بسم الله الرحمن الرحیم

شبکه مرتب سازی Bitonic Sort

خلاصه

توالی از عنا صر را Bitonic گویند که در ابتدا افزایش و سپس کاهش یابند و یا برعکس. مثل < 1, 2, 4, 7, 6, 0 > و همچنین هر شیفت چرخ شی از Bitonic نیز یک توالی Bitonic است. به عنوان مثال < 8, 9, 2, 1, 0, 4 > .

Bitonic یکی از الگوریتم های مرتب سازی برای ما شین های موازی ا ست که به عنوان یکی از بهترین الگوریتم های مرتب سازی از نظر پیچیدگی و هزینه شناخته میشود. یک سری مرتب شده یا سری صعودی است یا سری نزولی اما سری بایتو نیک ترکیبی از دو زیر سری صعودی و نزولی است.

1. مقدمه

در مرتب سازی Bitonic ما از دو مرحله استفاده میکنیم ، مرحله اول تبدیل اعداد نا مرتب به یک توالی Bitonic و سپس مرتب کردن توالی Bitonic به اعداد مرتب شده. بنابراین طبق این گفته همیشه اعداد ما باید توانی از 2 باشند.

- **حال به این موضوع می پردازیم که چطور رشته ی بایتونیک بسازیم؟**

این عمل به صورت ریکرسیو انجام می شود. مثلا اگر یک رشته 8 تایی داریم آنرا به دو رشته 4 تایی تقسیم می کنیم. یعنی الان دوتا رشته 4 تایی داریم که جداگانه بر روی آن کار می کنیم. این رشته 4 تایی را چطور بایتونیک کنیم؟ ساده است! رشته چهارتایی در حقیقت دوتا دوتایی است. زمانی این چهارتایی بایتونیک می شود که دوتای اول را نزولی و دوتای دوم را صعودی سورت کنیم. در این صورت یک چهارتایی بایتونیک داریم. برای چهارتایی دوم هم همین کار را تکرار می کنیم.

- **چطور رشته بایتونیک را سورت کنیم؟**

در بالا دیدم که یک رشته بایتونیک چهارتایی ساختیم (یعنی نصفش نزولی و نصفش صعودی بود). نصف چهار می شود دو. ما آیتم اول را با آیتمی که دوتا بعد از خودش هست مقایسه می کنیم. یعنی آیتم یک را با سه و آیتم دو را با چهار. چون قصد سورت کردن داریم (برخلاف مرحله قبل که در حال ساخت بایتونیک بودیم) لذا به صورت نزولی آیتم ها را مرتب میکنیم.

تا اینجا یک رشته 4 تایی را به صورت نزولی مرتب کردیم. اما رشته ما 8 تایی بود. مهم نیست! همین کارها را برای 4 تایی دوم انجام میدهیم با این تفاوت که در مرحله آخر به صورت صعودی سورت میکنیم (برخلاف چهارتایی اول که نزولی سورت شد). چرا؟ چون اگر 4تایی اول نزولی با شد و 4تایی دوم صعودی خود این توالی یک رشته بایتونیک 8 تایی می شود.

حالا طبق الگوریتم این رشته بایتونیک 8 تایی را هم سورت می کنیم.

نصف 8 تا میشود 4. آیتم اول رو با 4 آیتم بعدتر از خودش مقایسه میکنیم (سورت نزولی).

همین طور آیتم دوم را با 4 آیتم بعدتر از خودش و ...

باز نصف 4 تا میشه 2. آیتم اول رو با 2 تا بعدتر از خودش مقایسه میکنیم ...

باز نصف 2 تا میشه 1. آیتم اول رو با 1 آیتم بعدتر از خودش مقایسه میکنیم ...

در این مرحله یک 8 تایی سورت شده داریم.

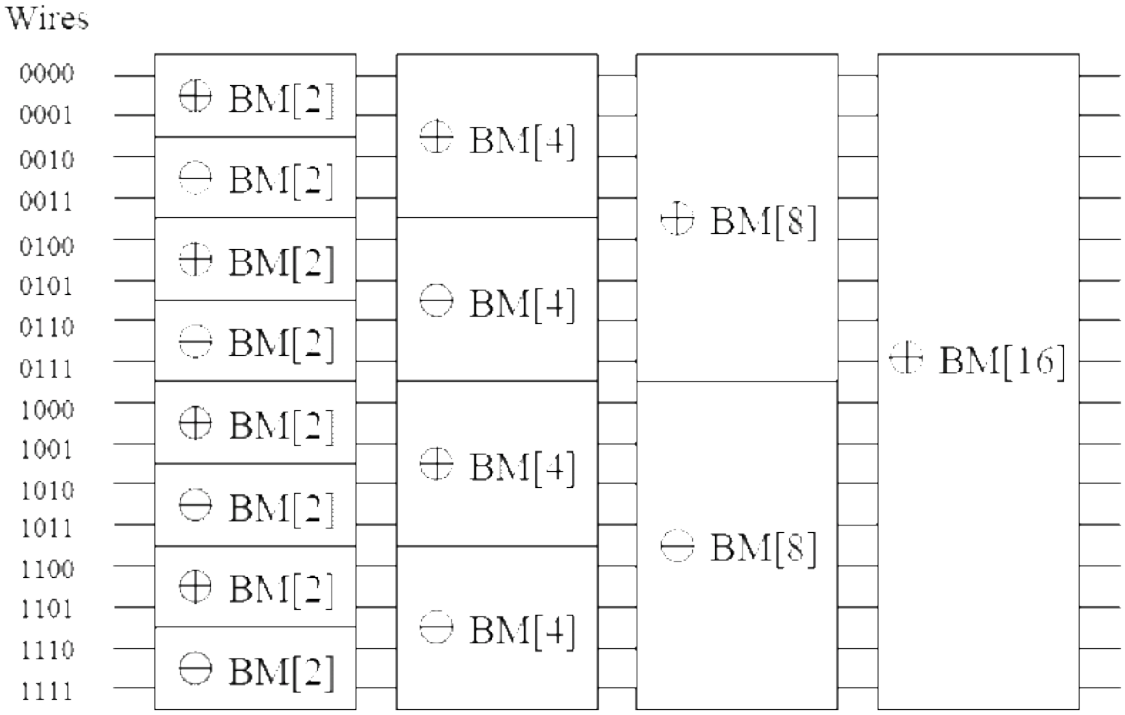
2. معرفی شبکه مرتب سازی Bitonic

این شبکه دارای Log n ستون است (تعداد گام ها) و هر ستون شامل n/2 مقایسه گر می باشد.

صعودی : مقایسه گرها از نوع $\oplus$ هستند. شبکه مرتب سازی Bitonic صعودی را با $\oplus BM[n]$ نشان میدهند.

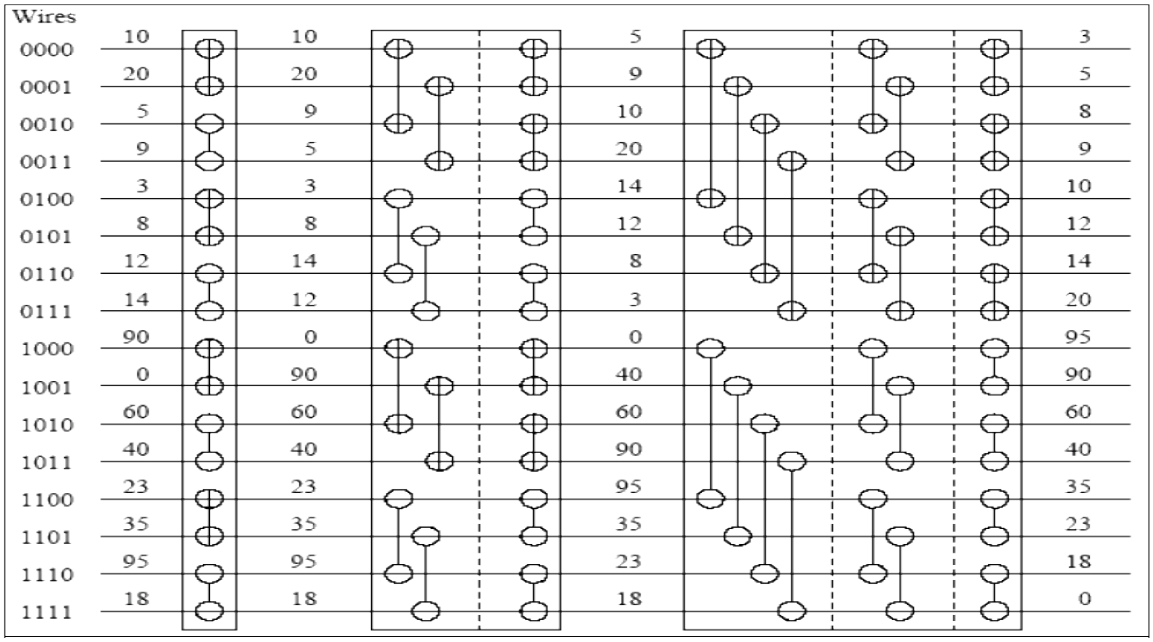
نزولی : مقایسه گرها از نوع Θ هستند. شبکه مرتب سازی Bitonic نزولی را با $BM[n]$ نشان می دهیم.

به عنوان مثال شکل زیر Bitonic Merging Sort را نشان می دهد که یک شبکه ی کامل است . ورودی دلخواه را گرفته و آن را مرتب میکند.



شکل شماره 1

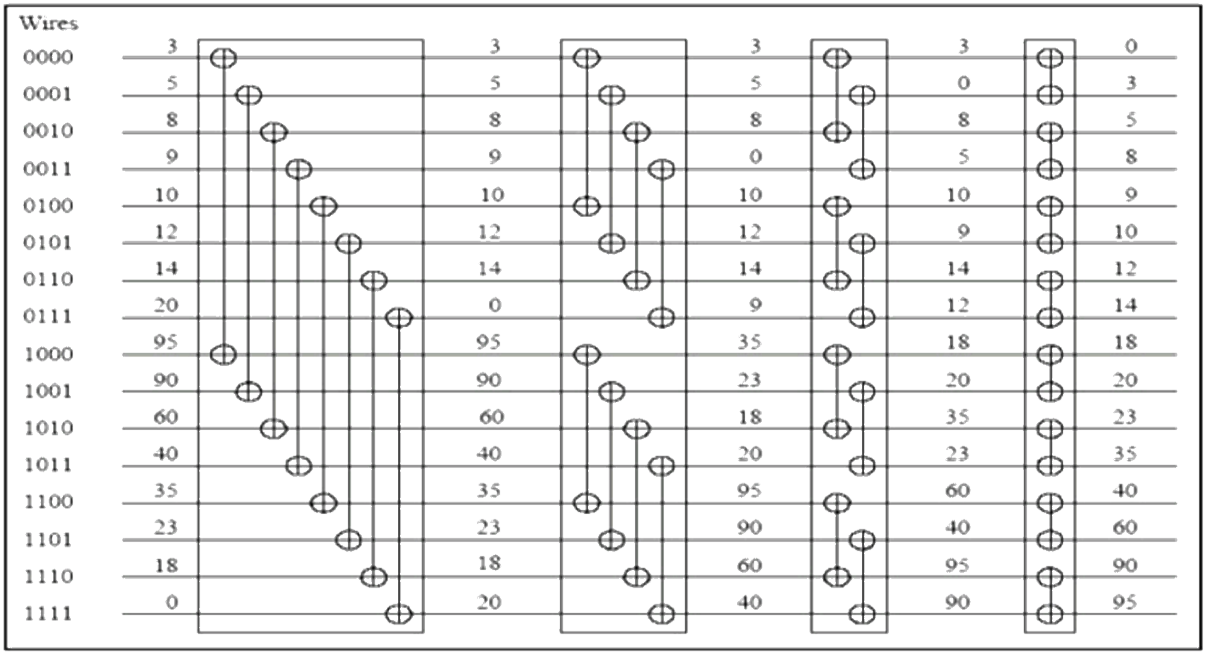
2.1 مرحله اول : تبدیل اعداد نامرتب به یک توالی Bitonic



شکل شماره 2

در تصویر شماره 2 ، 16 عدد به عنوان ورودی دریافت می کنیم و آنها را بر طبق گفته های بالا مرتب می کنیم.
در مثال بالا اعداد نا مرتب < 10, 20, 5, 9, 3, 8, 12, 14, 90, 0, 60, 40, 23, 35, 95, 18 > از ورودی ئریافت شده و طبق شکل به ترتیب اعداد 10 و 20 به صورت صعودی ، اعداد 5 و 9 به صورت نزولی ، 3 و 8 به صورت صعوعی و همینطور یکی در میان تا انتها به صورت صعودی نزولی مرتب شده اند. پس از طی مراحل بالا در انتها همانطور که مشاهده می شود ما یک توالی بایتونیک به صورت
< 3, 5, 8, 9, 10, 12, 14, 20, 95, 90, 60, 40, 35, 23, 18, 0 > در اختیاد داریم. پس وارد فاز دوم شده و طبق الگوی زیر توالی بایتونیک را به توالی مرتب تبدیل می کنیم. (در مقدمه گفته شد که برای مرتب سازی کامل اعداد از 2 مرحله استفاده می کنیم.)

2.2 مرحله دوم : تبدیل توالی بایتونیک به اعداد مرتب



شکل شماره 3

در این شکل به طور کامل مشخص است که اعداد بایتونیک شده ی ما در مرحله اول به عنوان ورودی دریافت شده و پس از انجام مقایسه ها و جابجایی طی 4 مرحله ، خروجی اعداد مرتب < 0, 3, 5, 8, 9, 10, 12, 14, 18, 20, 23, 35, 40, 60, 90, 95 > می باشد.

3. تحلیل پیچیدگی زمانی Bitonic Merging Sort

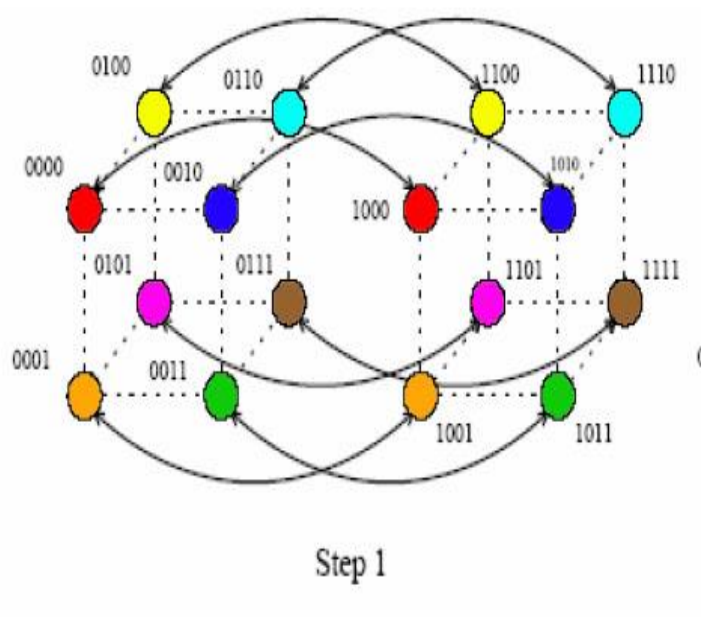
سطح آخر یک شبکه Bitonic با n ورودی و عمق $\log n$ است.سطح ما قبل آخر دو شبکه Bitonic موازی با n/2 ورودی است.
عمق کل شبکه برابر است با d(n)=d(n/2)+log n و تعداد کل سطوح برابر $\Theta(log^2 n)$ است.
چرا که $d(n) = \sum_{i=1}^{log\, n} i = (log^2 n + \log n) / 2 = \Theta(log^2 n)$
پیچیدگی در حالت موازی $\Theta(log^2 n)$ و در حالت سریال $(n\, log^2 n)$ است.

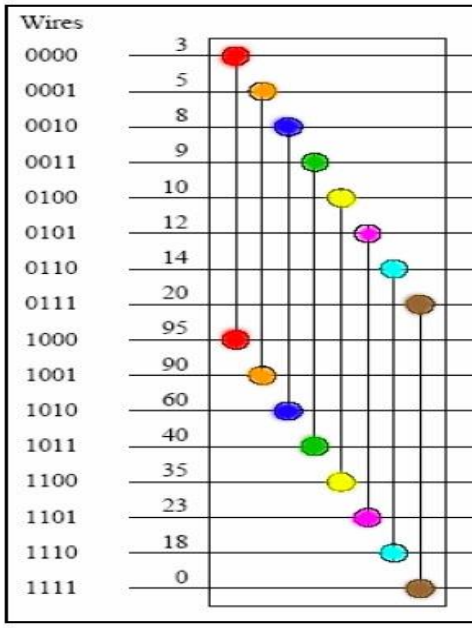
```
public class BitonicSorter implements Sorter
 {
        private int[a];    // sorting direction
  private final static boolean
ASCENDING=true, DESCENDING=false;
 public void sort(int[a])
 {
     this.a=a;
     bitonicSort(0, a.length, ASCENDING);
}
 private void bitonicSort(int lo, int n, boolean
dir)
  {
     if (n>1)
        {          int m=n/2;
                  bitonicSort(lo, m, ASCENDING);
                  bitonicSort(lo+m, m,

   private void bitonicMerge(int lo, int n, boolean dir)
  {
      if (n>1)
        {
             int m=n/2;
              for (int i=lo; i<lo + m; i++)
                 compare(i, i+m, dir);
                 bitonicMerge(lo, m, dir);
                 bitonicMerge(lo+m, m, dir);
  }     }

  private void compare(int i, int j, boolean dir)
    {       if (dir==(a[i]>a[j]))
                exchange(i, j);
    }
   private void exchange(int i, int j)
  {
             int t=a[i];
             a[i]=a[j];
 a[j]=t;    }
          }    // end class BitonicSorter
```

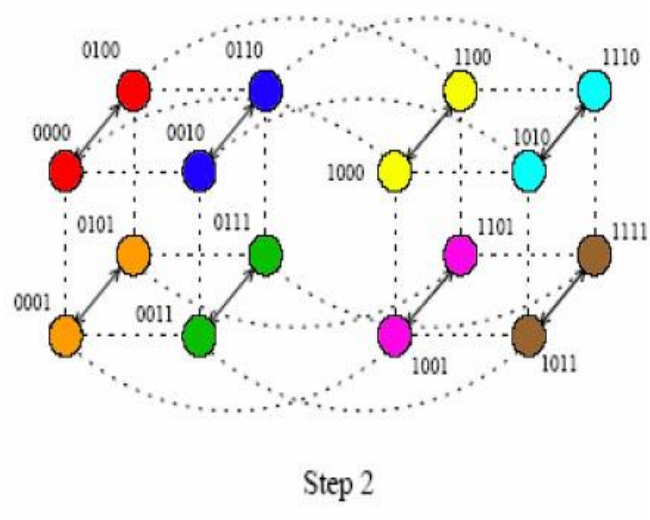
5. نگاشت Bitonic Sort به Hypercube

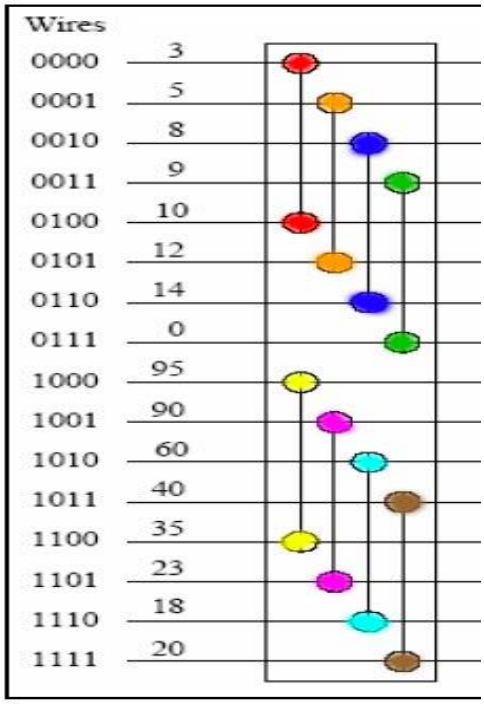
فرض کنید هر پرو سه یک عنصر دارد. هر پرو سه به یک نود از Hypercube نگاشت شود. مقایسه بین عناصری انجام می شود که Label پروسه آنها فقط در یک بیت متفاوت است. تمامی ارتباطات فقط بین همسایه هاست (چراکه نود های همسایه در Hypercube فقط در یک بیت تفاوت دارند) . هر ارتباط بین 2 نود نمایانگر یک Compare- exchange است.

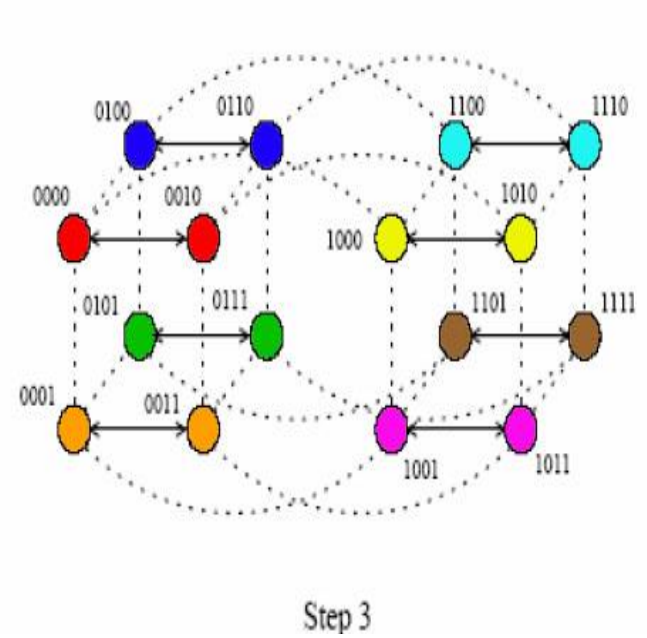
Bitonic Sort با 2^d ورودی به Hypercube با d بعد نگاشت می شود.

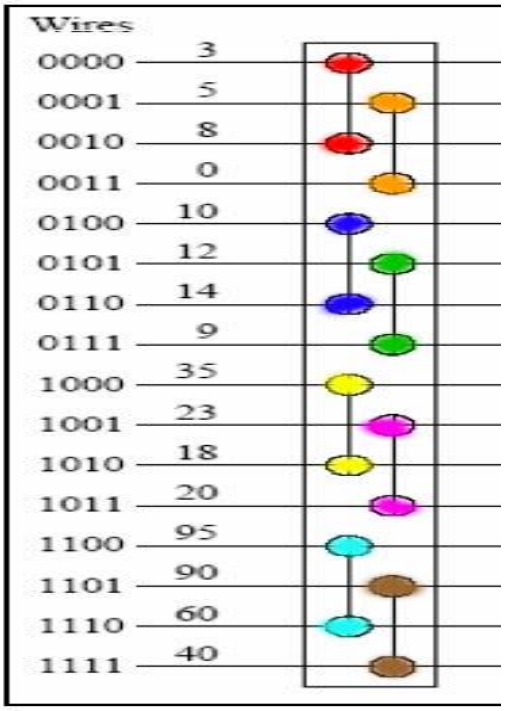


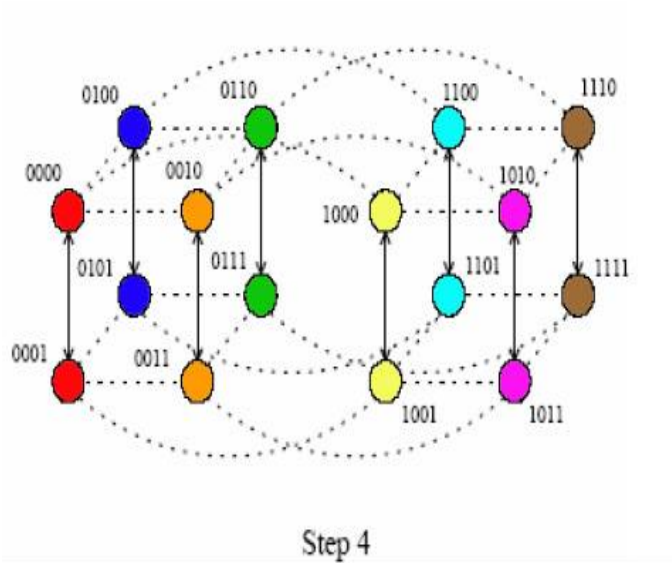


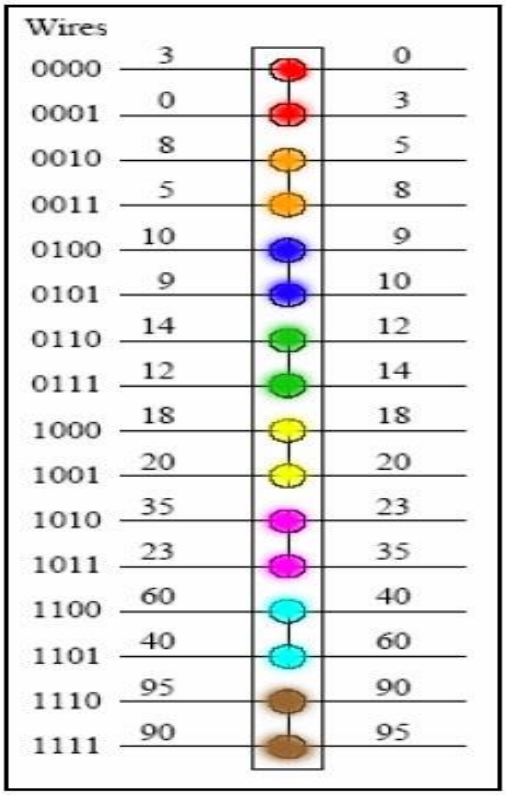





0100
0110
1100
1110
0000
0010
1010
1000
0101
0111
1101
1111
0001
0011
1001
1011
Step 3


Wires
0000 3
0001 5
0010 8
0011 0
0100 10
0101 12
0110 14
0111 9
1000 35
1001 23
1010 18
1011 20
1100 95
1101 90
1110 60
1111 40


0100
0110
1100
1110
0000
0010
1010
1000
0101
0111
1101
1111
0001
0011
1001
1011
Step 4


Wires
0000 3 0
0001 0 3
0010 8 5
0011 5 8
0100 10 9
0101 9 10
0110 14 12
0111 12 14
1000 18 18
1001 20 20
1010 35 23
1011 23 35
1100 60 40
1101 40 60
1110 95 90
1111 90 95

6. تحلیل پیچیدگی زمانی Bitonic Sort در Hypercube

در هر مرحله کلیه نود ها عمل Compare-exchange را در زمان $\Theta(1)$ انجام می دهند.کل تعدادمراحل $(\log n\ (\log n + 1))/2$ است.بنابراین $Tp = \Theta(\log^2 n)$است . در مقایسه با حالت سریال $(\Theta(n\log^2 n)$ همین الگوریتم بهینه است.اما نسبت به بهترین الگوریتم مرتب سازی سریال$(\Theta(n\log n)$) بهینه نیست.

7. کد Bitonic Sort در Hypercube

```
Procedure Bitonic_Sort(labal, d)
Begin
        For i := 0 to d-1 do
                For j := I downto 0 do
                        If (i + 1 )st bit of label ≠ jth bit of label then
                                Comp-exchange max(j);
                        else
                                comp-exchange min(j);
End Bitonic_Sort
```