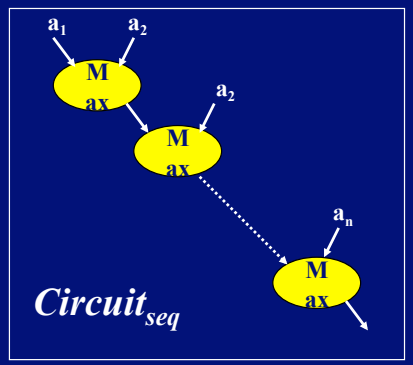
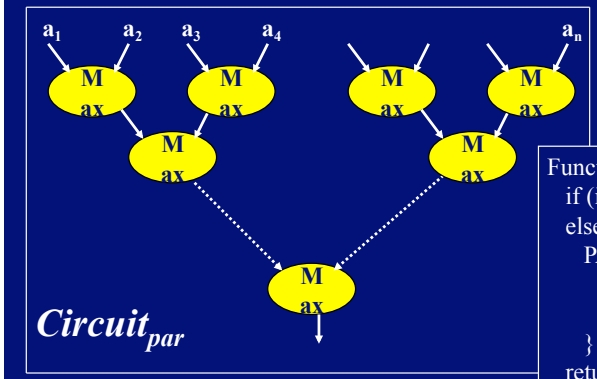
Basic serial circuit




```
res := a1 ;
For i :=2 .. n do
   res := Max ( res, an ) ;
Return res;
```

Time = n #procs = n units

Faster with Parallelism




```
Function max2(ai .. ak ) {
  if (i == k) return ai ;
  else {
    PARALLEL {
      rl = max2(ai .. a(i+k)/2 ) ;
      rh = max2(a(i+k)/2+1 .. ak) ;
    }
  return Max( rl, rh ) ;
}
```

Time = $\log_2$ n #procs = n units

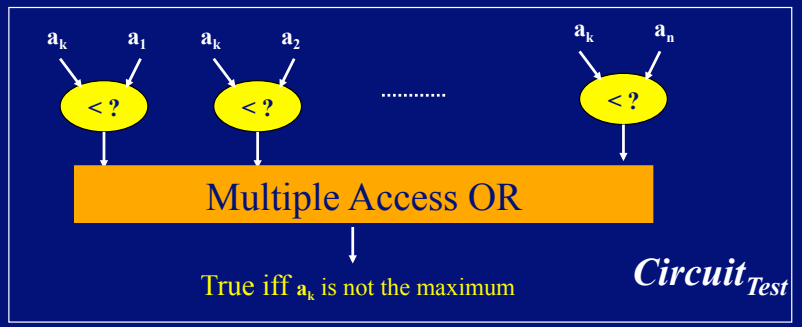
May Multiple Access help ?

•! Taking benefit of multiple access :

logical or of n bits in constant time

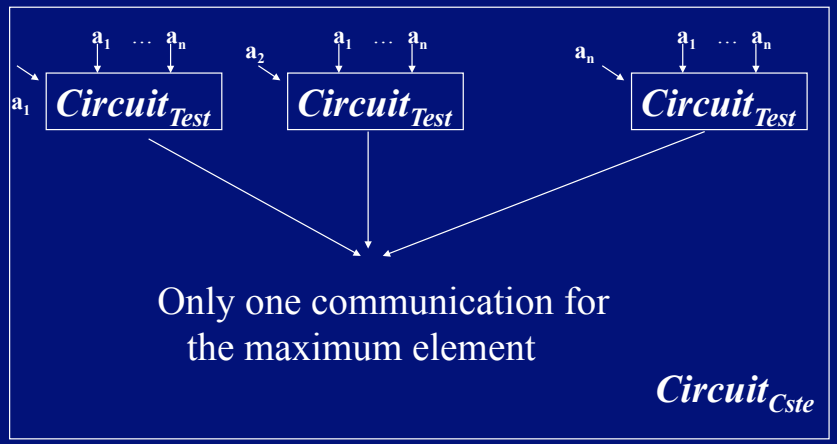
Ultrafast algorithm for testing the maximum

$a_k = \text{Max}(a_1 .. a_n)$ $a_k \geq a_i$ for i=1..n, i /= k



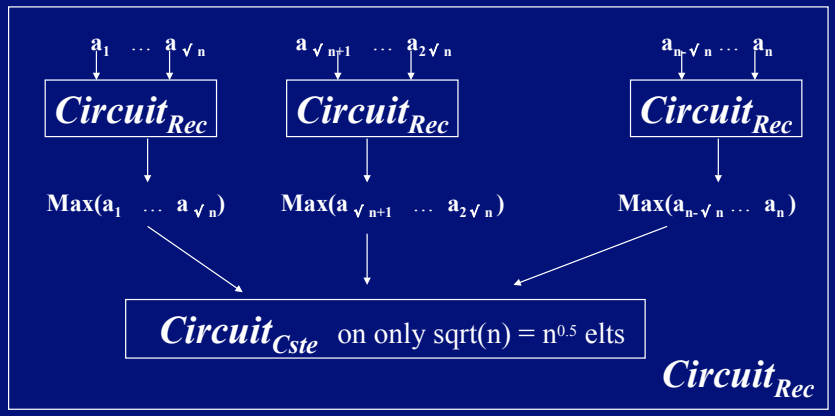
Time(n) = O(1) #units = O(n)

Application: computing the maximum



Time(n) = O(1) **#units = $O(n^2)$**

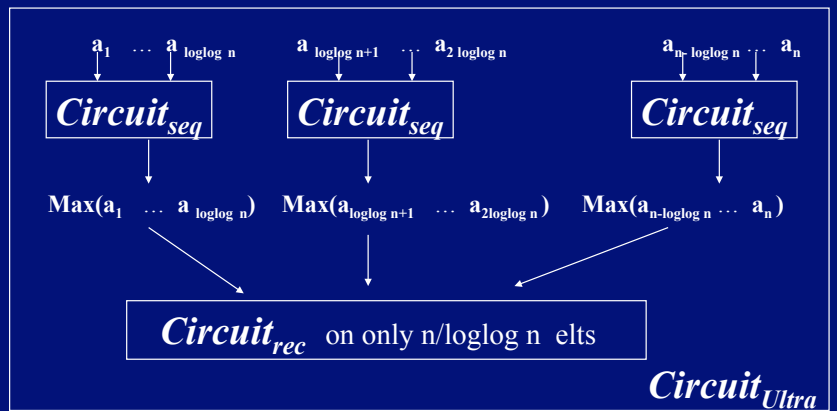
A recursive ultrafast parallel algo



- ! Time(n) = Time($n^{0.5}$) + O(1) = loglog n
- ! #procs (n) = $n^{0.5}$.#procs($n^{0.5}$) = n loglog n

Optimizing the number of units

- ! Take benefit of the sequential algorithm to minimize the number of units



Conclusion : an ultrafast algorithm

Final algorithm : time = loglog n #units=n

Technique used : « cascading »

mixing 3 algorithms to otain an ultrafast one

Important technique in parallelism and software engineering

Wireless communication : technical and theoretical issues