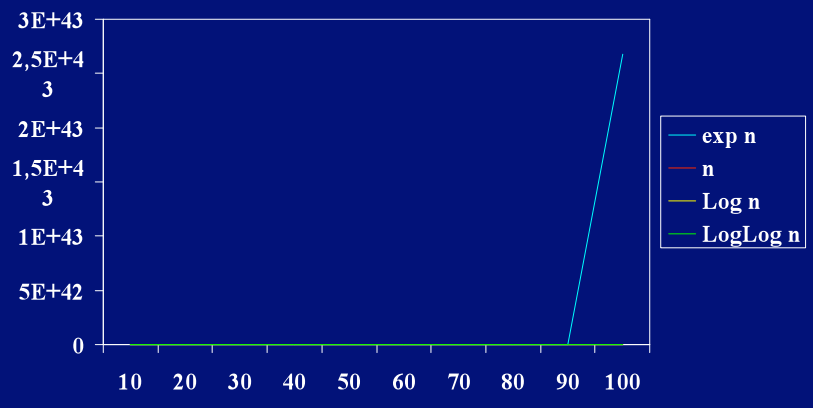
Designing ultra-fast algorithms

- Multiple Access (SDMA / FDMA / CDMA) provides concurrent access to memory in constant time :

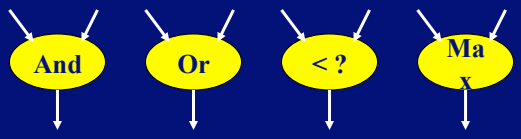
CRCW : Concurrent Read Concurrent Write

Algoritmic costs

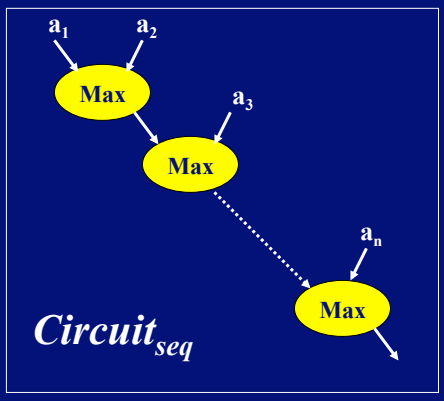


Computing the maximum

- Designing the fastest circuit to compute the maximum
- Input : n elements a_i of an ordered set $<$
  Output : the maximum element
- Available gates :



Basic serial circuit

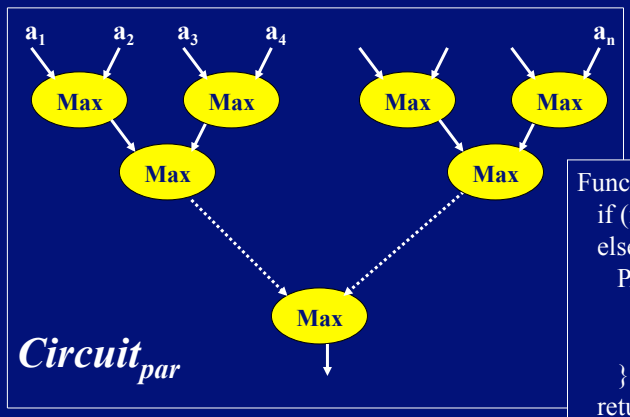



```
res := a1 ;
For i :=2 .. n do
   res := Max ( res, an ) ;
Return res;
```

D=Depth= n W = Work = n

(NB Work and depth are in number of Compare gates)

Faster with Parallelism




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

$D = \log_2 n$ $W = n$

May Multiple Access help ?

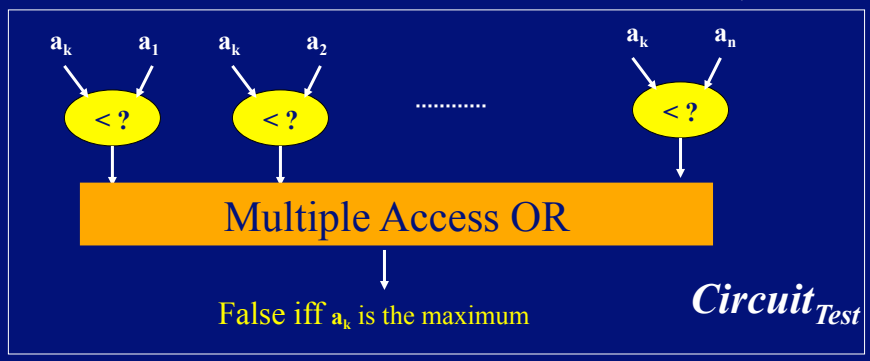
- Taking benefit of multiple access :

logical or of n bits in constant time

Ultrafast algorithm for testing the maximum

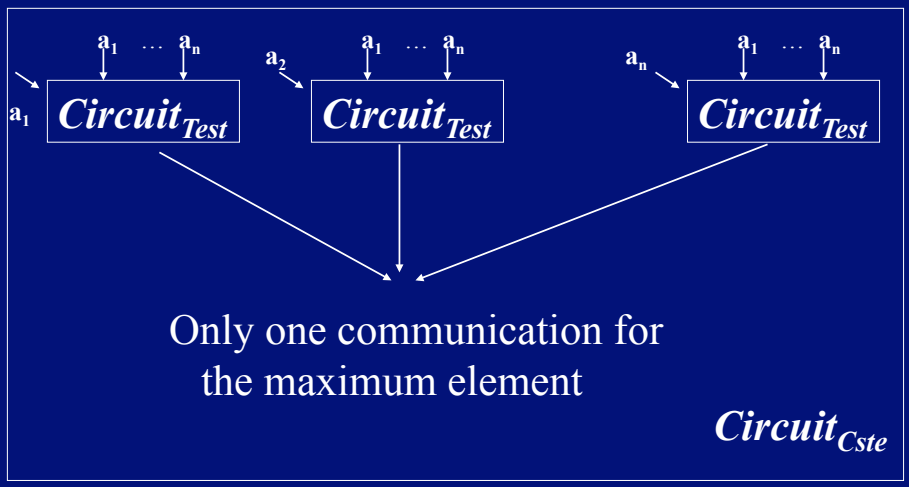
$a_k = \text{Max}(a_1 .. a_n) \Leftrightarrow a_k \geq a_i$ for $i \neq k \Leftrightarrow \text{AND}_{i \neq k}(a_k \geq a_i)$

$\Leftrightarrow \text{NOT}(\text{OR}_{i \neq k}(a_k < a_i))$



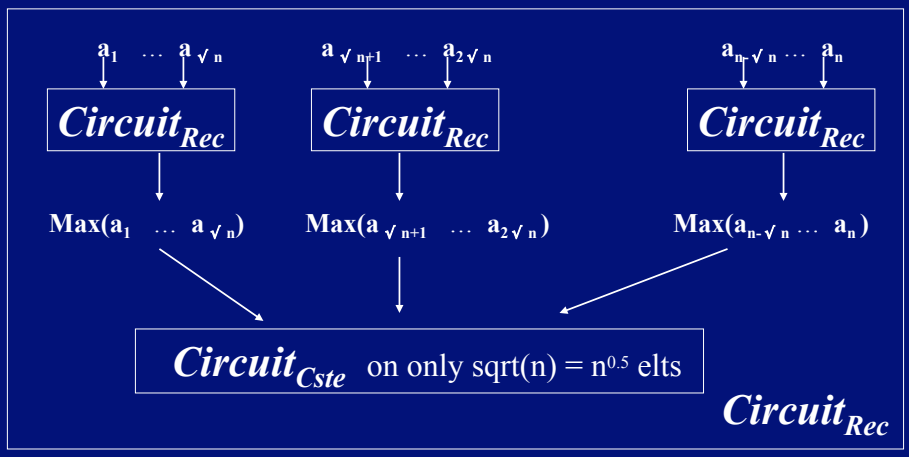
D= 1 ☺ W= n

Application: computing the maximum



D= 1 ☺ **W= n^2** ☹

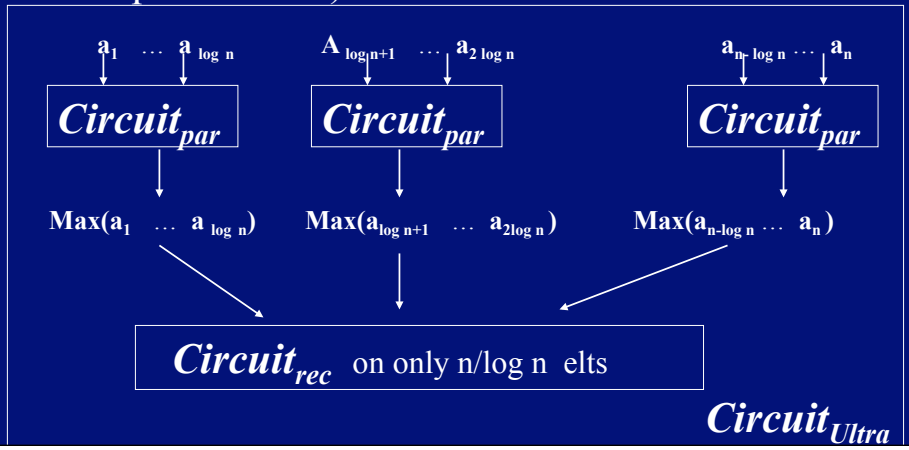
A recursive ultrafast parallel algo



- $D(n) = D(n^{0.5}) + 1 = \text{loglog } n$
- $W(n) = n^{0.5}.\#procs(n^{0.5}) = n \text{ loglog } n$

Optimizing the number of units

- Take benefit of the parallel algorithm to minimize the number of units (could be the sequential one)



Conclusion : an ultrafast algorithm

Final algorithm : depth= loglog n $_{+ \text{logloglog n}}$ ☺

work= n $+ _{\text{n loglogn / log n}}$ ☺

Technique used : « cascading »

mixing 3 algorithms to otain an ultrafast one !

Fundamental technique for parallel algorithms design and in software engineering too.

Both theoretical and practical issues.