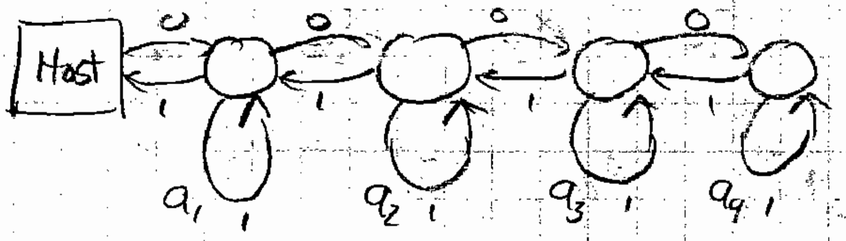
## Retiming

Recall Systolic Conversion Theorem: G can be retimed to be systolic if no neg-wt cycles in G-1

Ex. Priority queue
- Insert(x)
- Extract-Min - returns and deletes min elem.

Semisystolic design



Insert(x)
- Host 'broadcasts x.
- Each processor i:

```
If $x > a_i$, do nothing
If $x \le a_i$, send $a_i$ right
    if no value received from left
        then replace $a_i$ with $x$
        else replace $a_i$ with left value.
```

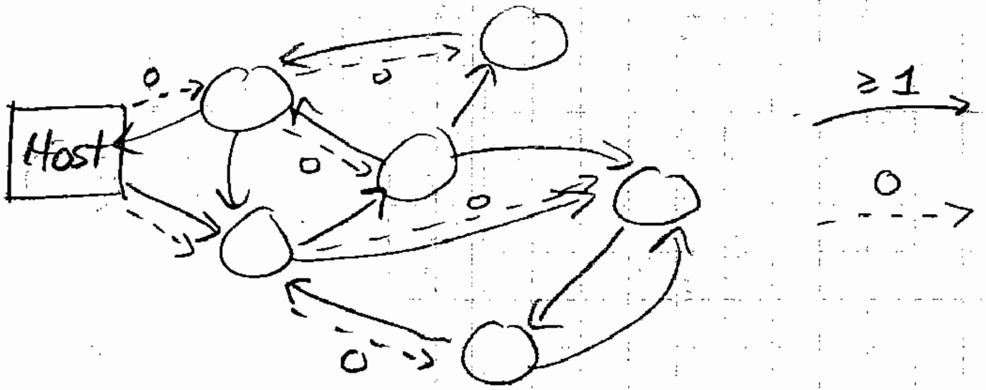
Extract-Min
- Host broadcasts "shift left"
- Each processor shifts its value left, accepts and stores value from right.

1 clock tick per operation.

2G-1 has no neg-wt cycles.
∴ 2G can be retimed to be systolic.

## Design methodology

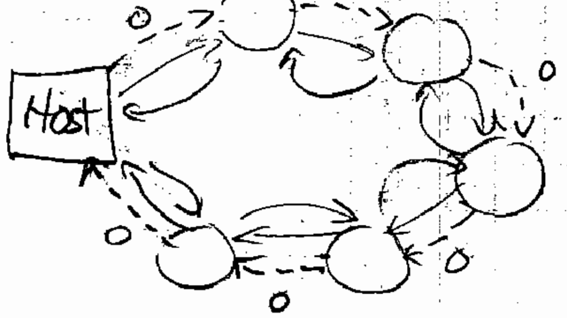
1. Build systolic circuit G to solve part of problem
2. Augment G with global comb. logic following breadth-first spanning tree of the undirected version of G, either toward or away from host, forming G'.



3. Theorem. 2G' − 1 contains no neg-wt cycles $\Rightarrow$ retime 2G' to be systolic. Note: 2G' has same topology as G.

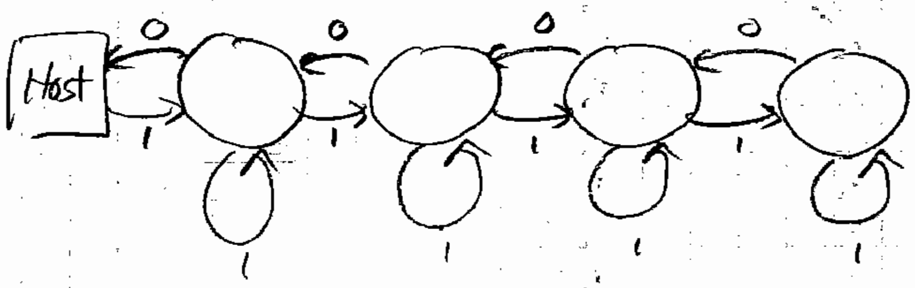
Proof. Consider cycle in 2G' − 1. Each 0-wt edge in G' takes path 1 step further from host according to breadth-first distance. Each ≥1-wt edge takes path ≤ 1 step closer. Thus, at most 1/2 edges in cycle have −1 wt in 2G' − 1, and rest have ≥ 1 wt. ☒

Ex. Priority queue with search



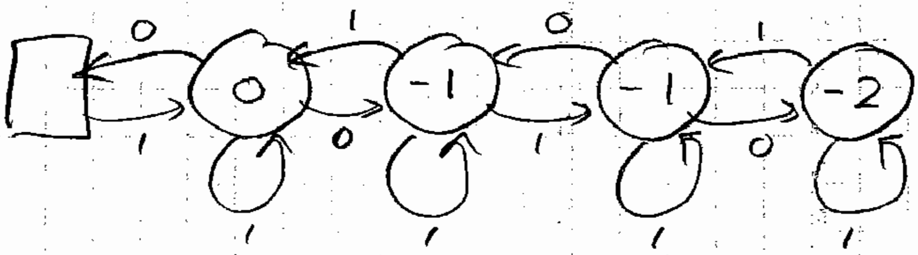
Broadcast + accumulate
- no const slowdown can be retimed to make systolic.
- must either broadcast or accumulate, not both for const. slowdow

Recall palindrome recognizer:



2G can be retimed to make systolic.

But, can achieve almost the same with <u>no</u> slowdown:



Clock period = longest path of comb. rippling.

<u>Theorem</u>: Retime to achieve clock period of c iff $G - 1/c$ has no neg-wt. cycles.

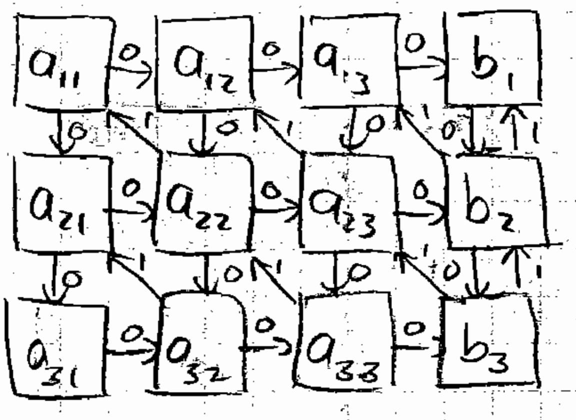
Proof. Homework ☒

6.896
2/25/04
L7.4

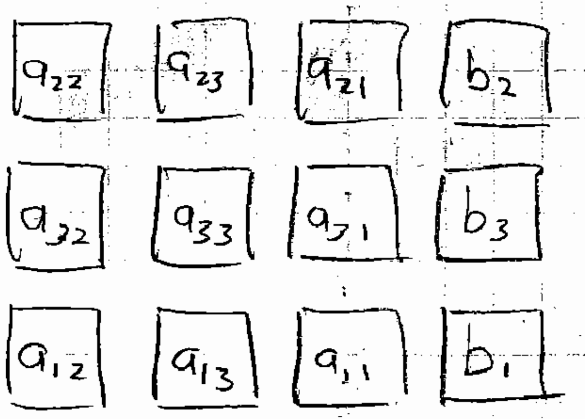
# Gaussian Elimination (revisited)

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - \frac{a_{ik}^{(k-1)} \cdot a_{kj}^{(k-1)}}{a_{kk}^{(k-1)}}$$

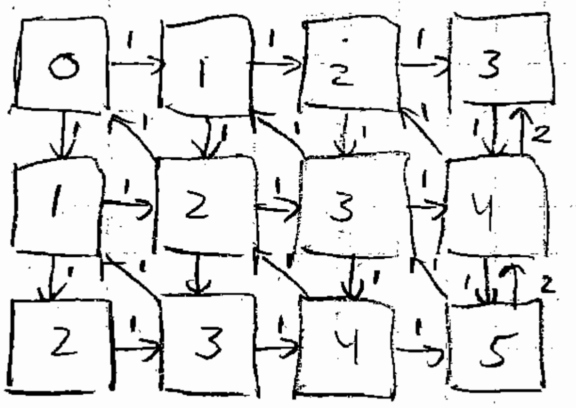
$A^{(0)}$



$A^{(1)}$



Retime 3G

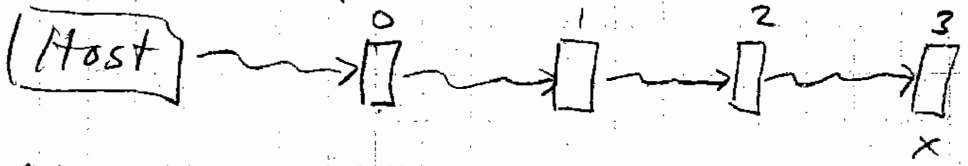


## Resetting

Set all state to ~~predefined~~ values in 1 clock tick.

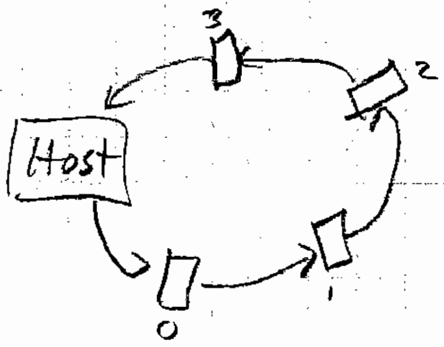
Idea 1: Broadcast, slowdown, retime
Problem: Can't use on circuits such as palindrome recognizer.

Idea 2: Sup. ≥d latches on every path from host to latch x.
Then, after reset, x's values are fixed for d steps.



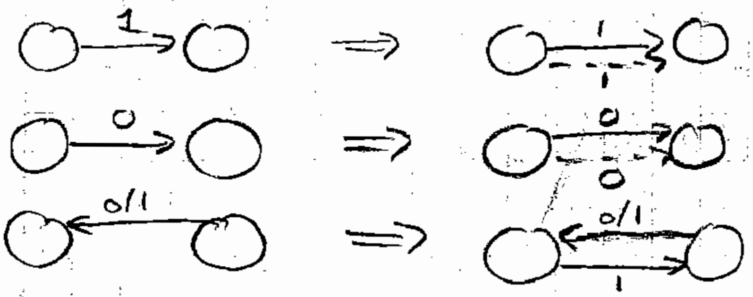
Thus, reset of x at time $t_0$ can be delayed up to d time steps. Then, reset to value x should have at time $t_0+d$.

Bug!



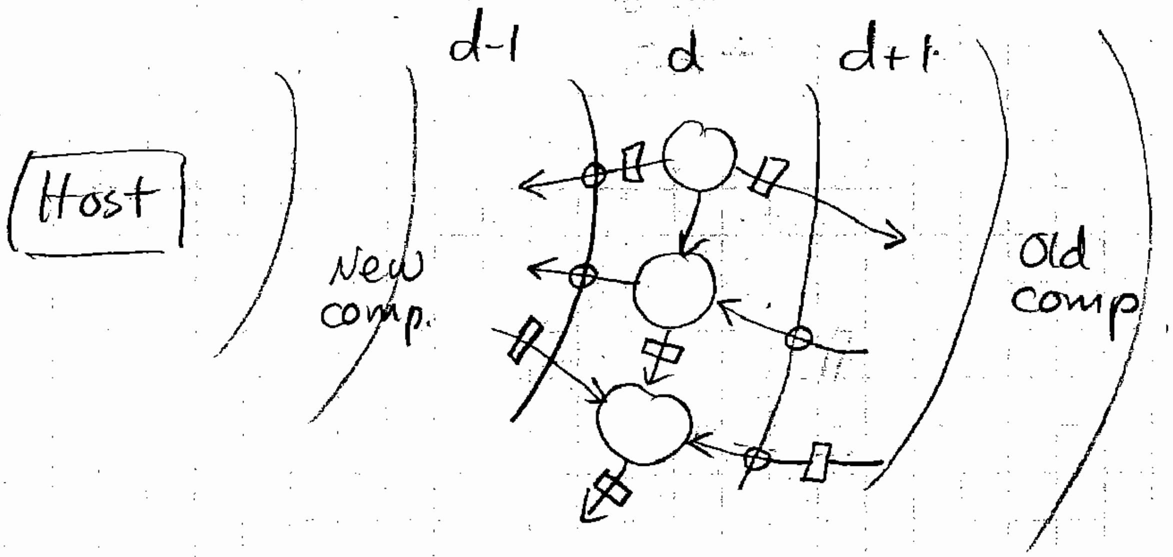
Latch 3 doesn't get reset till time $t_0+3$, but old values still output

Idea 3: Propagate wavefront of resets along graph edges (WLOG, each edge weight 0 or 1):



Create shortest-path tree from host in this graph.
Reset latches at level d to precomputed value at time $t_0+d$.

Reset wavefront.



Bug! Back contamination.

Fix: At $t_0+d+1$, edges from level $d+1$ to level $d$ are altered to act as if reset at time $t_0+d$.

Note: Reset edges do not affect systolic conversion.

Application Palindrome recognizer with reset.