\]

This implies:

\[
\rho' \cdot k \leq [\rho' \cdot (k - 1)] \quad \text{or} \quad \rho'' \cdot k \leq [\rho'' \cdot (k - 1)]
\]

Clearly, equality holds in one of the above expressions only if it holds in both of them and then Case (1) holds. Otherwise, either Case (3) holds or Case (4) holds; this proves that the above four cases are conclusive. As said, these cases are also disjoint, but this is not required for our proof.

Lemma 8 implies that two sorted sequences that are smoothly interleaved can be combined into a single sorted sequence by a m-network of depth two, as follows. By this lemma, each output key is the median of either three input keys or two input keys and a fictive key. In the latter case, the median is computed by a single gate.

Consider the former case of three real keys. Generally, computing the median of arbitrary three keys requires an m-network of depth 3. (The same holds for c-networks.) Fortunately, in our case, the relative order of two of those keys is given. Hence, the median can be computed by an m-network of depth two. In fact, there are two such m-networks. To make the construction unique, we pick one of them and use it exclusively.

Let us repeat how the network \( B' + B'' \) is constructed. Let \( n = |B'| + |B''| \). Lemma 4 provides a smooth partition \( (C', C'') \) of \( C^n \) such that \( |C'| = |B'| \) and \( |C''| = |B''| \). A Bitonic input vector \( b \) of width \( n \) is partitioned into two Bitonic vectors, \( b' \) and \( b'' \), according to the partition \( (C', C'') \) of \( C^n \). These two vectors are sorted separately by \( B' \) and \( B'' \) generating the sorted sequences \( s' \) and \( s'' \). For the time being, assume that \( b \) is a Unitonic permutation. By Lemma 6, \( s' \) and \( s'' \) are smoothly interleaved. By Lemma 8, \( s' \) and \( s'' \) can be merged by a combining network \( C \) of depth at most two. The resulting m-network is \( B' + B'' \). This construction implies that \( B' + B'' \) sorts all Unitonic vectors. It remains to show that, in fact, \( B' + B'' \) sorts all Bitonic vectors. To this end, we use the idea of conclusive sets [3] which, for the case at hand, is summarized in the following lemma.

**Lemma 9** [3] A network is a Bitonic m-sorter if and only if it sorts all Unitonic permutations.

There are exactly \( n \) Unitonic permutations of width \( n \). By Lemma 9, the Bitonic sorters of Figures 1 and 2 can be verified by the 5 Unitonic permutations of width 5.

The main construction of the section is summarized in the following lemma.
Lemma 10 (The Bitonic addition lemma) Let $B'$ and $B''$ be two Bitonic m-sorters. Then there is a Bitonic sorter, denoted $B' + B''$, of the form $\left( \begin{array}{c} B' \\ B'' \end{array} \right) C$ where $C$ is an m-network of depth at most two. Hence, $|B' + B''| = |B'| + |B''|$ and $d(B' + B'') \leq \max(d(B), d(B'')) + 2$.

4 Add-Multiply Bitonic m-sorters

This section studies the family of Bitonic m-sorters constructed recursively by the two operators ‘+$’ and ‘$\ast$’. It shows that this family has a member of width $n$ and depth at most $\lceil \log(n) \rceil + 3$, for any width $n$.

Definition 11 The family of Add-Multiply Bitonic m-sorters is defined recursively as follows:

- The following two trivial Bitonic m-sorters are Add-Multiply ones: $B_1$ is the Bitonic m-sorter of width 1 and depth 0 and $B_2$ is the Bitonic m-sorters of width 2 and depth 1.
- Let $B'$ and $B''$ be two Add-Multiply Bitonic m-sorters. Then $B' \ast B''$ and $B' + B''$ are Add-Multiply Bitonic m-sorters.

Clearly, for every $n$ there is an Add-Multiply Bitonic m-sorter of width $n$. Let $D_{AM}(n)$ be the minimal depth of such a m-network. Lemma 2 and 10 imply that:

Lemma 12 For every $i, j \in \mathbb{N}$

- $D_{AM}(i \cdot j) \leq D_{AM}(i) + D_{AM}(j)$
- $D_{AM}(i + j) \leq \max(D_{AM}(i), D_{AM}(j)) + 2$

The next observation is straightforward.

Lemma 13

- For every $n$, $D_{AM}(n) \geq \lceil \log(n) \rceil$.
- For every $j$, $D_{AM}(2^j) = j$.

The main result of this section is,

Theorem 14 For every $n$, $D_{AM}(n) \leq \lceil \log(n) \rceil + 3$.

Proof: We prove the lemma by induction on $n$. Clearly, the lemma holds for $n = 1$, so assume $n \geq 2$. Let $n = i \cdot 2^j + q$, where $i \in \{2, 3\}$ and $q < 2^j$. (This decomposition of $n$ is unique.) The case of $q = 0$ is simple so assume $q > 0$. By Lemma 12,

$D_{AM}(n) = D_{AM}(i \cdot 2^j + q) \leq \max(D_{AM}(i \cdot 2^j), D_{AM}(q)) + 2 = \max(D_{AM}(i) + D_{AM}(2^j), D_{AM}(q)) + 2$

It is not hard to see that $D_{AM}(3) = 3$; hence $D_{AM}(i) \leq 3$. By Lemma 13, $D_{AM}(2^j) = j$. Therefore,

$D_{AM}(i) + D_{AM}(2^j) \leq 3 + j$

The fact that $2 \cdot 2^j < n$ imply that:

$3 + j \leq \lceil \log(n) \rceil + 1$

Since $q < 2^j$, it follows that $\lceil \log(q) \rceil \leq j$. By the induction hypotheses,

$D_{AM}(q) \leq \lceil \log(q) \rceil + 3 \leq j + 3 \leq \lceil \log(n) \rceil + 1$

The above inequalities imply that:

$D_{AM}(n) \leq \lceil \log(n) \rceil + 3$
5 Bitonic m-sorters of minimal depth

This chapter studies the minimal depth of Bitonic m-sorters and establishes a non-trivial lower bound on this number. In this context we observe the following abnormal phenomena. For many key processing problems (such as sorting, merging or insertion) the smaller the input width the easier the problem. Surprisingly, this is not the case for Bitonic sorting, as demonstrated by the following lemma.

Lemma 15 For every odd \( n \) greater than one: \( D^M(n) \geq \lceil \log(n) \rceil + 1. \)

Proof: In fact, we prove a stronger claim. Namely, that computing the median of a Bitonic sequence of an odd width \( n \) requires an m-network of depth at least \( \lceil \log(n) \rceil + 1. \) Assume, for a contradiction, that \( N \) is such an m-network of depth \( \lceil \log(n) \rceil. \) Without loss of generality, we may assume that \( N \) is a layered graph – all maximal paths in \( N \) are of the same length. This is due to the fact that short paths can be extended by inserting ‘redundant’ gates that receive the same key twice. We may also assume that \( N \) is a balanced binary tree. This is due to the fact that any gate whose fanout is greater than one can be replaced by several copies of the same sub-network. None of these transformations change the depth of \( N \) and so we may assume that \( N \) is a balanced binary tree of depth \( \lceil \log(n) \rceil. \)

Such a balanced tree has exactly \( 2^\lceil \log(n) \rceil \) leaves. Therefore, there is an input port, \( p, \) whose fanout is at most one. Since \( N \) clearly has a path from every input port to its output port, the fanout of \( p \) is exactly one. Assume \( p \) enters a gate \( g \) and let \( q \) be the other input port that enters \( g. \) (The port \( q \) may enter additional gates.) We now focus on Bitonic input vectors that are permutations in which the median key (namely, \( \frac{n+1}{2} \)) enters \( p. \) Under such vectors, the gate \( g \) must transmit the median key on its outgoing edges. As shown shortly, there are two such input vectors, \( v', v'' \) with the following property. For every two indexes, \( x \) and \( y, \) \( v'_x < v'_y \) if and only if \( v''_x > v''_y. \) The gate \( g \) is either a MIN-gate or a MAX-gate; therefore, under one of these two vectors, the gate \( g \) does not transmit the median on its outgoing edges.

To present the above two vectors pick \( v' \) to be any Bitonic permutation in which the median key \( \left( \frac{n+1}{2} \right) \) enters \( p. \) Let \( v'' \) be derived from \( v' \) by inverting the order of the keys. Namely, any key \( k \) is replaced by \( n+1-k. \) Clearly \( v'' \) is a permutation. Since \( n \) is odd, \( \frac{n+1}{2} \) is a fixpoint of this transformation; hence, under \( v'' \) and \( v', \) the median enters the same input port \( p. \)

Lemma 15 implies the following abnormal phenomena. Let \( n \) be an odd integer greater than one and let \( n' = 2^\lceil \log(n) \rceil. \) By Lemma 15, \( D^M(n) > \lceil \log(n) \rceil. \) On the other hand, by the construction of Batcher [1], \( D^M(n') = \log(n'). \) That is, \( n' > n \) and \( D^M(n') < D^M(n). \) This abnormality w.r.t. Min-Max networks, and the straightforward translation of a c-network into an m-network, imply that the same abnormality holds w.r.t. comparator networks. In fact, by the next lemma proved in [7], this phenomena is even more widespread in the comparator model.

Lemma 16 ([7]) For every \( n \) that is not a power of two, \( D^C(n) \geq \lceil \log(n) \rceil + 1. \)

Lemma 16 does not hold in the Min-Max model. Namely,

Lemma 17 The equation \( D^M(n) = \lceil \log(n) \rceil \) holds for infinitely many \( n \) that are not powers of two.

Proof: Inequalities (10) and (11) imply that \( D^M(j \cdot 2^i) \leq D^M(j) + i, \) for every \( i \) and \( j. \) Hence, it is suffice to construct a single Bitonic m-sorter of the required property. We construct such a Bitonic m-sorter, referred to as \( B^{10}, \) of width 10 and depth 4.

To this end, we present two Bitonic m-sorters, \( B' \) and \( B'', \) both of width 10 and depth which is slightly larger than the desired one. Most of the output ports of both networks are of depth 4. Each network has some output ports of depth 5 but these are not the same output ports. Namely, for any \( i \in [1, 10], \) the depth of the \( i \)th output port of at least one of \( B' \) or \( B'' \) is 4. The network \( B^{10} \) is composed of 10 disjoint sub-networks of depth 4, one for each output port; each sub-network is selected
either from $B'$ or from $B''$. Note that this construction is valid only in the Min-Max model and not in the comparator model.

The two networks, $B'$ and $B''$, are generated by the Bitonic Multiplication operator of Nakatani et al. and the Bitonic m-sorter, let us call it $B^5$, of Figure 1. Our construction is based on the following property of $B^5$. All the output ports of $B^5$ have depth 3 except the 3’rd output port whose depth is 4. Let $B^2$ be the trivial Bitonic m-sorter of width 2 and depth 1. Consider the two Bitonic m-sorters, $B' \triangleq B^5 \ast B^2$ and $B'' \triangleq B^2 \ast B^5$. By Lemma 2,

$$B' \text{ is of the form } \begin{pmatrix} B^5 \\ B^5 \end{pmatrix} \begin{pmatrix} B^2 \\ B^2 \\ B^2 \end{pmatrix} \text{ and } B'' \text{ is of the form } \begin{pmatrix} B^2 \\ B^2 \\ B^2 \\ B^2 \end{pmatrix} \begin{pmatrix} B^5 \\ B^5 \end{pmatrix}.$$  

The forms of $B'$ and $B''$ imply that for each output port $p$ of $B'$ or $B''$, the depth of $p$ is 5 if and only if there is a path that ends at $p$ and pass through the 3’rd output port of a copy of $B^5$. Hence, by Lemma 1, the only output ports of $B'$ whose depth is 5 are the 5’th and the 6’th output ports. Similarly, the only output ports of $B''$ whose depth is 5 are the 3’rd and the 8’th output ports. As promised, these numbers are distinct.

Lemma 17 have two interesting corollaries. The first is that certain computations can be performed faster in the Min-Max model as compared to the comparator model. The second corollary is that all prior techniques for constructing Bitonic m-sorters, including our Add-Multiply technique, fail to produce optimal networks, for infinitely many widths.

The first corollary immediately follows from Lemmas 16 and 17.

**Theorem 18** For infinitely many $n$, $D^M(n) < D^C(n)$.

Theorem 18 implies that certain tasks can be performed in the Min-Max model faster than in the comparator model. However, we do not know whether the gap $D^C(n) - D^M(n)$ is bounded or not.

Consider the second corollary of Lemma 17. All prior construction techniques and our Add-Multiply technique are recursive methods which construct wider Bitonic sorters out of smaller ones. The recursion naturally stops at Bitonic sorters of width 1 or 2. We next show that all these techniques do not construct Bitonic m-sorters of width $n$ and minimal depth, for infinitely many $n$. Moreover, this holds even when these techniques are mixed during the recursion process.

By definition, the Add-Multiply technique includes the technique of Nakatani et al. Moreover, close examination reveals that the techniques of Batcher [1] and of Batcher and Liszka [9] are special cases of the Add-Multiply technique. Therefore, it is suffice to show the above claim only for the Add-Multiply technique.

Actually, we prove a stronger claim by induction. Namely, that $D^{AM}(n) \geq \lceil \log(n) \rceil + 1$, for every $n$ that is not a power of two. Let $n$ be as above and let $B$ be an Add-Multiply Bitonic m-sorter of width $n$. Since $n > 2$, $B$ is constructed either by Multiplication or by Addition. The case of Multiplication is easy. So, let $B = B' + B''$ for some two Add-Multiply Bitonic m-sorters $B'$ and $B''$. Hence, $B$ is of the form $\begin{pmatrix} B' \\ B'' \end{pmatrix} C$ for some combining network $C$.

The hard case is where $|B'| \neq |B''|$. Actually, in this case we do not use the induction hypothesis. Assume, without loss of generality, that $|B'| > |B''|$. A straightforward reachability argument imply that the depth of every output port of $B'$ is at least $\lceil \log(|B'|) \rceil$. Since $|B| = |B'| + |B''|$, it follows that $\lceil \log(|B'|) \rceil \geq \lceil \log(n) \rceil - 1$.

The ‘+’ operator is based on Lemma 8 and, in the case at hand, we have $n' \neq n''$. A careful examination of the proof of Lemma 8 in this case reveals that some output of $C$ is computed as the
median of three input keys (rather than two input keys and a fictive key). This implies that $C$ has a path of two gates from some output port of $B'$ to some output port of $B$. This, and the fact that the depth of any output port of $B'$ is at least $\lceil \log(n) \rceil - 1$, imply that $d(B) \geq \lceil \log(n) \rceil + 1$.

References


