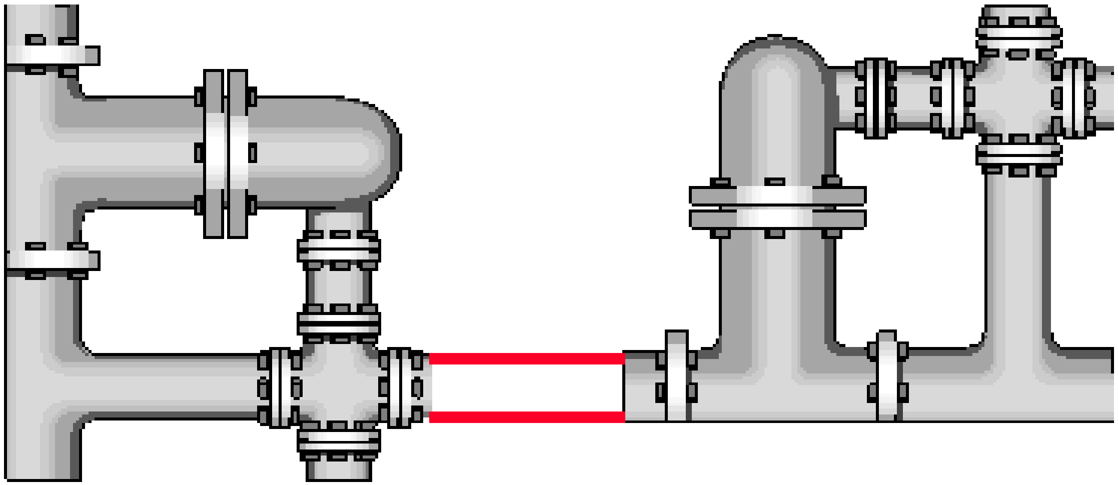
EL318: Lecture 3-1

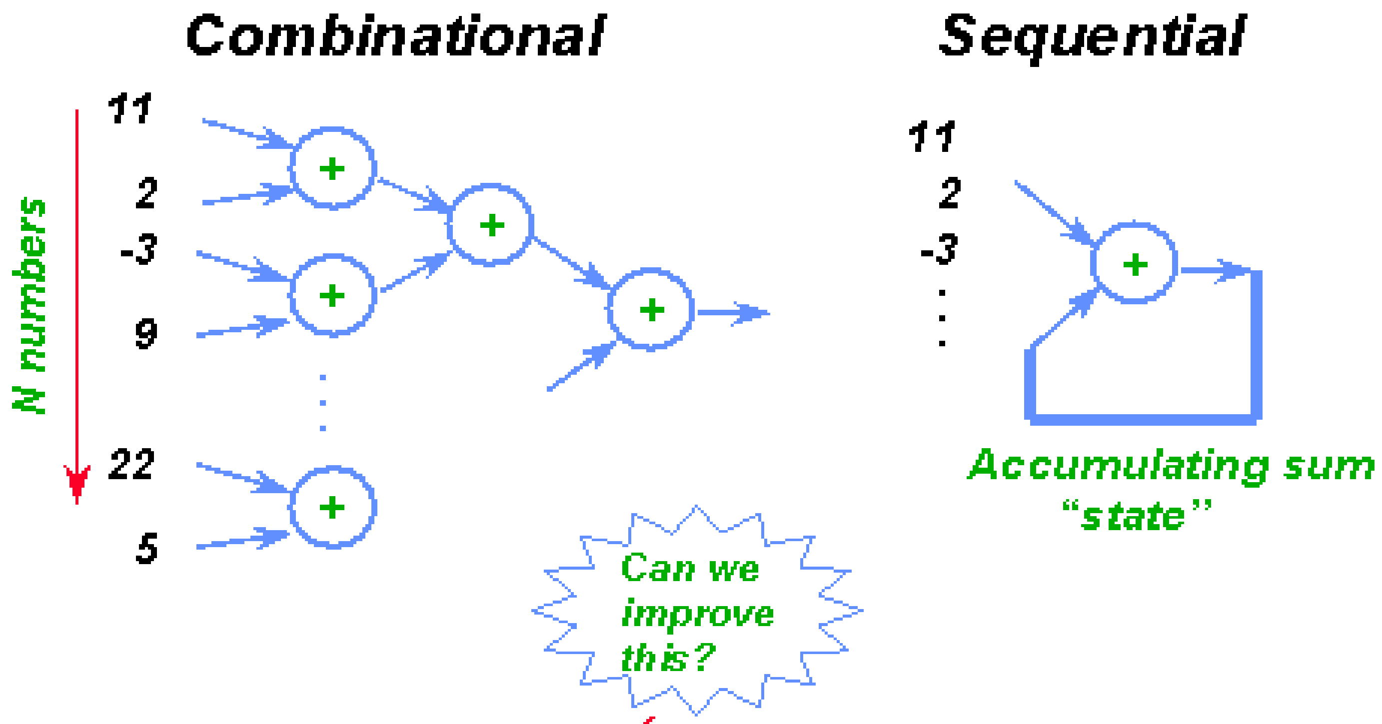
Pipelining

With thanks to Srini Devadas at MIT and his course *Computation Structures*

Pipelining



Motivation: Adding a column of numbers



Size grows as N
Delay grows as $\log_2 N$

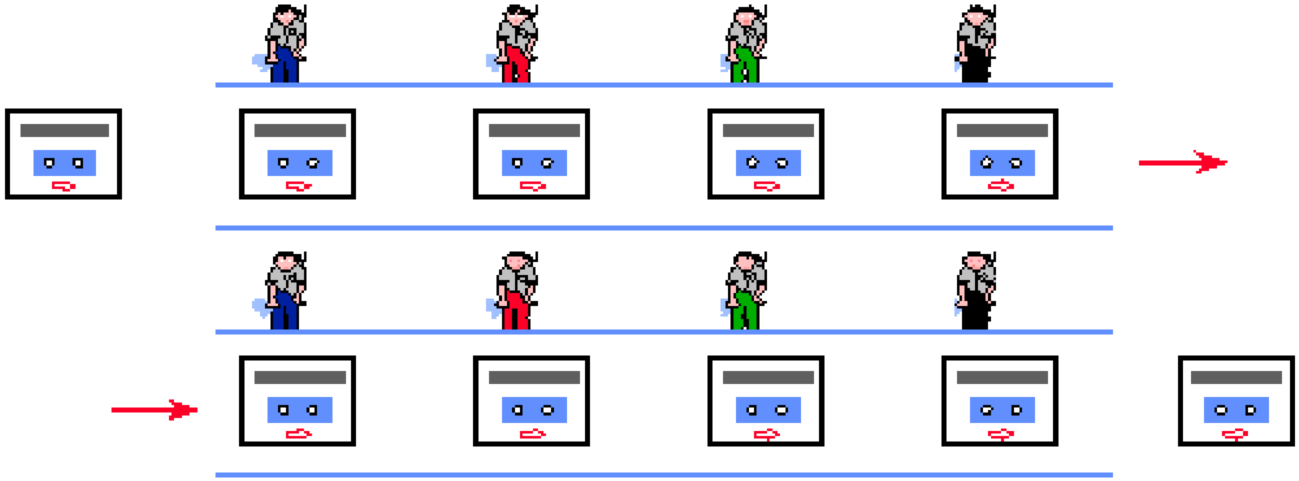
Size grows as 1
Delay grows as N

Brent's Theorem

Let A be a given algorithm with a parallel computation time of t. Suppose that A involves a total number of m computational operations. Then A can be implemented using p processors in $O(m/p + t)$ parallel time.

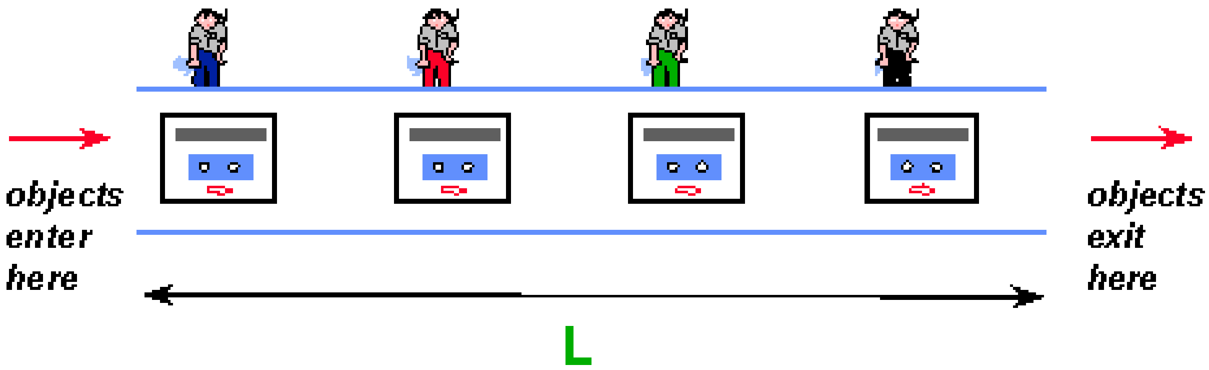
Proof: Let $m(i)$ be the number of computational operations performed (in parallel) in step i of A, the algorithm provided. Using p processors this can be simulated in $m(i)/p+1$ time. Now summing over all i, $1 \leq i \leq p$, we get the stated parallel computation time because $m=m(1)+m(2)+\cdots+m(p)$.

Observe... An Assembly Line



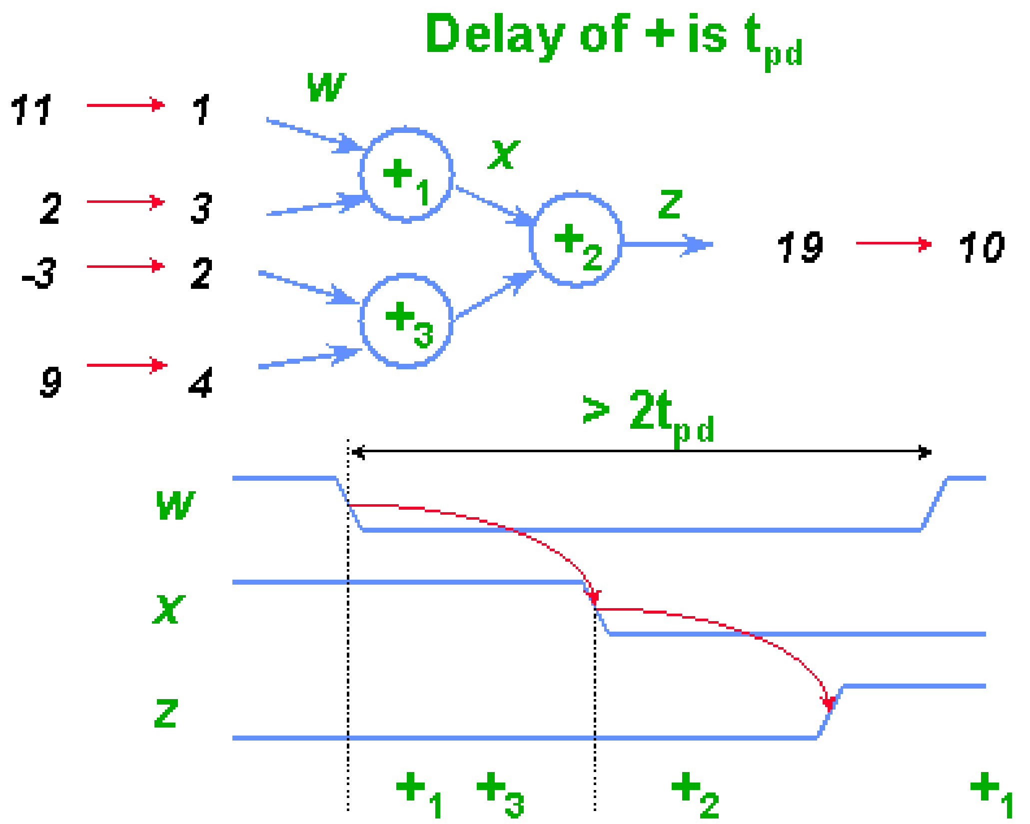
- Length L objects (and workers), moves at rate R objects/minute
- Each task should take at most _____ minutes
- An object is processed in _________ minutes
- Workers always have an object in front of them

Definitions

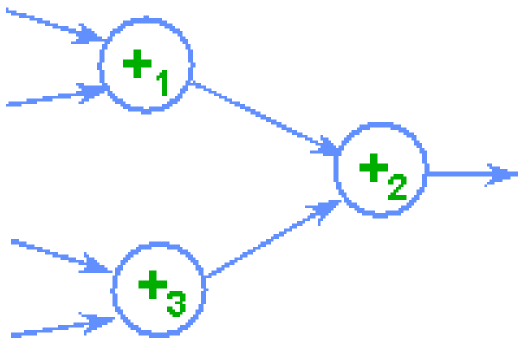


- **Latency**: Time for one object to pass through the system is **L/R**
- **Throughput**: Rate of objects going through is **R**

Adding Numbers: A Closer Look



Combinational Throughput and Latency

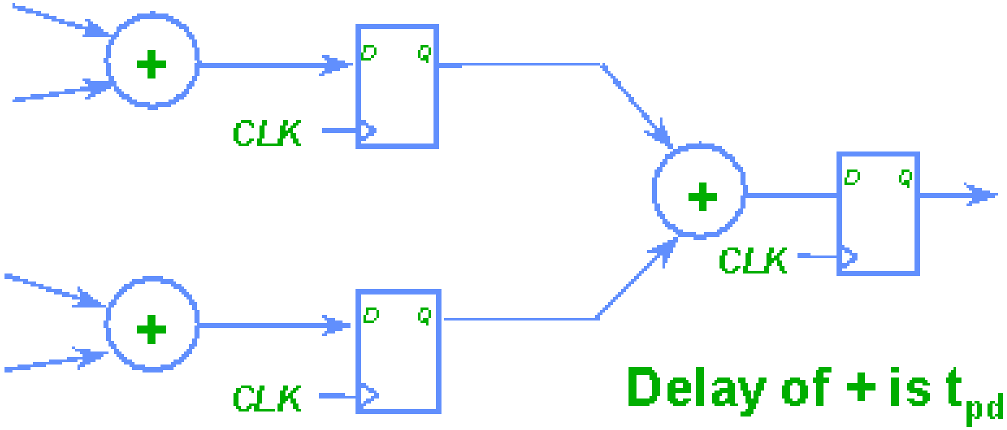


Delay of + is t_{pd}

Latency = $2\ t_{pd}$

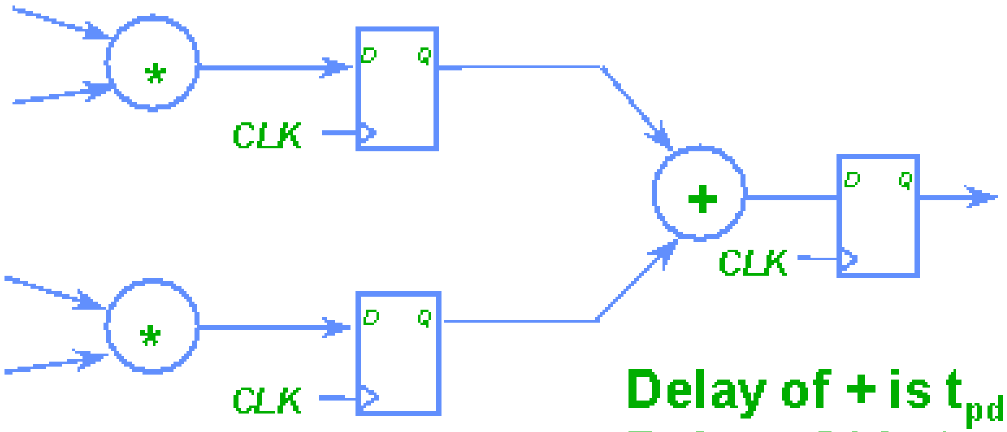
Throughput = $\dfrac{1}{2\ t_{pd}}$

Pipelined Throughput and Latency



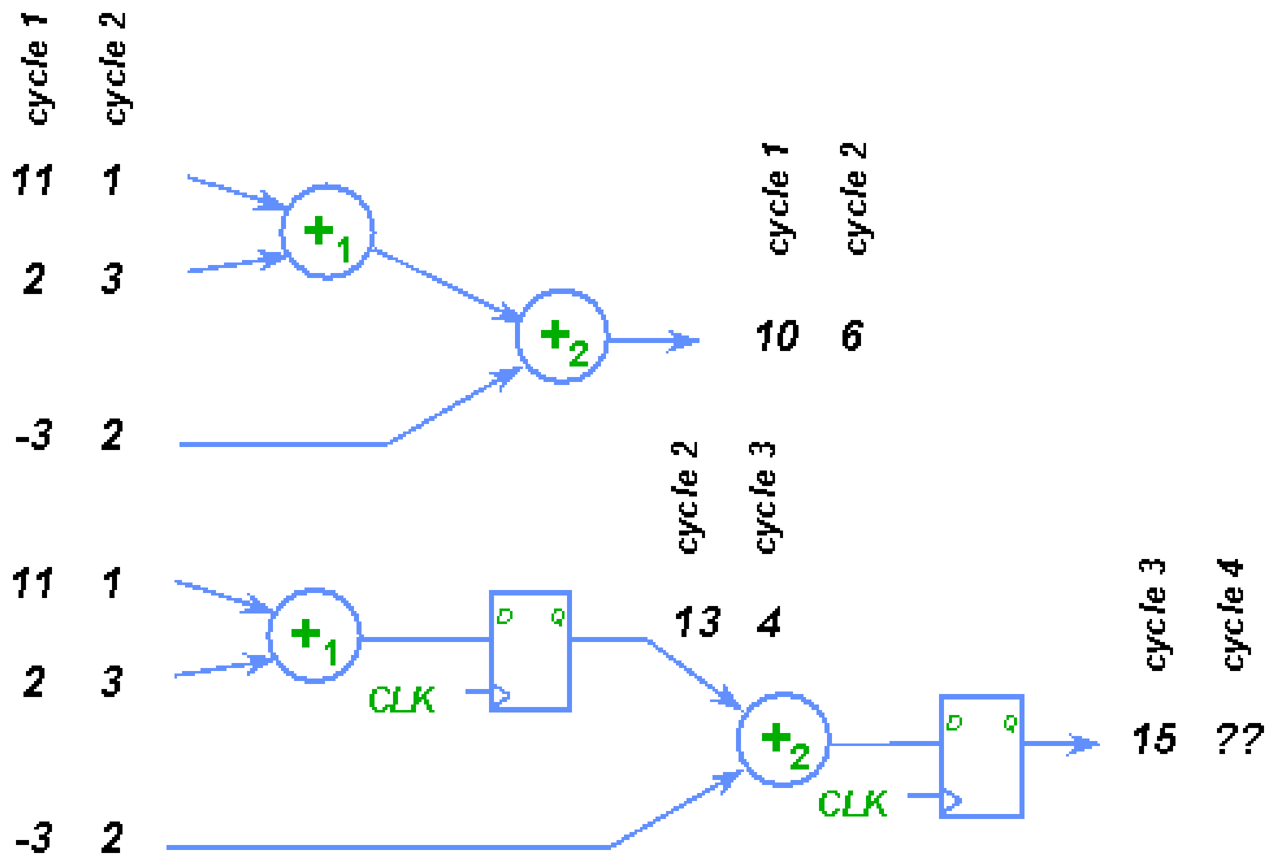
Delay of + is t_{pd}

Inhomogenous Pipeline



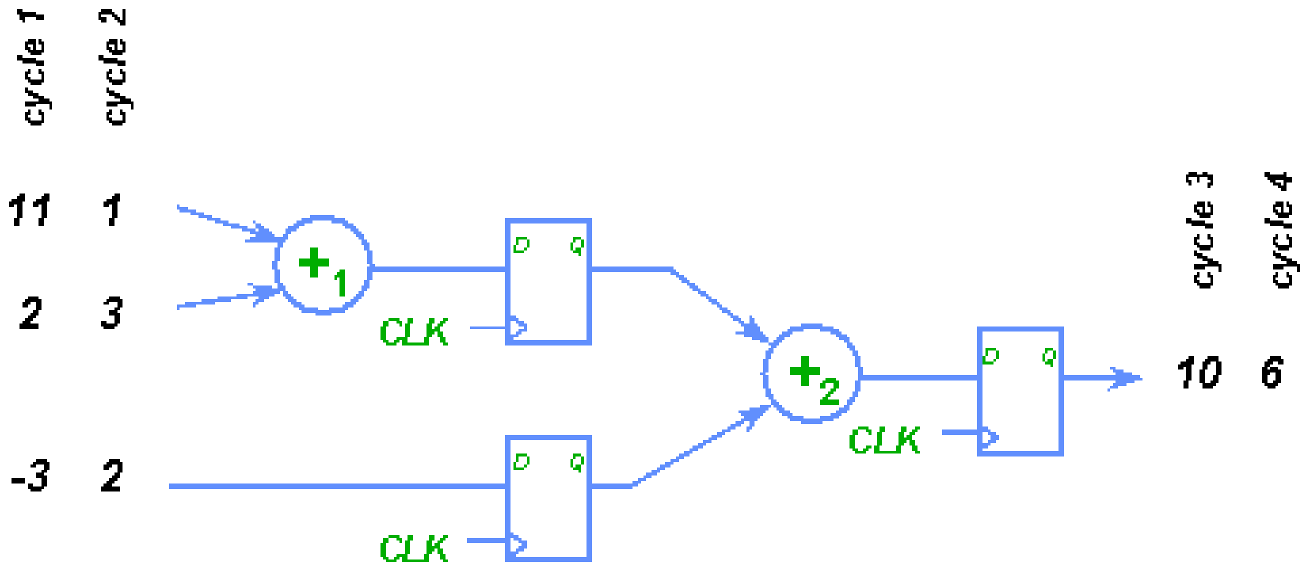
Delay of + is t_{pd1}
Delay of * is t_{pd2}

ILL-Formed Pipelines



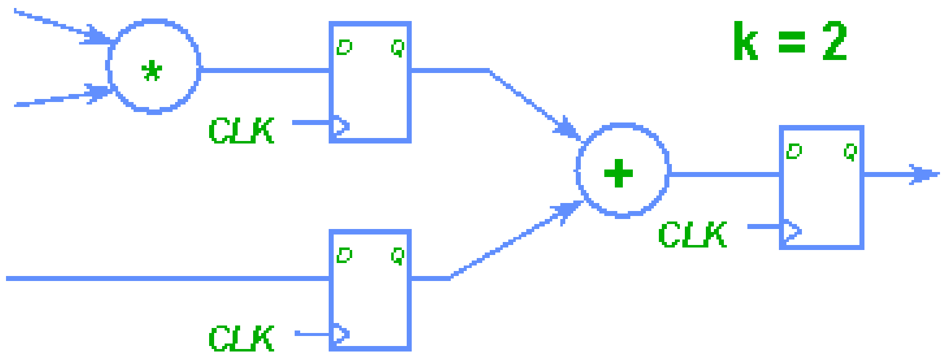
Pipeline Well-Formedness

- **Same number of flip-flops along any path from any input to any output**
  - **Insures that every computational unit sees inputs in phase**



k-Pipelines

- **k flip-flops on each input-output path**
- **Always have flip-flops on each output**



Clock period is maximum propagation delay from flip-flop output (or circuit input) to flip-flop input