# Layout of the Batcher Bitonic Sorter

(Extended Abstract)

Shimon Even * S. Muthukrishnan † Michael S. Paterson ‡ Süleyman Cenk Şahinalp §

## Abstract

The grid-area required by a sorting net for input vectors of length $N$ is shown to be at least $(N-1)^2/2$. Of all sorting nets which use $o(N^2)$ comparators, the bitonic sorting net of Batcher has been known to have a layout of $O(N^2)$, but the hidden constant factor has not been investigated. A straightforward use of known techniques leads to a layout of grid-area $20.25N^2$.

We present area-efficient layouts of the bitonic sorter. First, we describe a flip-bitonic sorting net – it is isomorphic to Batcher's bitonic sorter but leads naturally to a layout of grid-area less than $4N^2$. Second, we present a butterfly-based layout of the bitonic sorter with grid-area of $3N^2 + O(N)$. The former does not use *knock-knees* while the latter relies on them and is more compact.

*This work was done at Bell Labs, Lucent Technologies, while the author was on leave from the Technion. Computer Science Department, Technion - Israel Inst. of Tech., Haifa, Israel 32000, even@cs.technion.ac.il.

†Bell Labs, Murray Hill, NJ, USA. muthu@research.bell-labs.com.

‡Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK; msp@dcs.warwick.ac.uk.
The third and fourth authors were supported in part by ESPRIT LTR Project no. 20244 — ALCOM-IT.

§Department of Computer Science, University of Warwick, Coventry, CV4 7AL, UK; and Center for BioInformatics, University of Pennsylvania, Philadelphia, PA, USA. cenk@dcs.warwick.ac.uk.


**SPAA 98 Puerto Vallarta Mexico**


## 1 Introduction

A *sorting net* has $N$ input terminals and $N$ output terminals. If $N$ (real) numbers are fed into the input terminals, the same numbers appear sorted on the output terminals.

The nets discussed in this paper are constructed of comparison boxes, called *comparators*, and fixed *wires*. Each of the comparators has two input terminals and two output terminals. If two numbers are fed to the two inputs, the same numbers emerge on the outputs; in our drawings, the smaller of these emerges upwards and the greater downwards. See Figure 1.

All wires are directed links which connect a net input terminal or a comparator output to a net output terminal or a comparator input. If we view the net as a directed graph, where the comparators and net terminals play the role of the vertices, and the wires play the role of the directed edges, then the resulting directed graph

is assumed to be acyclic.

Sorting nets have been studied since the fifties [1]. They are useful as fast circuits to perform sorting of data, as well as for message routers, by sorting the addresses which the messages are aimed for.

In trying to pack sorting nets into a small area or space, it is natural to investigate how much grid area is necessary for embedding the the corresponding directed graph. In this study the following rules for a graph layout on the grid are used:

- Vertices of the graph are mapped to grid-points, at most one vertex per grid-point.
- Edges of the graph are to be routed along grid paths by an edge disjoint mapping. Namely, an edge of the grid may not belong to more than one routing path. Note that at most two paths can meet at a grid-point. In particular, the meeting may be *straight* or *knock-knee*: in the former, neither path changes direction at the intersection point and in the latter, both the paths change directions simultaneously.
- If a vertex is mapped to a grid-point, then all paths representing edges incident on this vertex must begin or end at that grid-point, and no other path is allowed to pass through that point.

The *grid-area* of a layout is defined to be $A$ if there is an $a \times b$ rectangle $R$ which encompasses the layout, namely, all grid-points of the layout are either inside $R$ or on its boundary, and $A = (a+1)\cdot(b+1)$ is the least real number for which this holds. It is not required that the sides of $R$ be parallel to the grid lines or that $a$ or $b$ be integers. For more information on the reasons for this definition, see the appendix of [2].

We present the following results in this paper. We present a lower bound of $0.5(N-1)^2$ on the grid-area needed for any sorting net (Section 2). Of all sorting nets which use $o(N^2)$ comparators, the bitonic sorting net of Batcher has been known to have a layout of $O(N^2)$, but a straightforward use of known techniques leads to a layout of grid-area $20.25N^2$. Our main results are two improved area-efficient layouts of the bitonic sorter. We describe a flip-bitonic sorting net – it is isomorphic to Batcher's bitonic sorter but leads naturally to a layout of grid-area less than $4N^2$ (Section 3). We then present a butterfly-based layout of the bitonic sorter with grid-area of $3N^2+O(N)$ (Section 4). The former does not use knock-knees while the latter relies on them and is more compact.

## 2 Lower and Upper bounds on the Area of Sorting Nets

Consider a net with $N$ input terminals and $N$ output terminals, of a structure similar to a sorting net, except that instead of comparators one has switching boxes. The difference is that a switching box has to be set externally, and does not decide on its own whether the numbers on its inputs should stay in the same order on its outputs, or be swapped. Such a net is called *rearrangeable* (see, for example [3]) if every permutation on the vector of $N$ input numbers, fed to the input terminals, can be realized by some setting of the switching boxes.

The following theorem and its proof proceed along lines initiated by Thompson [4].

**Theorem 2.1** *The grid-area of a rearrangeable net for $N$ inputs is greater than $\frac{1}{2}(N-1)^2$.*

The proof is omitted from this abstract.

Since every sorting net is rearrangeable, the lower bound on the grid-area, as in Theorem 2.1, applies to sorting nets as well.

There is a sorting net of grid-area $(N+1)^2$. One such net was presented in [5]. Yet, this net may not be suitable, since it has $\Omega(N^2)$ comparators, and the depth of the net, i.e., the maximum length of a directed path, in terms of comparators, from input to output, is $2N-3$. The bitonic sorter of Batcher [8] has $O(N \log^2 N)$ comparators. We will discuss its layout area. Asymptotically, nets with fewer comparators are known [6] [7], but they hide an enormous constant, and use expanders, which make their layouts in small grid-area very unlikely to exist.

# 3 A Flip-Bitonic Sorter Without Knock-knees

In this section a new method to lay out the bitonic sorter is described. The recursive construction of this sorter is described from scratch, including the proposed layout. The grid-area is shown to be less than $4N^2$. Lines in the layout are slanted at 45° with respect to the input and output columns. It is somewhat reminiscent of Wise's layout of the butterfly net. See [9].

Let us start with a few definitions and observations.

The *zero-one principle* says that if a network sorts all $2^N$ binary input vectors, then it sorts any $N$-vector of real numbers. This theorem is attributed by Knuth [1] to W.G. Bouricius. See [1] for a proof. For this reason, only sorting of zeros and ones is discussed.

A word of zeros and ones is called *bitonic* if it is of the form $0^a1^b0^c1^d$, and either $a$ or $d$ is zero. (Clearly, $a+b+c+d=N$.) *

*Batcher called a real vector bitonic if it is first nondecreasing and then nonincreasing, or if there is a circular rotation of its elements which brings it to this form. By using the zero-one principle we can restrict our discussion to the definition given here.

Let $w \in \{0,1\}^*$, i.e., $w$ is a binary word. Denote by $w^r$ the reversed word, i.e., the word resulting from a left-right flip of $w$.

**Lemma 3.1** *If the binary word $w_1w_2$ is bitonic, so is $w_1^r w_2^r$.*

The proof is omitted from this abstract.

Assume $\chi_1 = x_1x_2\cdots x_N$, while $\chi_2 = x_{N+1}x_{N+2}\cdots x_{2N}$. If $\chi_1\chi_2$ is bitonic, $\chi_1\chi_2^r$ is called *flip-bitonic.*

However, if $\chi_1\chi_2$ is bitonic, by Lemma 3.1, so is $\chi_1^r\chi_2^r$. Thus, $\chi_1^r\chi_2$ is also flip-bitonic.

Our purpose is to construct a sorter of flip-bitonic input vectors. Such a net, for $N=2^n$ inputs, is denoted by $\mathcal{FM}_n$. Its construction is recursive, and $\mathcal{FM}_1$ consists of a single comparator. The construction of $\mathcal{FM}_{n+1}$ consists of two copies of $\mathcal{FM}_n$, one above the other, with an additional layer of comparators and wiring in front of it. This is done as follows.

Consider now the *array net* shown in Figure 2. It consists of a column of input terminals fed with an input vector $x_1x_2\cdots x_{2N}$, an array of $N$ comparators and a column of output terminals producing the output vector $y_1y_2\cdots y_{2N}$.

**Lemma 3.2** *If the input vector $x_1x_2\cdots x_{2N}$ of the array net is binary and flip-bitonic, then its output vector $y_1y_2\cdots y_{2N}$ satisfies one of the following two conditions:*

1. *The vector $y_1y_2\cdots y_N$ is all zeros and $y_{N+1}y_{N+2}\cdots y_{2N}$ is bitonic.*
2. *The vector $y_{N+1}y_{N+2}\cdots y_{2N}$ is all ones and $y_1y_2\cdots y_N$ is bitonic.*

The proof is omitted from this abstract.

Next, we add to the array net *flip-wiring*, which converts each of $y_1 y_2 \cdots y_N$ and $y_{N+1} y_{N+2} \cdots y_{2N}$ from bitonic to flip-bitonic. See Figure 3. The purpose of the additional flip-wiring is to enable the connection of two half-size sorters of flip-bitonic vectors. This completes the description of $\mathcal{FM}_{n+1}$ and the proof, by induction on $n$, that it sorts. See Figure 4(b) and (c) for $\mathcal{FM}_2$ and $\mathcal{FM}_3$, respectively.

The construction of the complete *Flip-Bitonic Sorter*, $\mathcal{F}_n$, is done recursively. $\mathcal{F}_1$ is just a comparator, drawn with diagonal lines, just as $\mathcal{FM}_1$. To build $\mathcal{F}_{n+1}$, use two copies of $\mathcal{F}_n$, an upper and a lower, and attach the $2^{n+1}$ outputs to the inputs of $\mathcal{FM}_{n+1}$. See Figure 5.

It is easy to see that the layout of $\mathcal{F}_n$ is enclosed by a rectangle of height $\sqrt{2} \cdot (N-1)$ and length $\sqrt{2} \cdot (2N - (2 + \log N))$. Thus, the grid-area is bounded by $4N^2$.

# 4 Bitonic Sorter Using Knock-knees

In this section we present a tighter area layout for the bitonic sorter. There are two main differences between this construction and the one presented in Section 3, namely, the construction here has lines aligned with the input/output columns, and it uses knock-knees. We start our description with a few definitions.

A *butterfly permutation* $\mathcal{B}_{\ell,k}$, $k \leq \ell$, is a bipartite graph with $L$ input and $L$ output nodes, where $L = 2^\ell$. Each input node $I_{\ell,k}[i]$, $0 \leq i \leq L-1$, is connected to two output nodes, $O_{\ell,k}[i]$, and $O_{\ell,k}[\bar{i}^k]$, where $\bar{i}^k$ denotes the binary number obtained by flipping the $k^{\text{th}}$ most significant bit of the binary number $i$.

The bitonic sorter network relies on *bitonic merger* networks. The bitonic merger network $\mathcal{BT}_n$ with $N = 2^n$ inputs is constructed recursively. $\mathcal{BT}_1$ is a single comparator. $\mathcal{BT}_{n+1}$ with inputs $X[0, \ldots, 2N-1]$ consists of the following two layers. The first layer comprises $N$ comparators, $C_0, \ldots, C_{N-1}$. The inputs of $C_i$ are $X[i]$ and $X[N+i]$ for each $i$. The second layer comprises an upper and a lower copy of $\mathcal{BT}_n$. The upper (respectively lower) output of $C_i$ is the $i^{\text{th}}$ input to the upper (lower, respectively) copy of $\mathcal{BT}_n$. The $N$-vector of outputs from the upper copy of $\mathcal{BT}_n$ followed by that of the lower copy forms the $2N$-vector output of $\mathcal{BT}_{n+1}$.

The bitonic sorter $\mathcal{BTS}_n$ on $N = 2^n$ inputs is constructed recursively. $\mathcal{BTS}_1$ is a single comparator. $\mathcal{BTS}_{n+1}$ is constructed by feeding the inputs from an upper and a lower copy of $\mathcal{BTS}_n$; the $N$ outputs of the upper copy are fed to the $N$ upper inputs of a $\mathcal{BT}_{n+1}$ in that order, and the $N$ outputs of the lower copy of $\mathcal{BTS}_n$ are fed in the reversed order to the lower $N$ inputs of the $\mathcal{BT}_{n+1}$. The entire network $\mathcal{BTS}_n$ sorts $N$ inputs [8]. (For alternative but equivalent versions of the bitonic sorter networks, see [1]).

Our layout for the bitonic merger is based on layouts of the butterfly permutations $\mathcal{B}_{\ell,1}$, and $\mathcal{B}_{\ell,2}$. The primary challenge in obtaining such layouts is to avoid column and row conflicts amongst the connections. Consider two distinct connections, one starting from grid row $r$, and another ending at grid row $r$. Clearly, the former must turn away from grid row $r$ at, or to the left of, the column used by the latter to reach grid row $r$. For any row $r$, we refer to these two columns as the *in-column* and the *out-column*, respectively.

Initially, we restrict ourselves to two types of connections: (i) straight connections, and (ii) *jogs*. A jog is a horizontal-vertical-horizontal path from input node $(k_1, 0)$ to output node

$(k_2, x)$, $k_2 \neq k_1$, with a vertical segment on some column $c$, $0 < c < x$. We use two types of jogs: a jog that has source $(r_1, 0)$ and destination $(r_2, x)$ is a *down-jog* if $r_1 < r_2$, and an *up-jog* if $r_1 > r_2$ (see Figure 7).

**Lemma 4.1** *A butterfly permutation $\mathcal{B}_{\ell,k}$, for $k = 1, 2$, can be laid out on a grid with $2L + 1$ rows and $L - 1$ columns by using only straight connections and jogs. The column assignment of a jog with input row $r$ is given by the following function $f(r)$: $f(r) = L - 2i + 1$, for $r = 4i$; $f(r) = L - (r/2\overline{\ell}^k) + 1$, for $r = 4i + 2$.*

**Proof.** We first describe the input-to-output connections of $\mathcal{B}_{\ell,k}$ on the grid. Its $2L$ input lines are mapped to grid points $(0,0), \ldots, (2L-1, 0)$. Input node $i$ is connected to the two input lines on rows $2i$ and $2i + 1$. The output lines are mapped to grid points $(1, L+1), \ldots, (2L, L+1)$. Output node $i$ is connected to the two output lines on rows $2i + 1$ and $2i + 2$.

The connections from the input lines to the output lines are as follows: $(2i+1, 0)$ is connected to $(2i + 1, L + 1)$ via a straight connection, and $(2i, 0)$ is connected to $(\overline{i} + 2, L + 1)$ (rather than to $(\overline{i}, L+1)$ for reasons which will be clear later), via a jog.

It is clear that the given column-assignment function $f$ is one-to-one, hence the assignment is free of column conflicts. Therefore we only need to show that the layout results in conflict-free row assignments as well. To prove this, we simply show that for any row $r$, $o_r = i_r + 1$, where $o_r$ is the out-column, and $i_r$ is the in-column of $r$. Consider $r = 4i$, $0 \leq i \leq L/2 - 1$. The in-column, $i_r$, is $L - 2i + 1$. The input row of the jog whose output row is $r = 4i$ is $2\overline{(2i-1)}^k$), hence the out column, $o_r$, of $r$ is $L - 2i$. Similarly for $r = 4i + 2$, its in-column is $f(r)$ and its out-column is $f(r) + 1$. See Figure 8 for the layout of a 32 input butterfly permutation $\mathcal{B}_{\ell,1}$. ∎

In our bitonic sorter layout, we will need to merge $N/(2L)$ butterfly permutations of the form either $\mathcal{B}_{\ell,1}$, or $\mathcal{B}_{\ell,2}$ on the same set of columns. If we simply concatenated these layouts as described in the above lemma, we would need to use $N + N/(2L)$ rows. This is wasteful. In what follows, we describe a procedure to merge the $N/(2L)$ butterfly permutations $\mathcal{B}_{\ell,1}$ by using only $N + 1$ rows. We leave the details of how to merge the butterfly permutations $\mathcal{B}_{\ell,2}$ to the full paper.

Henceforth we focus on the layout of the butterfly permutation $\mathcal{B}_{\ell,1}$, which, for brevity, we will call the *butterfly*. The following observation (from the proof of Lemma 4.1) will prove useful.

**Observation 4.2** *For any row $r$, if the jog that has source $(r, 0)$ has column assignment $c$, then the jog that has the destination $(r, x)$ has the column assignment $c + 1$.*

We say that $(r, c) : (r, c+1)$ is an *up-slot* if the jog with source $(r, 0)$ is an up-jog with column assignment $c$, and the jog with destination $(r, x)$ is a down-jog with column assignment $c+1$. Similarly, we say that $(r, c) : (r, c + 1)$ is a *down-slot* if the jog with source $(r, 0)$ is a down-jog with column assignment $c$, and the jog with destination $(r, x)$ is an up-jog with column assignment $c + 1$. See Figure 8 for up and down slots.

**Lemma 4.3** *Between columns $c$ and $c+1$, there is precisely one of an up-slot or a down-slot. If there is an up-slot between columns $c$ and $c + 1$, then there is a down-slot between columns $c + 1$ and $c + 2$, and vice versa.*

We now show how to modify our layout from Lemma 4.1 for melding two butterfly

permutations $\mathcal{B}_{\ell,1}$ with smallest number of rows possible. We retain all the straight connections and all the jogs as they are except the jogs on rows 0 and $2L$, which are modified to include several turns. The jog in row $2L$ goes through $(2L,1)$,$(2L,2)$,$(r_2,2)$,$(r_2,3)$,$(2L,3)$,$(2L,4)$, $(r_4,4),\ldots$ etc., where $(r_2,2) : (r_2,3)$, $(r_4,4) : (r_4,5),\ldots$ are up-slots. The jog in row 0 goes through $(0,0)$,$(0,1)$,$(r_1,1)$,$(r_1,2)$,$(0,2)$,$(0,3)$, $(r_3,3)$,$(r_3,4)$,$(0,4),\ldots$ etc., where $(r_1,1) : (r_1,2)$, $(r_3,3) : (r_3,4),\ldots$ are down-slots. We concatenate $N/(2L)$ copies of the butterfly of $L$ input nodes.

The $i$th such butterfly, $0 \leq i \leq N/(2L) - 1$ uses input lines from $2iL$ to $2(i+1)L$ (there are $2L+1$ lines for each) using the modified construction above from Lemma 4.1. In all, the total number of rows is $N+1$. Notice that successive copies of the butterfly from Lemma 4.1 in the layout above share a common row, that is, the bottommost line in the upper butterfly and the topmost line of the lower butterfly share the same row. This causes no problems since our modified construction guarantees that these two lines have no common row portions because of the alternating property of up- and down-slots from Lemma 4.3.

**Theorem 4.4** *There exists a layout of the bitonic sorter that uses an area of at most* $3N^2 + O(N)$.

**Proof.** The layout of the complete bitonic sorter $\mathcal{BTS}_n$ uses bitonic mergers $\mathcal{BT}_n,\ldots,\mathcal{BT}_1$. The merger $\mathcal{BT}_\ell$ uses one layer of comparators followed by two $\mathcal{BT}_{\ell-1}$ mergers. In $\mathcal{BT}_\ell$, the wiring between the input nodes and the first layer of comparators is *precisely* the butterfly $\mathcal{B}_\ell$. The wiring between the first layer of comparators of $\mathcal{BT}_\ell$ and the two $\mathcal{BT}_{\ell-1}$ mergers is realized as a butterfly permutation $\mathcal{B}_{\ell,2}$, which concatenates a $\mathcal{B}_\ell$ butterfly to two $\mathcal{B}_{\ell-1}$ butterflies as in Lemma 4.1. Hence, altogether, the bitonic merger $\mathcal{BT}_{\ell+1}$ uses $L+2L-1$ columns for $L = 2^\ell$. This implies that the total number of columns $\mathcal{BTS}_n$ uses is $3N + O(1)$ and the total area it requires is $3N^2 + O(N)$. ∎

The construction in Theorem 4.4 has the drawback that the number of knock-knees along some connections may be large. In each of the layouts of the butterfly permutations, all connections with the exception of the topmost and the bottommost ones have either two knock-knees, or no knock-knees at all. Now we show how to modify that construction so that each connection has about the same (small bounded) number of knock-knees.

**Lemma 4.5** *The butterfly permutations* $\mathcal{B}_{\ell,1}$, *and* $\mathcal{B}_{\ell,2}$ *with* $L = 2^\ell$ *input and output nodes (hence,* $2L$ *input and output lines) have a layout on the grid in which each input-to-output connection involves a constant number of knock-knees; the number of rows is* $2L+1$ *and the number of columns is* $L$. *These layouts are meldable in the sense that any* $k$ *such layouts can be concatenated on the same set of columns by only using* $2kL+1$ *rows.*

**Proof.** (Sketch) To obtain the layout of the butterfly permutation $\mathcal{B}_{\ell,1}$, we begin with the layout in Lemma 4.1 without the modifications done in the melding step, and borrow notation and definitions from there. So our starting point is as shown in Figure 8 for $\ell = 4$. We divide the layout into two halves, top and bottom, Furthermore we divide each half into two quarters, left and right.

We first describe our construction for the top-left quarter, and then show that the construction

for each of the quarters can be performed independently, resulting in an implementation of the circuit with a constant number of knock-knees per connection.

Our construction relies on iterative application of a *row-exchange* operation, defined as follows. The up-slots of the top-left quarter of the layout in Lemma 4.1 from left to right have decreasing row indices. Specifically, they are at positions $(L-2,1) : (L-2,2)$, $(L-6,3) : (L-6,4), \ldots,$ $(L-(4i+2),2i+1) : (L-(4i+2),2i+2), \ldots,$ $(2, L/2-1) : (2, L/2)$.

Consider the bottommost (hence leftmost) up-slot $(L-2,1) : (L-2,2)$, and the topmost (hence rightmost) up-slot $(2, L/2-1) : (2, L/2)$. We use these up-slots to exchange the row of the ceiling connection with the row of the down-jog of the topmost up-slot. This is done as follows: We first move the straight row segment of the ceiling connection between $(0,1)$ and $(0, L/2)$ to $(2,1)$ and $(2, L/2)$ and connect the locations $(0,1)$ and $(2,1)$, and the locations $(0, L/2)$, and $(2, L/2)$ by two column segments. Simultaneously, we move the straight row segment of the down-jog of the topmost up-slot between $(2,2)$ and $(2, L/2-1)$ to $(0,2)$ and $(0, L/2-1)$ and connect the locations $(2, L/2-1)$ and $(0, L/2-1)$ by a column segment.

Note that we cannot connect the locations $(2,2)$ and $(2,0)$ by a column segment to avoid column conflicts. Hence we move the straight row segment of the down-jog of the topmost up-slot between $(2,1)$ and $(2,2)$ to the row of the bottommost up-slot: $(L-2,1) : (L-2:2)$, and connect the locations $(L-2,1)$ and $(2,1)$ and the locations $(L-2,2)$ and $(2,2)$ by two column segments. Figure 10 demonstrates how a row-exchange operation is performed.

This completes the row-exchange operation between the ceiling connection and the row segment of the down-jog of the topmost up-jog. Note that as a result of such an exchange both the topmost and the bottommost up-slot are moved to row 0.

We iteratively apply the same operation to move the new ceiling connection to another row. Once a ceiling connection is moved, it is never exchanged again. The knock-knees incurred during this operation will be only constant in number per connection as they are generated during the row-exchange operations which are performed at most twice per connection.

Note that the row-exchange operations in all four quarters are performed independent of each other. Moreover, the columns of the up-slots and the down-slots are preserved. Hence two or more butterflies can be melded as in Lemma 4.3 with only a constant number of knock-knees per connection.

The layout for butterfly permutation $\mathcal{B}_{\ell,2}$ can be constructed in a similar fashion. We leave the details of this construction to the full paper. ∎

## Acknowledgments

We would like to thank Peter Winkler and Jeff Lagarias for help in the study of a proper definition for the grid area, Tiziana Calamoneri and Ami Litman, for arousing our interest in the problem, and Rajmohan Rajaraman and Mike Garey for discussions.

## References

[1] D.E. Knuth, *The Art of Computer Programming*, Vol. 3: *Sorting and Searching*, Addison-Wesley, 1973. See Section 5.3.4.

[2] S. Even *Layout Area of Sorting Networks*, manuscript, 1997.

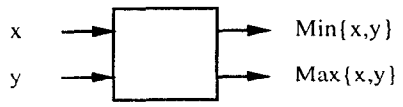


Figure 1: A comparison box

[3] N. Pippenger, "Telephone Switching Networks", *Proceedings of Symposia in Applied Mathematics*, Vol. 26, American Mathematical Society, 1982, pp. 101-133.

[4] C. Thompson, "Area-time Complexity for VLSI", *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*, May 1979, pp. 81-88.

[5] W.H. Kautz, K.N. Levitt and A. Waksman, "Cellular Interconnection Arrays", *IEEE Trans. on Computers*, Vol. C-17, No. 5, May 1968, pp. 443-451. See Fig. 2, page 444.

[6] M. Ajtai, J. Kolmós and E. Szemerédi, "Sorting in $C \log N$ Parallel Steps", *Combinatorica*, Vol. 3, 1983, pp. 1-19.

[7] M.S. Paterson, "Improved Sorting Networks with $O(\log N)$ Depth", *Algorithmica*, Vol. 5, 1990, pp. 75-92.

[8] K. Batcher, "Sorting Networks and their Applications", *Proc. of the AFIPS Spring Joint Computing Conf.*, Vol. 32, 1968, pp. 307-314.

[9] D.S. Wise, "Compact Layouts of Banyan/FFT Networks", *VLSI Systems and Computations*, H.T. Kung, et. al. (editors), Computer Science Press, 1981, pp. 186-195.

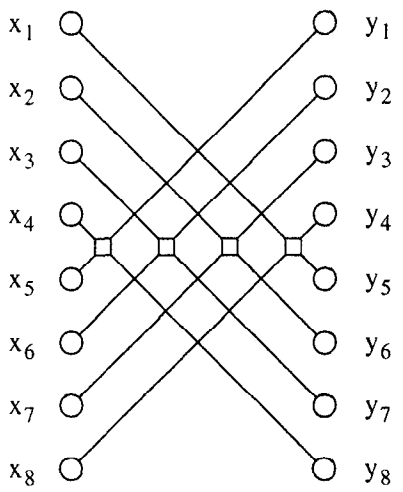


Figure 2: The array net

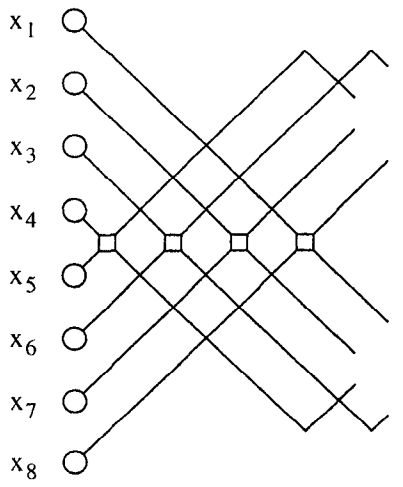


Figure 3: The array net with flip-wiring

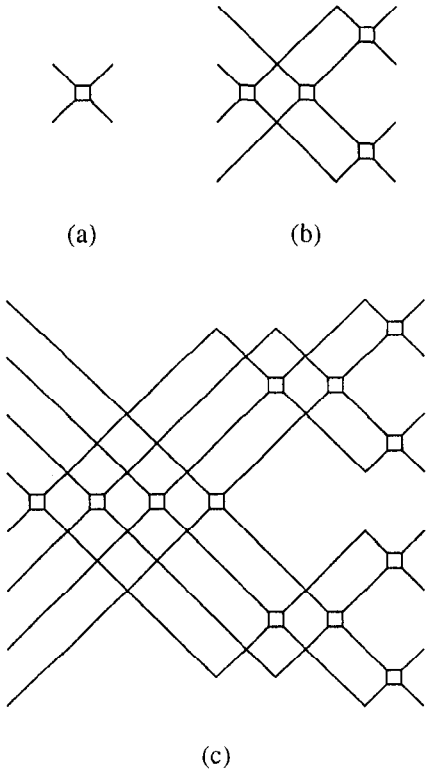


Figure 4: The sorting net for flip-bitonic input vectors

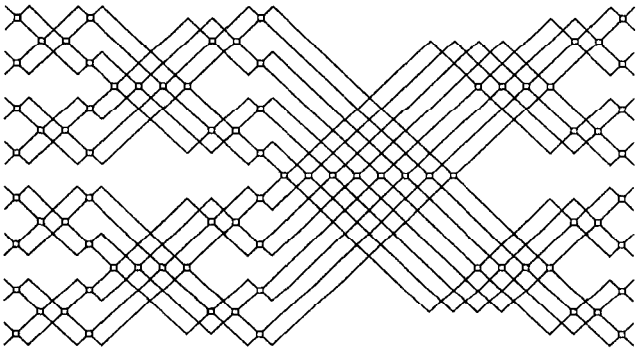

Figure 5: The sorting net for input vectors of length 16

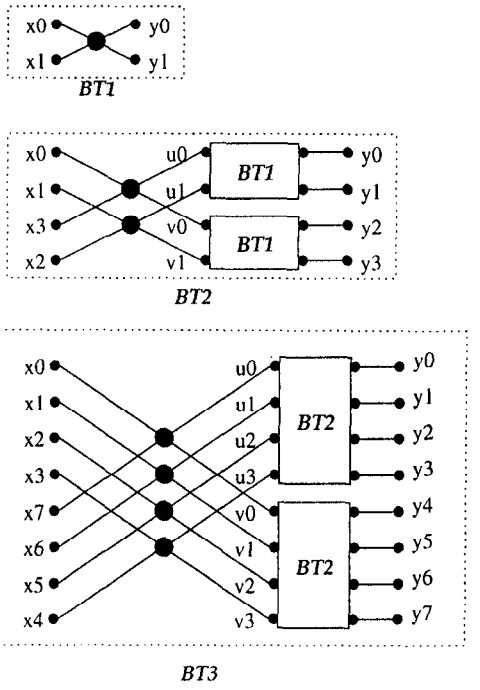


Figure 6: The recursive construction of the bitonic merger networks with 2, 4, and 8 input and output lines.

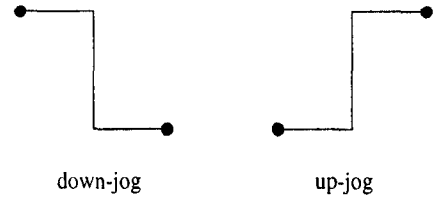


Figure 7: The two types of jogs: the down-jog and the up-jog.

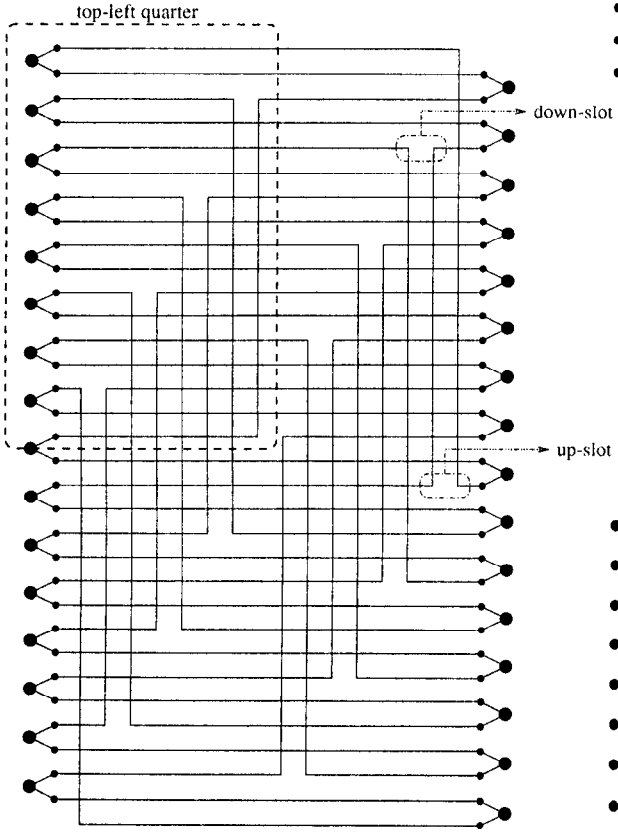


Figure 8: The jog-only implementation of the butterfly with 16 input (output) nodes. The dashed rectangles demonstrate the notions of quarters and slots in the layout.

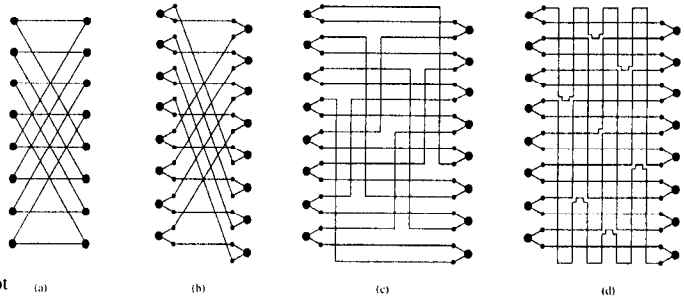


Figure 9: (a) Butterfly with 8 input (output) nodes, 16 input (output) lines. (b) The input and output line assignments on the grid leading to the jog-only implementation. (c) The jog-only implementation after the column assignments to the jogs, (d) The meldable implementation with non-constant knock-knees per connection.

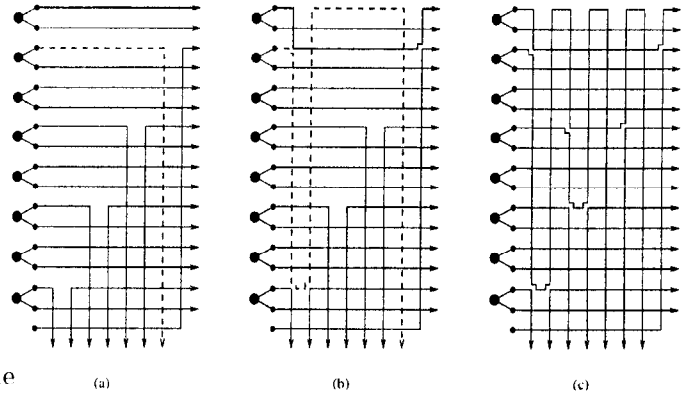


Figure 10: (a) The top-left quarter of the jog-only implementation of a 16 input (output) node butterfly. (b) The row-exchange operation between the ceiling connection (thick line) with the straight row connection of the down-jog of the topmost up-slot. (c) The complete implementation with constant knock-knees per connection.