On a connection pattern between 2^N elements.

We consider $M = 2^N$ elements identified by an address running from 0 through M-1. We shall call elements with an even (odd) address "even (odd) elements". In the kind of arrangements we are going to consider, M is constant (e.g. 1024). Each element has the ability of receiving, storing, manipulating and sending information: it is "a little computer".

In order to enable them to exchange information there will be a set of paths, each path leading from an element to (usually) another element. Together the paths form "the connection pattern". We are going to study a specific connection pattern; in most applications that we could think of, the paths will be used for one-way traffic only. (Besides information via the paths, the elements may receive, via other connections, control information for resetting, for synchronization etc.; these additional connections won't be mentioned any further.)

The pattern we are going to study consists of 2M paths. With $M1 = M/2$, for $0 \leq k < M1$, each element k and each element k+M1 will have two outgoing paths, one to element 2k and one to element 2k+1. As a result each element will also have two incoming paths. (Elements 0 and M-1 have the exceptional property of having an outgoing path leading to itself.)

The fundamental property of this connection pattern is that in N steps it provides a connection from any element A to any element B. To see this we form a 2N-digit number, formed by concatenating the N binary digits of address A with the N binary digits of address B. Over this number we place a window, N bits wide, initially in the left-most position, so that through the window we see the digits of address A. When the window is moved over one position to the right N times, we shall see the digits of address B when the window is in its final position. The numbers seen in the N-1 intermediate positions of the window are now in that order interpreted as the addresses of the elements via which the information will travel from A to B. The proposed connection pattern provides precisely the paths leading from one element to the next (the corresponding value of k is the value of the N-1-bit number formed by the overlapping bits of the two successive N-bit addresses).

Selection in N steps.

We assume that each element stores a value v[j] of an array v[0:M-1]. We regard permuting these values in such a way that for given value of j, v[j] is the contents of element nr. 0 as "selecting v[j]" -element 0 may be the only element connected to the outer world! -.

Denoting with c[i] the contents of element i, we restrict ourselves primarily to the M permutations P[j], such that

$$c[i] = v[i \uparrow j] \qquad \text{with} \quad 0 \leq j < M$$

where with "↑" we denote the "bit-wise exclusive or" ("bit-wise sum modulo 2"). From the above formula it is clear that in P[j] : c[0] = v[j]. The permutation P[j] can be changed into P[j'] in N steps under control of $s = j \uparrow j'$, each step being controlled by the next digit (in order from left to right) of s. If the corresponding digit of s = 0, we command a step "rotate", i.e. elements i from the lower half (i.e. $i < M1$) send their contents to their even receiver and elements from the upper half (i.e. $i \geq M1$) send their contents to their odd receiver. If the corresponding digit of s is = 1, we command the step "rotate and invert", i.e. elements from the lower half send to their odd receiver and elements from the upper half send to their even receiver. In N steps all the digits of s have been processed, the state P[j] has been changed into P[j'].

Hyperfast Fourier Transform.

With $d = \exp(2\pi i/M)$ we have to compute

$$b[j] = \sum_{i=0}^{M-1} a[i] * d^{i*j} \qquad \text{for } 0 \leq j < M \quad .$$

Our initial state is c[i] = a[i] for $0 \leq i < M$. For h running from 0 through N-1 we do N times the following step.

Each element i sends its contents c[i] to both its receivers, i.e. c[k] and c[k+M1] arrive both at elements 2k and 2k+1. All even receivers perform simultaneously the assignment

$$c[2k] := c[k] + c[k+M1]$$

while odd receivers perform the assignment

$$c[2k+1] := (c[k] - c[k+M1]) * f(k,h)$$

where $f(k,h) = d^r$ and the value of r is formed by forcing the h right-most binary digits of k to zero.

The determination of the constants f(k,h) can be accomplished by extending the contents of each element i with a variable $e[i] = f(k,h)$. They can be initialized for h = 0; after the adjustment of the c-values, the e-values can be adjusted simultaneously by

$$e[2k] := e[k]^2$$
$$e[2k+1] := e[k]^2$$

i.e. the lower half senders have to broadcast their e-value to both their receivers, who square the received value to form their new value of e.

After these N steps $c[i] = b[mir(i)]$

where the binary representation of mir(i) consists of the digits of the N-bit binary representation of i, but in the reverse order.

If we want to reduce the traffic along the paths and to do away with the squaring needed to form the new e-value, we can store in each element the N f(k,h)-values: they are constants!

In either case, the Hyperfast Fourier Transform uses the paths of the connection pattern for one-way traffic only.

Autocorrelation coefficients.

After a Fourier Analysis as described above, each element multiplies its c-value by $\bar{c}$, its complex conjugate. Then, again in N steps not basicly different from the above, a Fourier Synthesis is performed. The Synthesis, however, requires traffic along the paths in opposite direction.

Sorting.

The concatenation of two sequences of length K, sorted in opposite directions, is a "bitone cycle of length 2K". A bitone cycle is a cyclic arrangement of values with only one local maximum and only one local minimum. In contrast to the sorted sequence, the bitone cycle is an undirected object.

A bitone cycle of length 2K is easily split into two bitone cycles of length K with the property that the maximum element of the left(right)hand cycle is at most equal to the minimum element of the right(left)hand cycle. The split is a directed operation.

In order to create an ordered sequence -say "an upward sequence"- we arrange them into a bitone cycle of length M and perform "an upward split". Thereafter both halves are subjected to an upward sort.

In order to create a bitone cycle of length M the two halves are sorted independently in opposite directions. (Note that, because a bitone cycle is un undirected object, it does not matter whether the lefthand half is sorted into an upward or into a downward sequence!)

The above considerations replace the task of an upward sort of length M by two sorts of length M/2, an upward split followed by two upwards sorts of length M/2. These latter two sorts have already been prepared, because after the split, both halves are bitone cycles!

We can describe the algorithm as follows (comments have been inserted between braces):

```
k:= 0;
while k < N
   do k:= k + 1;
      {at this moment the sequence consists of 2^(N-k) bitone cycles of length
      2^k; mark them "up - down - up - down ....." in the order in which they
      occur}
      h:= k;
      while h > 0
         do {at this moment the sequence consists of 2^(N-h) bitone cycles of
            length 2^h; consecutive bitone cycles with the same marking are
            groupwise ordered according to the marking}
            split the bitone cycles of length 2^h according to their
            marking and give the children the marking of the parent;
            h:= h - 1
      od
od
```

With our connection pattern this can be implemented with the aid of the operation "rotate" (see above) and the operation "rotate and select". In the latter operation, each element sends its contents to both its receivers, but, depending on the marking of the pair, the even receiver selects the smallest value and the odd receiver selects the largest (upwards) or the other way round (downwards).

The program then becomes

```
k:= 0;
while k < N
   do k:= k + 1; h:= N - k; set marking;
      while h > 0 do rotate; h:= h - 1 od;
      h:= k;
      while h > 0 do rotate and select; h:= h - 1 od
   od
```

Each receiver(pair) can set its marking "up" when the k-th bit from the right of the binary representation of its ordinal number is = 0 (for the case k = N, this digit is = 0 for all receivers) and "down" if that bit is equal to 1.

From the program it follows that the M elements are sorted in N^2 steps. Again the connection pattern is used for one-way traffic only.

13th June 1973 Edsger W.Dijkstra