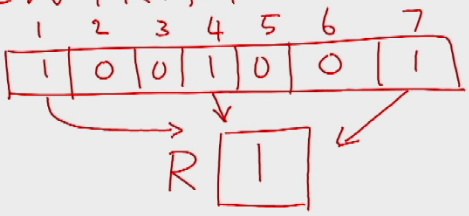
OR of n bits on a TOLERANT CRCW PRAM



OR of *n* bits on TOLERANT CRCW PRAM

If there is 1-bit in locations 2..n then P_1 has company

Input: Array $A[1 \ldots n]$ of n bits.
Output: A bit R that is the OR of the bits in A.

Step 1: pardo for $I = 1$ $R = 1$;

Step 2: pardo for $1 \leq I \leq n$
if $(I == 1)$ $R = 0$;
else if $(A[I] == 1)$ $R = 0$;

Step 3: pardo for $I = 1$ if $(R == 0$ && $A[1] == 1)$ $R = 1$;

Step 4: return R;

only processor
1st processor compulsorily
only the 1st processor

O(1) time
n processors

Simulations among CRCW Models

- If an algorithm that runs on model A in T time can be simulated on model B in $O(T)$ time, then B is at least as powerful as A; $A \preccurlyeq B$
- Since all CRCW PRAMs are similar except in their write conflict resolution rules, we need to bother about the simulation of writes only
- Suppose in a particular concurrent write of the simulated PRAM, processor i wants to write value $V[i]$ in cell $M[i]$

ARBITRARY $\preccurlyeq$ PRIORITY

- An algorithm written for ARBITRARY can be run on PRIORITY without any change
- ARBITRARY assumes only that in every concurrent write some one processor succeeds, and that is guaranteed by PRIORITY

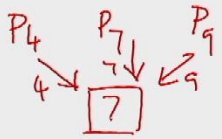
COMMON $\preccurlyeq$ ARBITRARY:

- An algorithm written for COMMON can be run on ARBITRARY without any change

COLLISION $\preccurlyeq$ ARBITRARY:

- Each step of collision can be simulated by three steps of ARBITRARY.
- (1) Make every processor i, write the integer i in cell $M[i]$
- (2) Make every processor i, if it had failed in step 1, write a special collision symbol in cell $M[i]$
- (3) With every processor i that succeeded in step 1, if cell $M[i]$ did not change its contents during step 2, write $V[i]$ in cell $M[i]$

COLLISION ≼ ARBITRARY:



- Each step of collision can be simulated by three steps of ARBITRARY.
- (1) Make every processor i, write the integer i in cell $M[i]$
- (2) Make every processor i, if it had failed in step 1, write a special collision symbol in cell $M[i]$
- (3) With every processor i that succeeded in step 1, if cell $M[i]$ did not change its contents during step 2, write $V[i]$ in cell $M[i]$

COLLISION ≼ ARBITRARY:

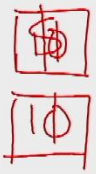
T : COLLISION

3T : ARBITRARY

- Each step of collision can be simulated by three steps of ARBITRARY.
- (1) Make every processor i, write the integer i in cell $M[i]$
- (2) Make every processor i, if it had failed in step 1, write a special collision symbol in cell $M[i]$
- (3) With every processor i that succeeded in step 1, if cell $M[i]$ did not change its contents during step 2, write $V[i]$ in cell $M[i]$

TOLERANT ≼ COLLISION:

- Assume that there is an auxiliary cell attached to each original cell
- First let the processors attempt to increment their corresponding auxiliary cells
- (If the auxiliary cell contains $-1, decrement)
- If they succeed they go on to write in the original cell
- Otherwise, they should do nothing



CRCW PRAMs

- All the above relations are in fact strict
- can be replaced by $\prec$ in each case
- This is because, in each case, we can find a problem P such that P has a time lower bound of $\Omega(T)$, for some T, on the weaker model, but can be solved in $o(T)$ time on the stronger model.

ARBITRARY ≤ PRIORITY

P_1 : $O(T)$ PRIORITY

$\Omega(T_1)$ ARBITRARY where $T_1 : \omega(T)$

- An algorithm written for ARBITRARY can be run on PRIORITY without any change
- ARBITRARY assumes only that in every concurrent write some one processor succeeds, and that is guaranteed by PRIORITY

COMMON vs COLLISION, TOLERANT

- COMMON is incomparable with COLLISION or TOLERANT
- there are problems P1 and P2 such that P1 can be solved faster on COMMON than on COLLISION, and P2 can be solved faster on TOLERANT than on COMMON

Self Simulating

✓PRIORITY

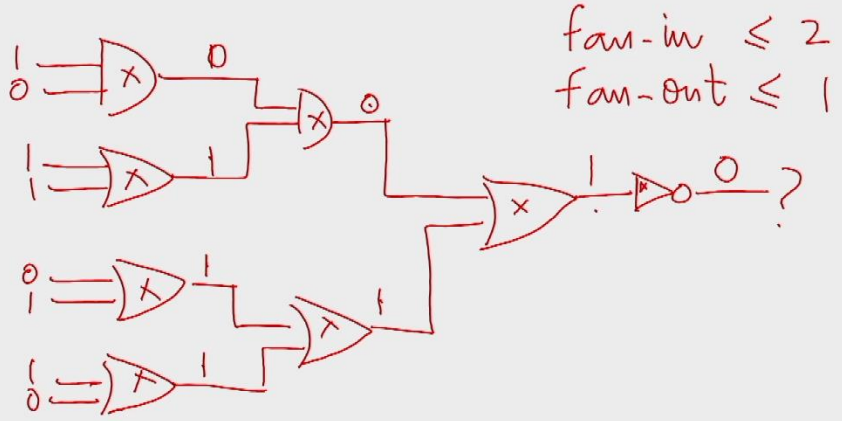
✓ARBITRARY

✓COMMON

COLLISION

TOLERANT

Boolean Circuit evaluation on EREW PRAM

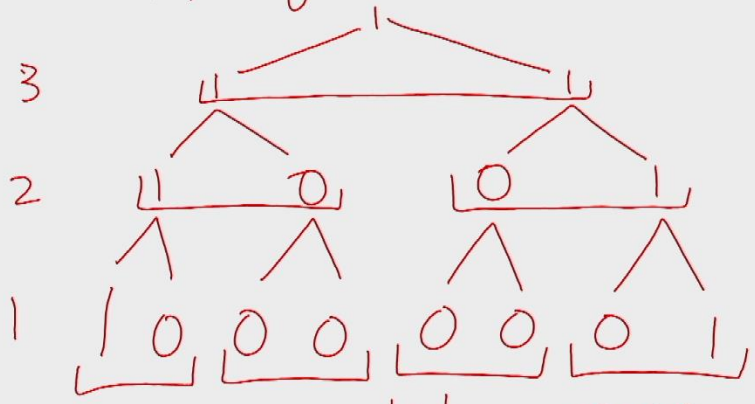


Time = depth of the circuit

instructions = O(size of the ckt)

EREW PRAM

OR of n bit on an EREW PRAM



1 processor

2 processors

4 processors

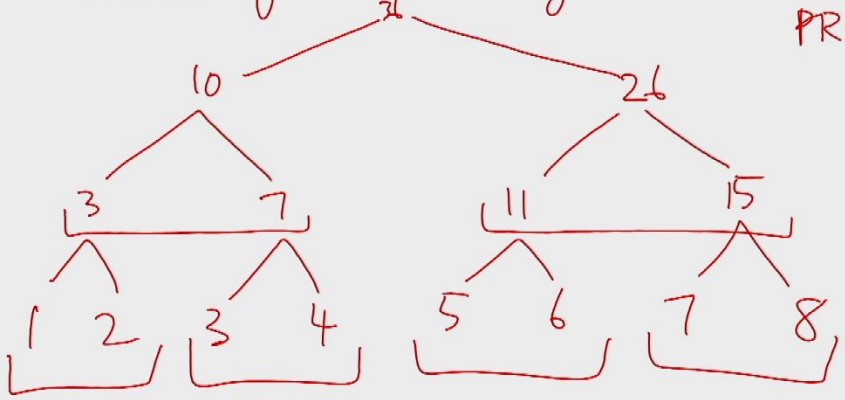
n bit array

O(log n) time

n/2 processors

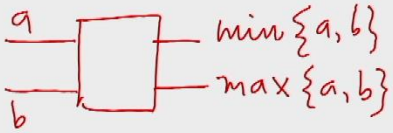
Sum of n integers

EREW PRAM



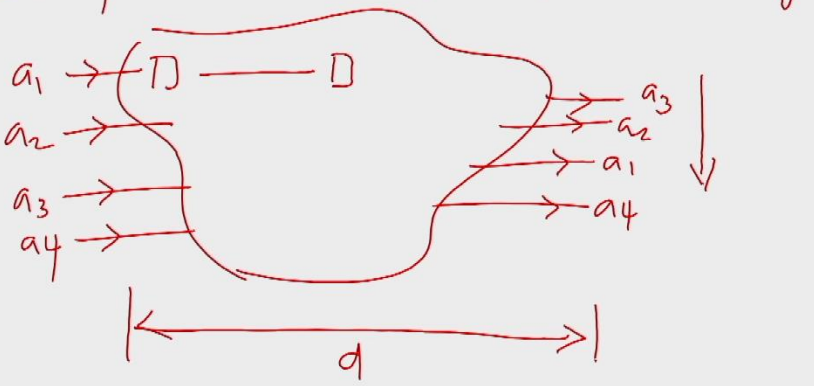
Comparator Networks

Comparator



Network of comparators
- Merging
- Sorting

Comparator n/w for sorting

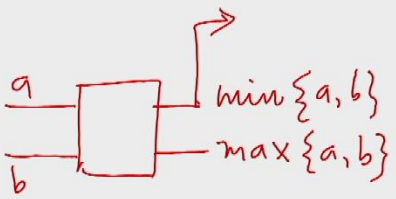


Exercise

Show that a comparator network of depth d & cost n can be simulated on a EREW PRAM of size n in time $O(d)$

Comparator Networks

Comparator



Network of comparators

- Merging
- Sorting

one output drives only one input