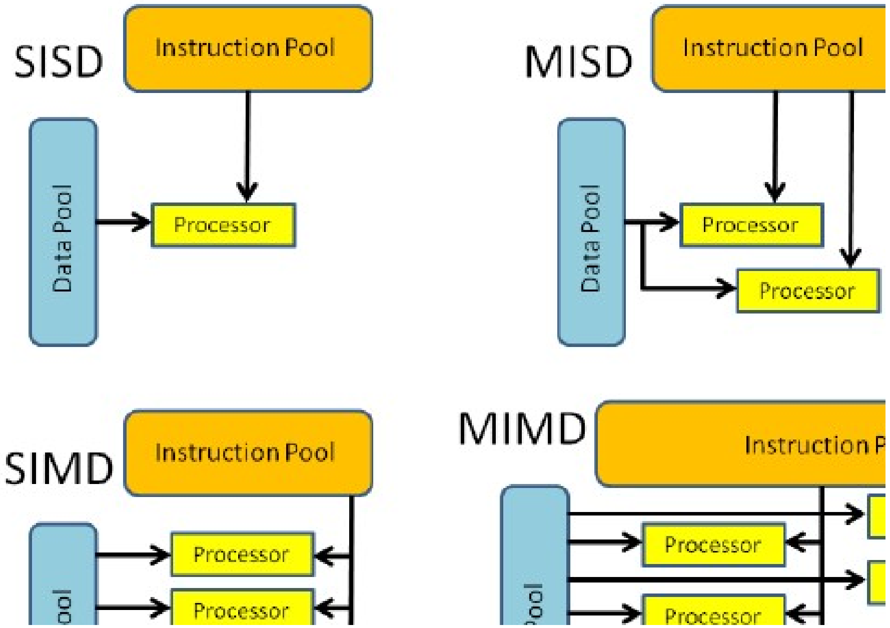
§9 Parallel Algorithms

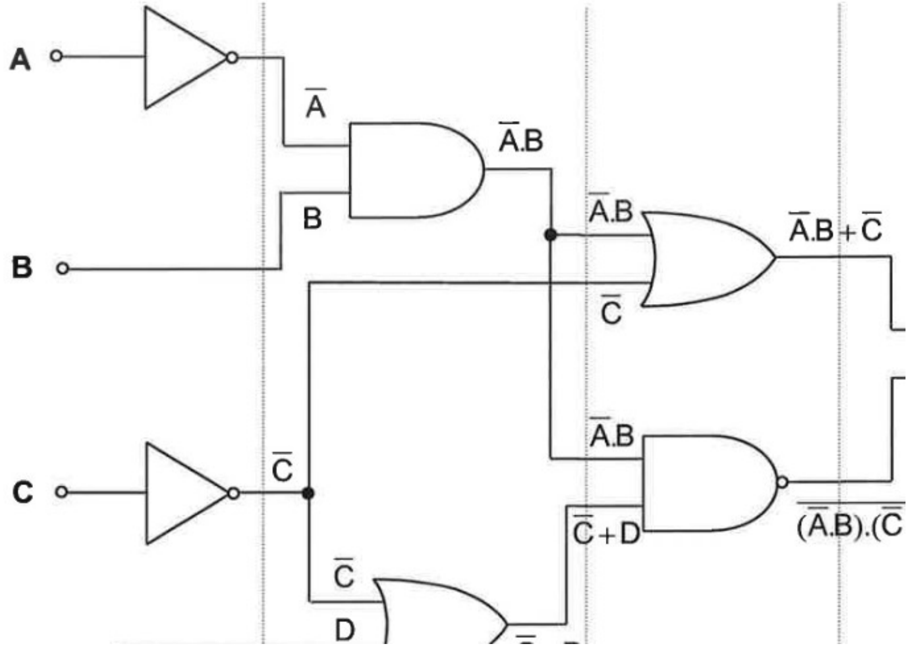
- Classification
- Parallel Prefix
- Graph Reachability
- Carry-Lookahead
- Sorting Networks

Classification

- *fine*-grained parallelism vs. *distributed* computing
- SISD/MISD/SIMD/MIMD
- PRAMs: CRCW/CREW/EREW

Here: <u>algorithms</u>, not programs ("*abstraction*")





Primitives and *Cost* Measures

Size = #binary *gates*

Alternative **Size**
=#cores/#CPUs/#PCs,
→ Communication **Volume,**

Work **:= Span · #CPUs**

Brent's Principle:
Work, Size ≥ ***Time***

(seq.opt.) **Time**

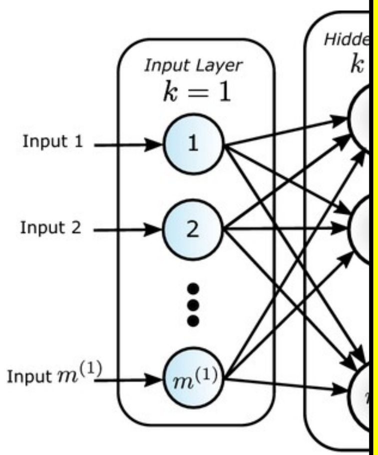
Speedup := Time/Depth

Overhead := Size/Time

Depth = "parallel runtime"

Span

Here: *Circuits*



Lemma:
Any circuit of *binary* gates, <u>depending</u> on N inputs, has
- size ≥ N-1 and
- depth ≥ $\log(N)$

Parallel N-ary Maximum

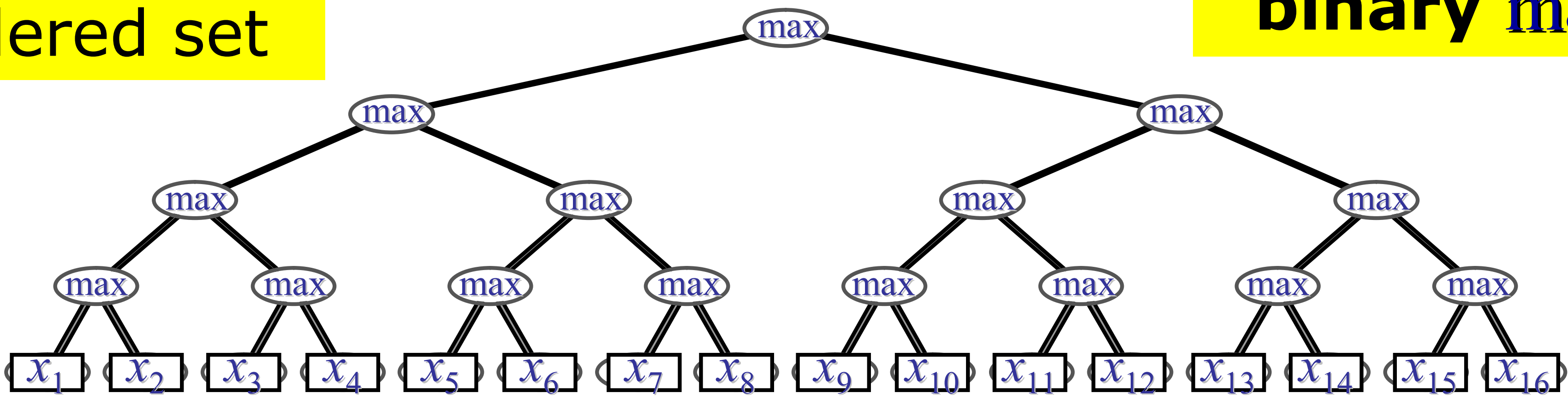
Size = $O(N)$ binary op.s

Depth = $O(\log N)$

$(X,\leq)$ totally ordered set

optimal!

Primitive op: **binary** max



Brent: ***Size* $\geq$ *Time***

(ignore *magnitude* of operands...)

(seq.opt.) **Time** = $O(N)$

Speedup = Time/Depth = $O(N/\log N)$

Overhead = Size/Time = $O(1)$

$$(x_1,\ldots,x_N) \to \max_{1\leq n\leq N} x_n$$

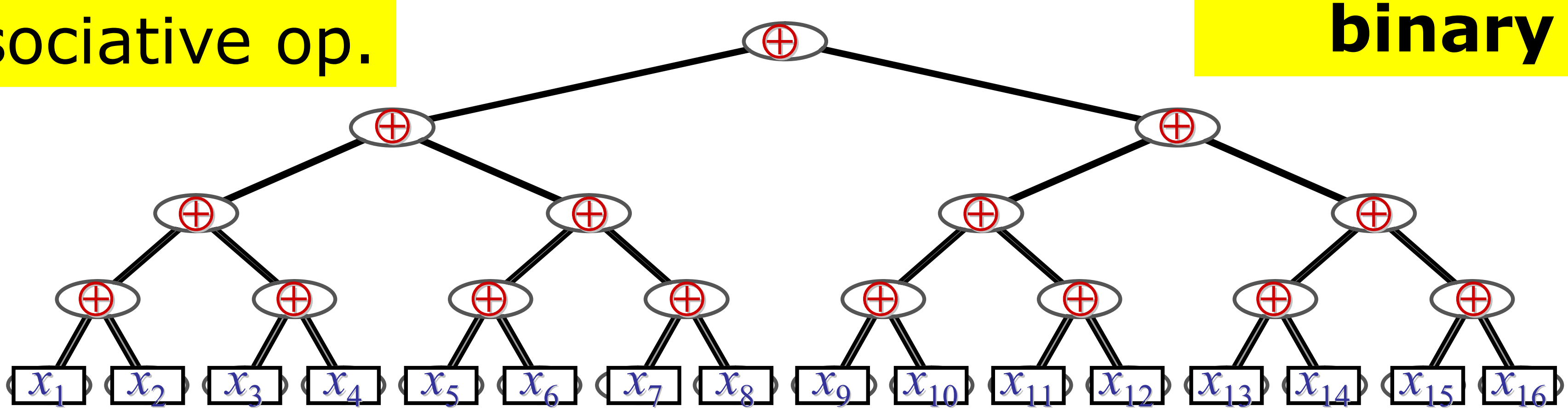
Parallel N-ary *associative* $\oplus$

Size = $O(N)$ binary op.s

Depth = $O(\log N)$

$(X,\oplus)$ set with associative op.

Primitive op: **binary** $\oplus$



$(x_1,\ldots,x_N) \to \bigoplus_{1\le n\le N} x_n$

(seq.opt.) **Time** = $O(N)$

Speedup = Time/Depth = $O(N/\log N)$

Overhead = Size/Time = $O(1)$

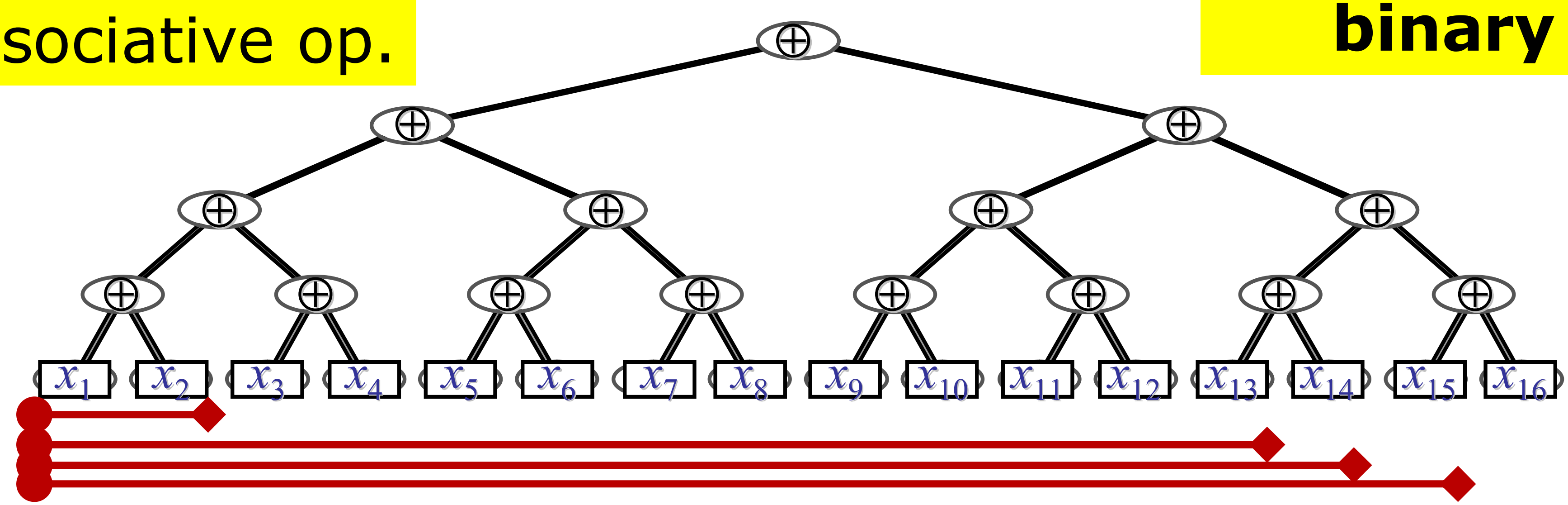
(Generalized) Prefix Sum / *Scan*

Size = $O(N^2)$ binary op.s

Depth = $O(\log N)$

$(X,\oplus)$ set with associative op.

Primitive op: **binary** $\oplus$



(seq.opt.) **Time** = $O(N)$

Speedup = Time/Depth = $O(N/\log N)$

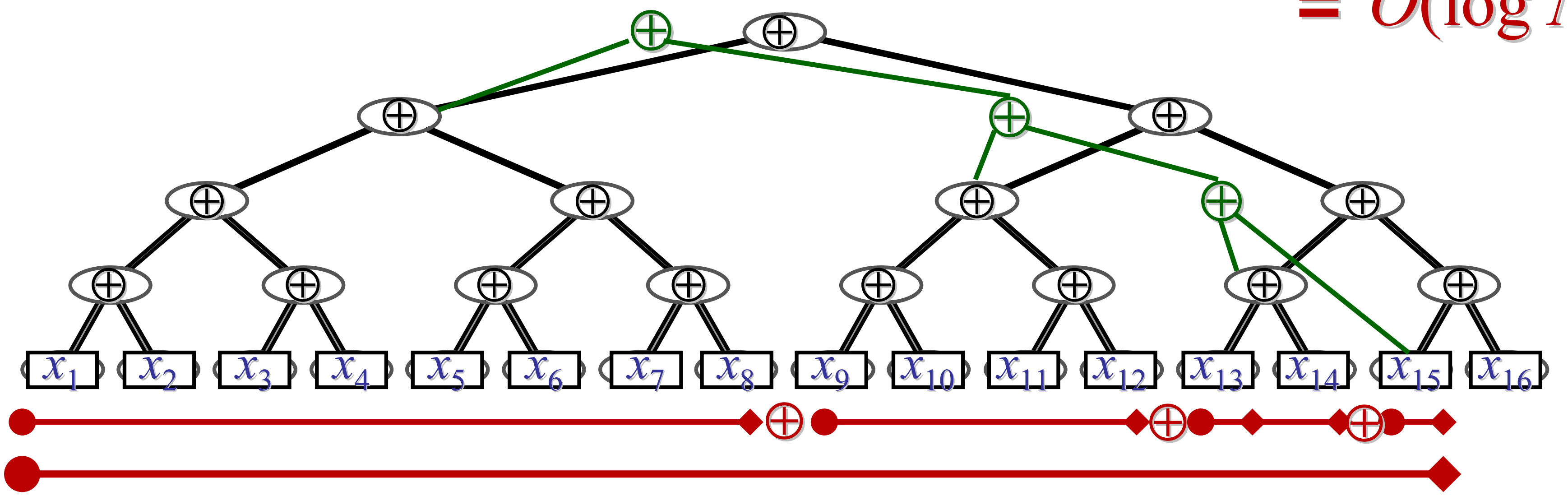
Overhead = Size/Time = $O(N)$

$(x_1,\ldots,x_N) \to \left(\bigoplus_{1\leq n\leq M} x_n : M=1\ldots N\right)$

Parallel *Prefix*

Size = $O(N)$ binary op.s
+ $O(N)$ = $O(N)$

Depth = $O(\log N)$
+ $O(\log N)$
= $O(\log N)$



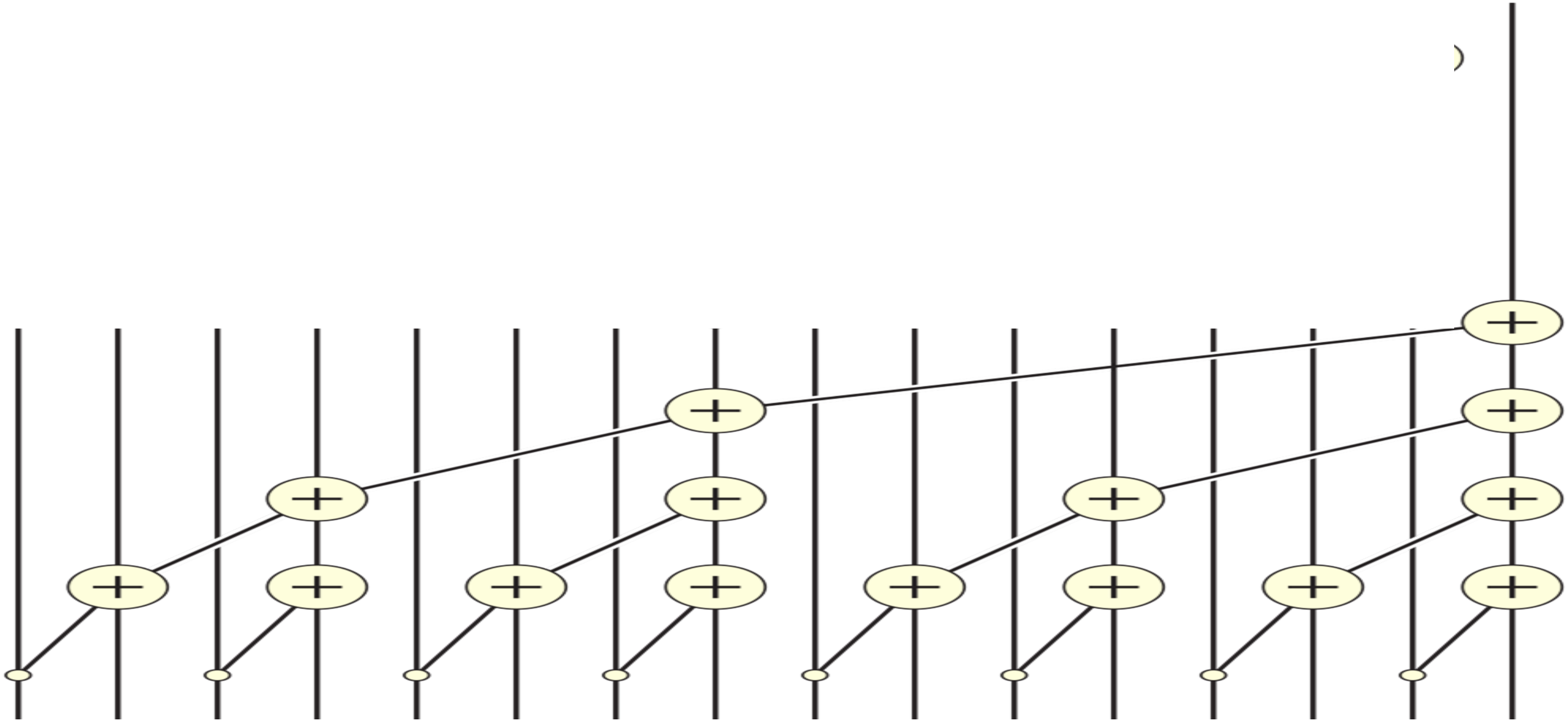
$(x_1,\ldots,x_N) \to \left(\bigoplus_{1\le n\le M} x_n : M=1\ldots N \right)$

(seq.opt.) **Time** = $O(N)$

Speedup = Time/Depth = $O(N/\log N)$

Overhead = Size/Time = $O(1)$

Parallel *Prefix*



(seq.opt.) **Time** = $O(N)$

Speedup = Time/Depth = $O(N/\log N)$

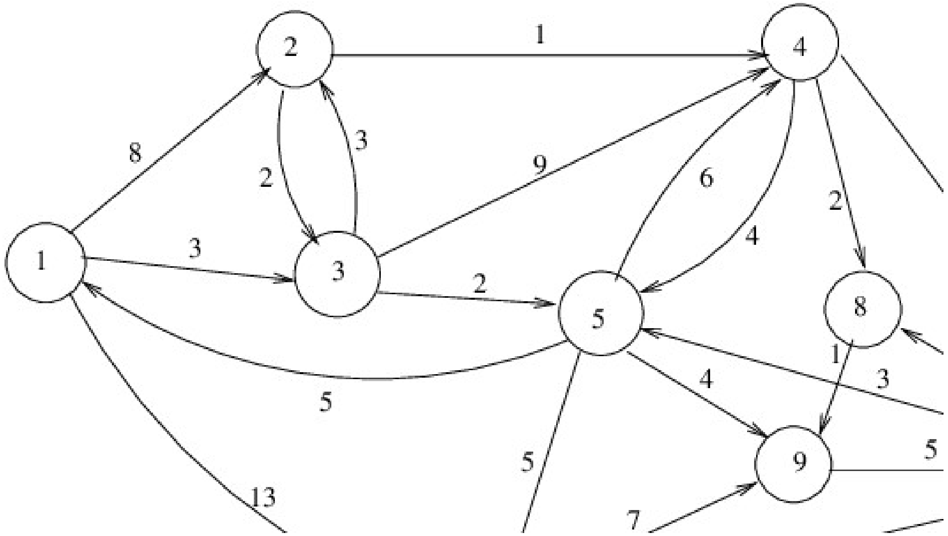
Overhead = Size/Time = $O(1)$

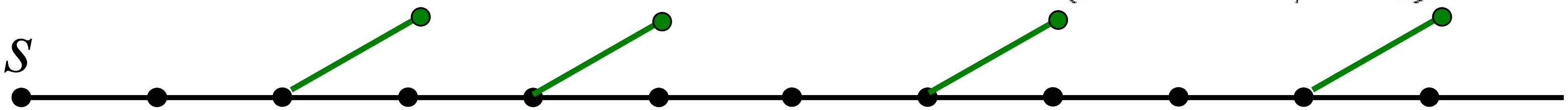
$(x_1,\ldots,x_N) \rightarrow$
$\Big(\bigoplus_{1\leq n\leq M} x_n :$
$M=1\ldots N\Big)$

Sequential Graph Reachability

Input: $K,s,t\in\mathbb{N}$ and directed graph as $N\times N$
`0/1` adjacency matrix A, vertices $V=\{1\ldots N\}$

Output: Is t reachable from s within distance K **?**





Dijkstra: Time = $O(N^2)$

Parallel Graph Reachability

Size = $O(N^3 \cdot \log K)$ **Depth** = $O(\log N \cdot \log K)$

Input: $K,s,t \in \mathbb{N}$ and directed graph as $N \times N$
`0/1` adjacency matrix A, vertices $V=\{1...N\}$

Output: Is t reachable from s within distance K **?**

Answer: $(A^K)_{s,t}$ Bool. matrix power

$N \times N$ *Boolean* matrix product

depth $O(\log N)$
size $O(N^3)$

$$(X \cdot Y)_{I,J} = \boxed{\bigvee_L} \; X_{I,L} \wedge Y_{L,J} \qquad \forall \; I,J = 1..N$$

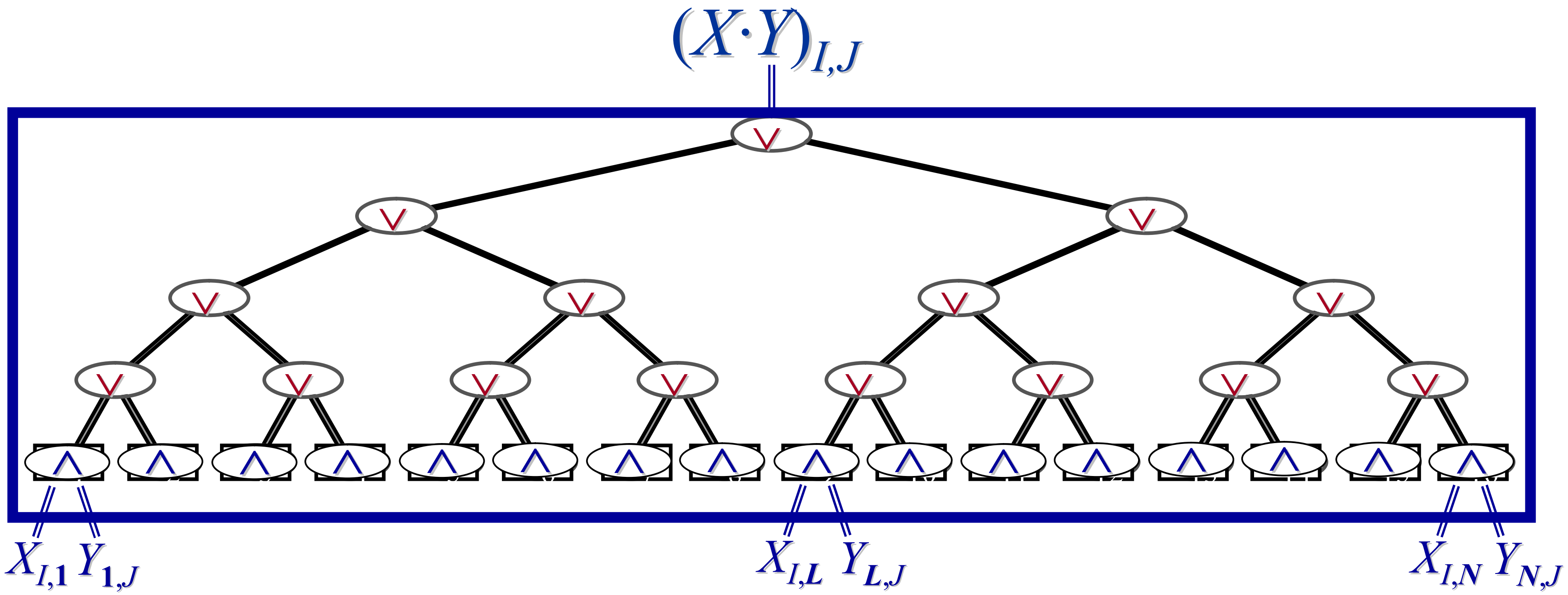
Matrix powering $A \to A^K$
by repeated squaring $A^2, A^4, A^8, \ldots$

$O(\log K)$ repetitions

Dijkstra: Time = $O(N^2)$

Associative $\bigvee$

Explicit Parallel Reachability



$$(X\cdot Y)_{I,J} = \bigvee_L X_{I,L} \wedge Y_{L,J} \qquad \forall\ I,J=1..N$$

Matrix powering $A \to A^K$
by repeated squaring $A^2, A^4, A^8, \ldots$

$O(\log K)$ repetitions

Explicit Parallel Reachability

$\boxed{(A\cdot A)_{1,1}}$ ...... $\boxed{(A\cdot A)_{I,J}}$ ...... $\boxed{(A\cdot A)_{N,N}}$

$$A\cdot A = A^2$$

$\boxed{(A^2\cdot A^2)_{1,1}}$ ...... $\boxed{(A^2\cdot A^2)_{I,J}}$ ...... $\boxed{(A^2\cdot A^2)_{N,N}}$

$$A^2\cdot A^2 = A^4$$

$$(X\cdot Y)_{I,J} = \bigvee_L X_{I,L} \wedge Y_{L,J} \qquad \forall\ I,J=1..N$$

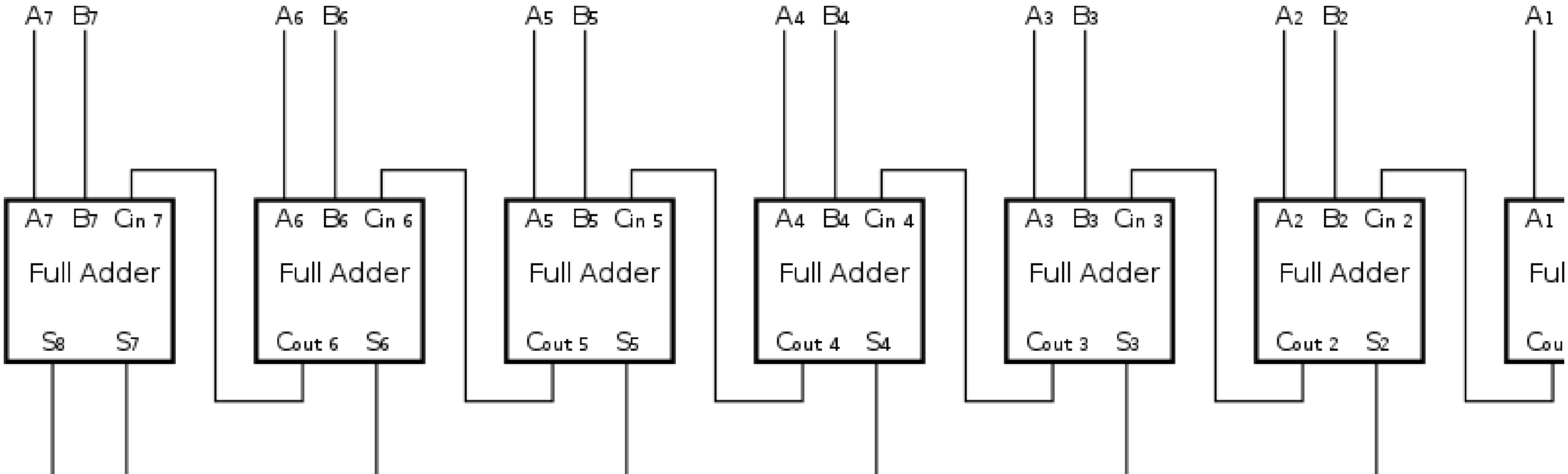
Matrix powering $A \to A^K$
by repeated squaring $A^2, A^4, A^8, \ldots$

$O(\log K)$ repetitions

Sequential *Ripple-Carry* Adder

Input: two n-bit integers in binary,

$A=(A_0,\ldots,A_{N-1})_2$ and $B=(B_0,\ldots,B_{N-1})_2$



(seq.opt.) **Time** = $O(N)$

Depth = $O(N)$

Speedup = Time/Depth = $O(1)$

Output: $S:=A+B$ in binary,

$S=(S_0,\ldots,S_{N-1},S_N)_2$

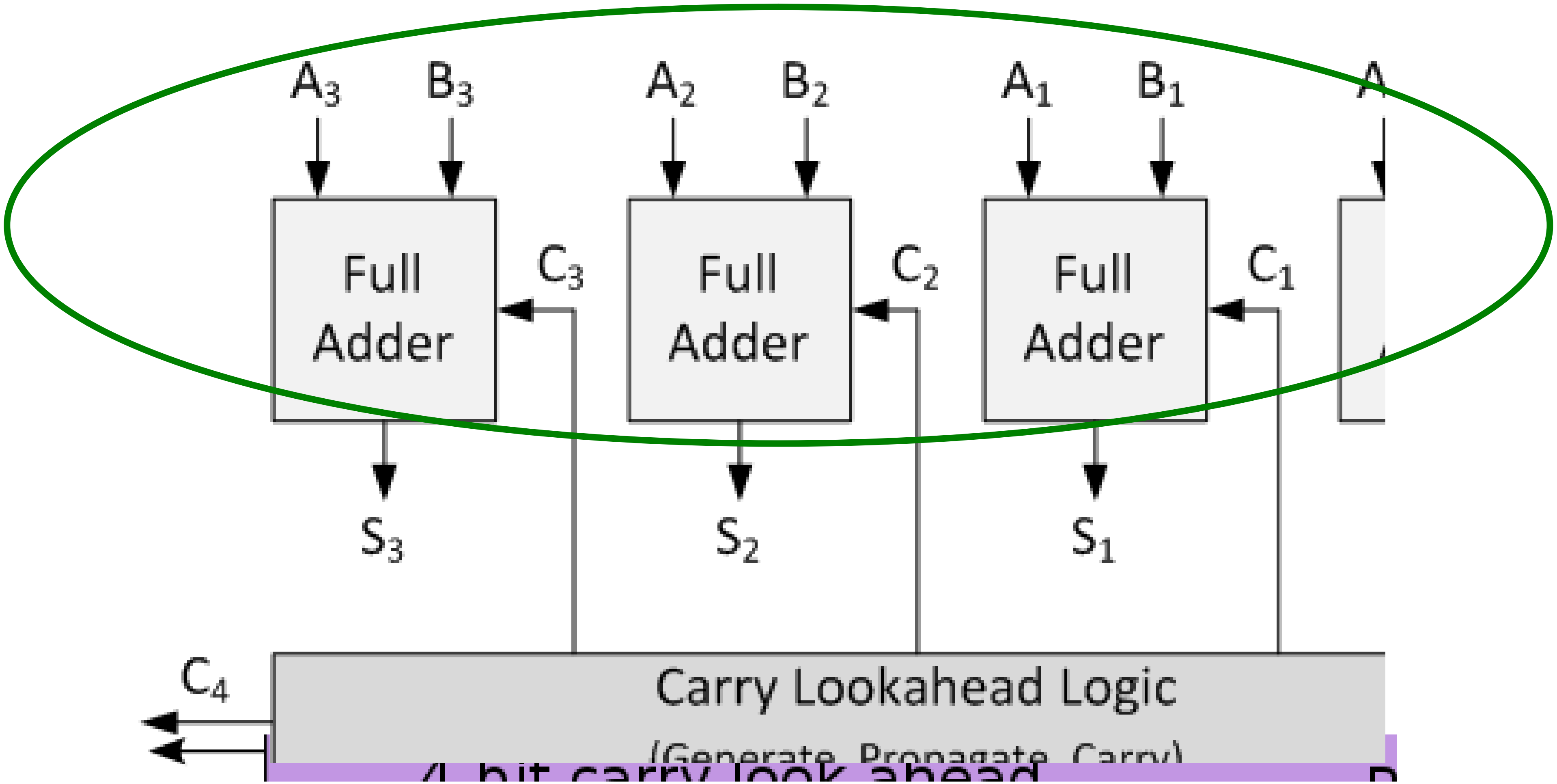
Operations:

binary $\wedge, \vee, \neg$

Carry-*Lookahead* Adder

Size = $O(N)$ binary bit op.s
+ $O(N)$ = $O(N)$

Depth = $O(\log N)$
+ $O(1)$ = $O(\log N)$



Carry Predictor

Size = $O(N^3)$

Depth = $O(\log N)$

$g_M := A_M \wedge B_M$ ***g****enerate* new carry to stage #M+1

$p_M := A_M \vee B_M$ ***p****ropagate* (possible) previous carry

$$C_{M+1} := g_M \vee (p_M \wedge C_M)$$

$C_N = g_{N\text{-}1} \vee g_{N\text{-}2} \wedge p_{N\text{-}1} \vee g_{N\text{-}3} \wedge p_{N\text{-}2} \wedge p_{N\text{-}1} \vee \quad \ldots$

$\ldots \vee g_0 \wedge p_1 \wedge p_2 \wedge \ldots \wedge p_{N\text{-}2} \wedge p_{N\text{-}1}$

Final Goal: Calculate $(C_1 \ldots C_N)$ in *parallel*!

Carry Predictor / Prefix Sum

Size = $O(N)$

Depth = $O(\log N)$

$g_M := A_M \wedge B_M$ ***generate*** new carry to stage $\#M{+}1$

$p_M := A_M \vee B_M$ ***propagate*** (possible) previous carry

$$C_{M+1} := g_M \vee (p_M \wedge C_M) \qquad P_{M+1} := p_M \wedge P_M \qquad P_0 = 0$$

Lemma: The following operation © on *pairs* of bits

$$(C,P) © (C',P') := (\, C \vee (P \wedge C'),\, P \wedge P' \,)$$

is (a) *idempotent* and (b) *associative*.

Power of Abstraction!

$(x_1,\ldots,x_N) \to (©_{1\leq n\leq M}\, x_n \;:\; M=1\ldots N\,)$, © associative

Parallel Sorting, *Revisited*

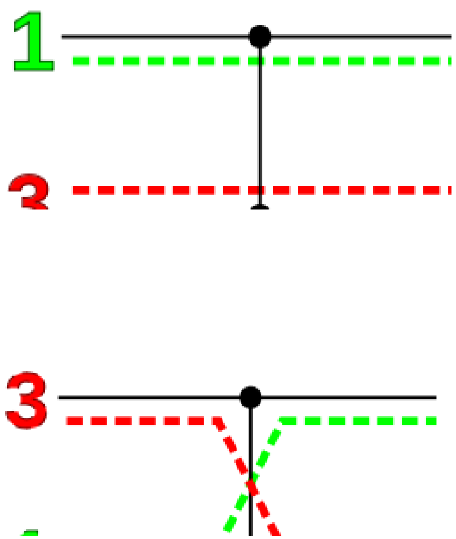
$(X,\leq)$ totally ordered set

Gate semantics $(n>m)$:

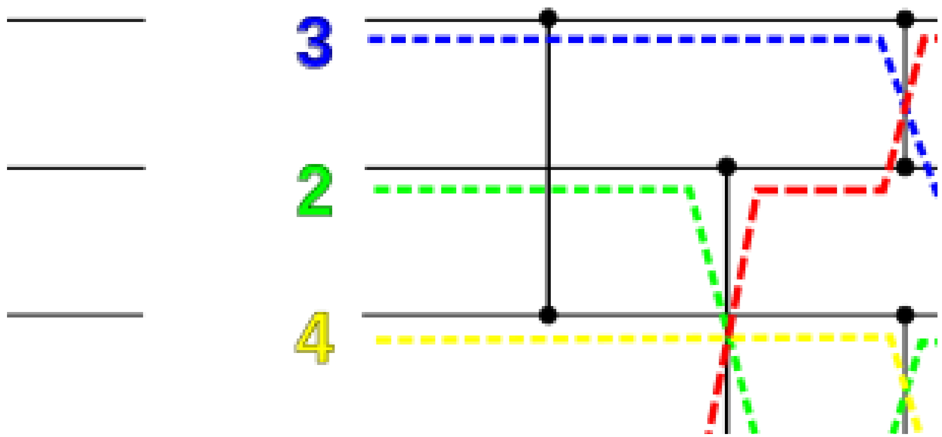
“`If` $x[n]<x[m]$ `then`

`swap` $x[n] \leftrightarrow x[m]$”

primitive/gate:



Example: *Sort4*



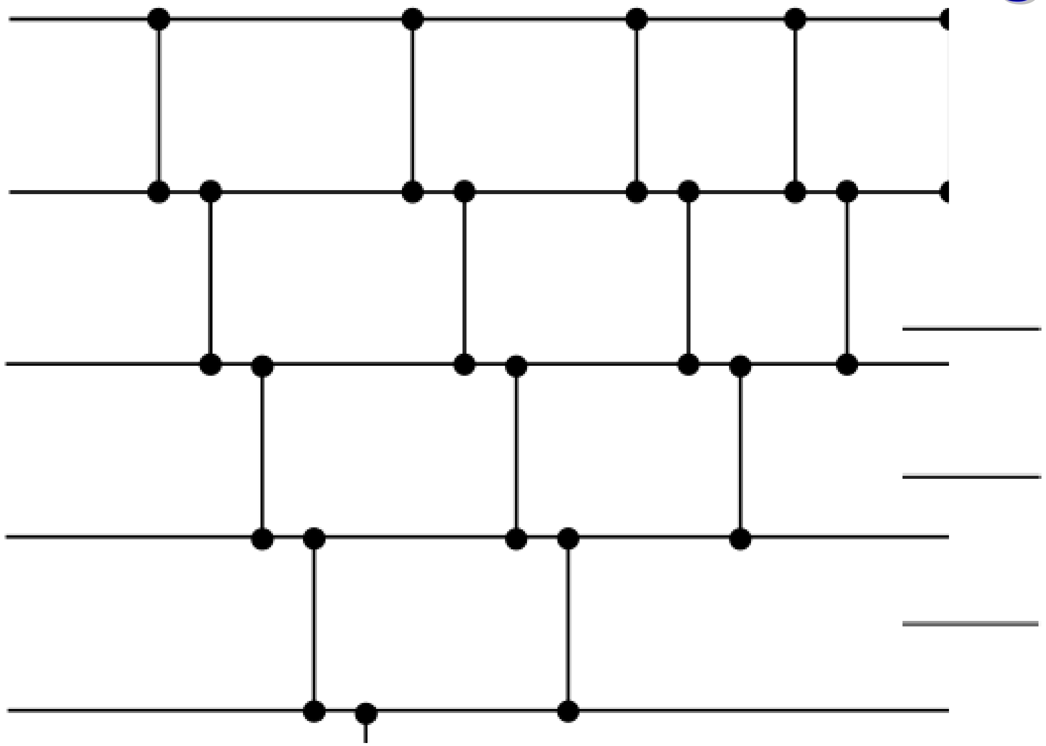
Bubble Sorting Network

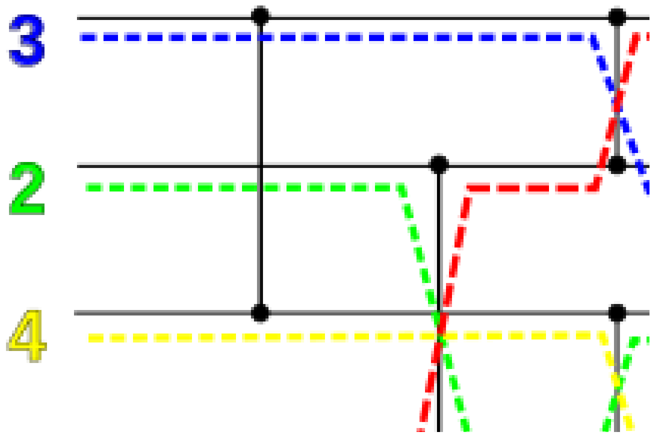
Size $\rightarrow$ opt.time $O(N\cdot\log N)$ **?** **Depth** $\rightarrow$ $O(\log N)$ **?**

(Brent's Principle)

***Bubble* Sorting Network:**

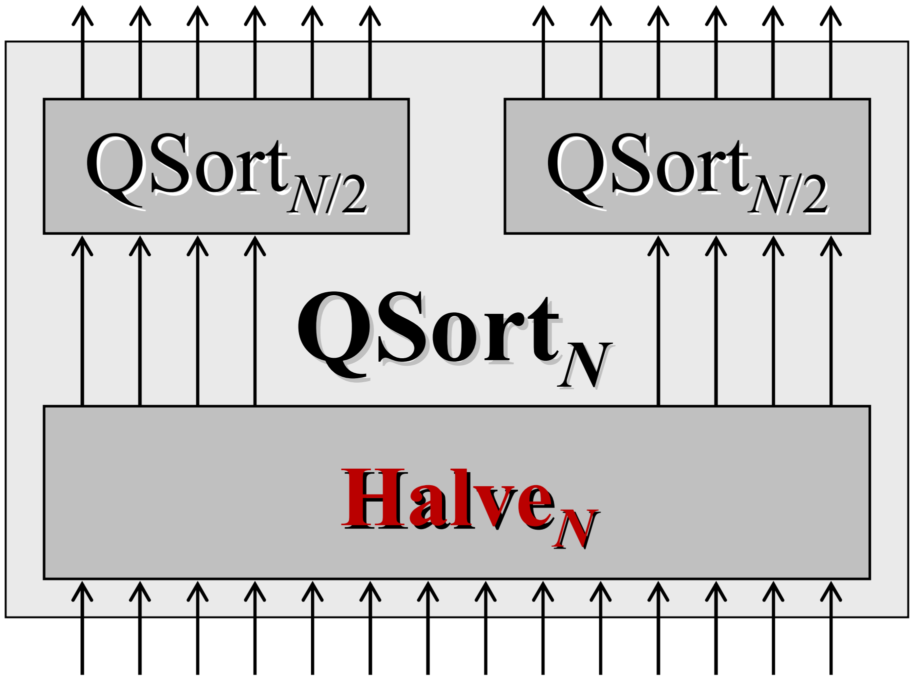
Size $O(N^2)$, Depth $O(N)$





Parallelizing *QuickSort* ?

Size → opt.time $O(N \cdot \log N)$ **?** **Depth** → $O(\log N)$ **?**



***Bubble* Sorting Network:**

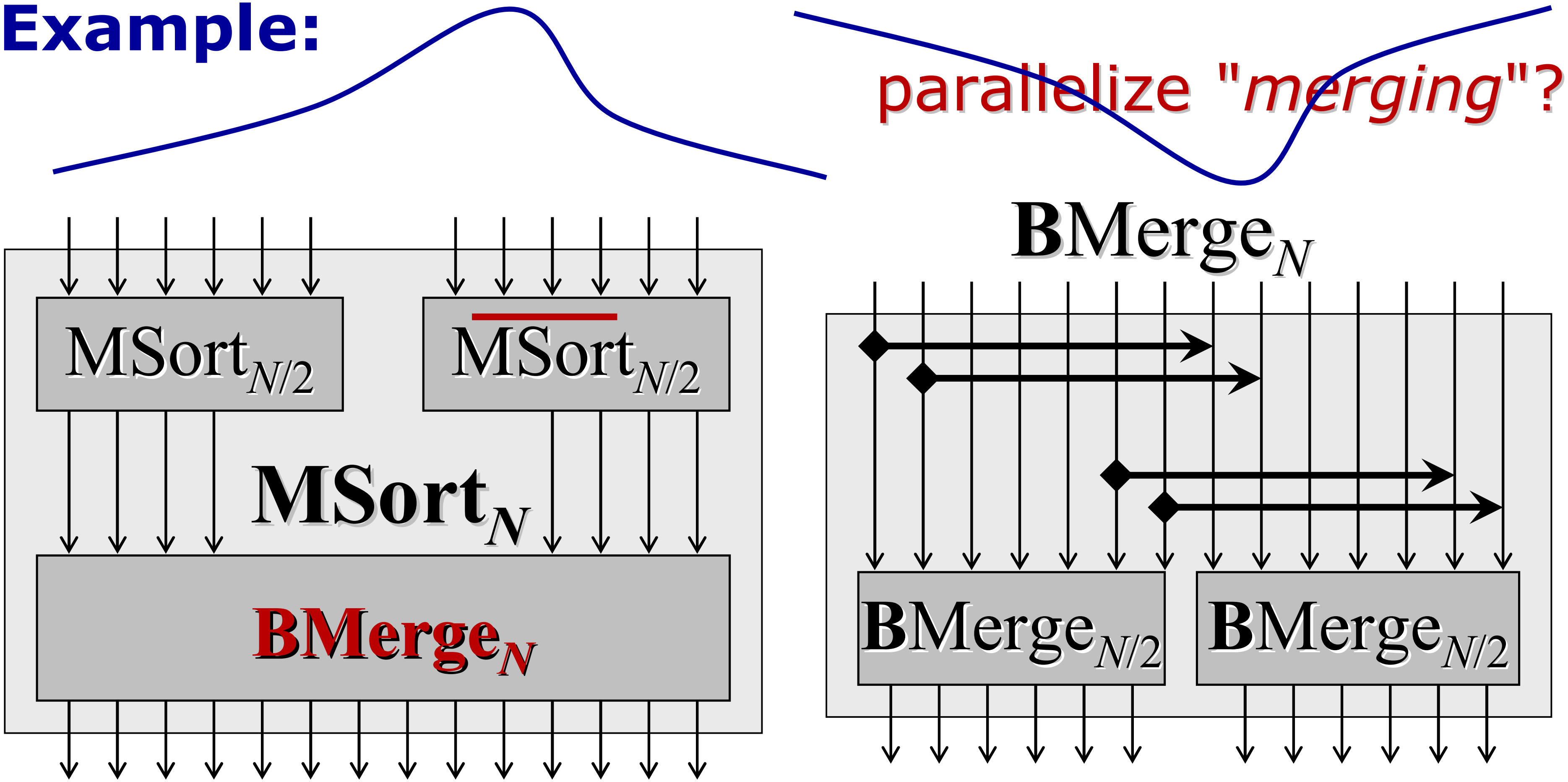
Size $O(N^2)$, Depth $O(N)$

Recall: *QuickSort* requires smart "halving" **!**

Parallelizing *Mergesort* ?

Example:

parallelize "*merging*"?



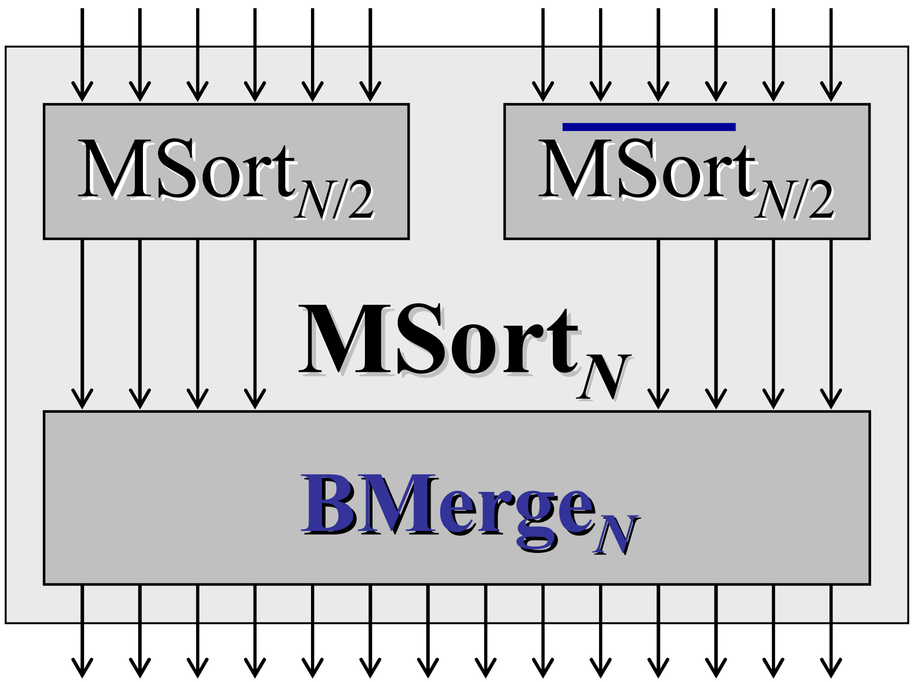
Def: Call $x[1...N]$ ***bitonic*** if, for some $M \le N$, or *cyclic shift*

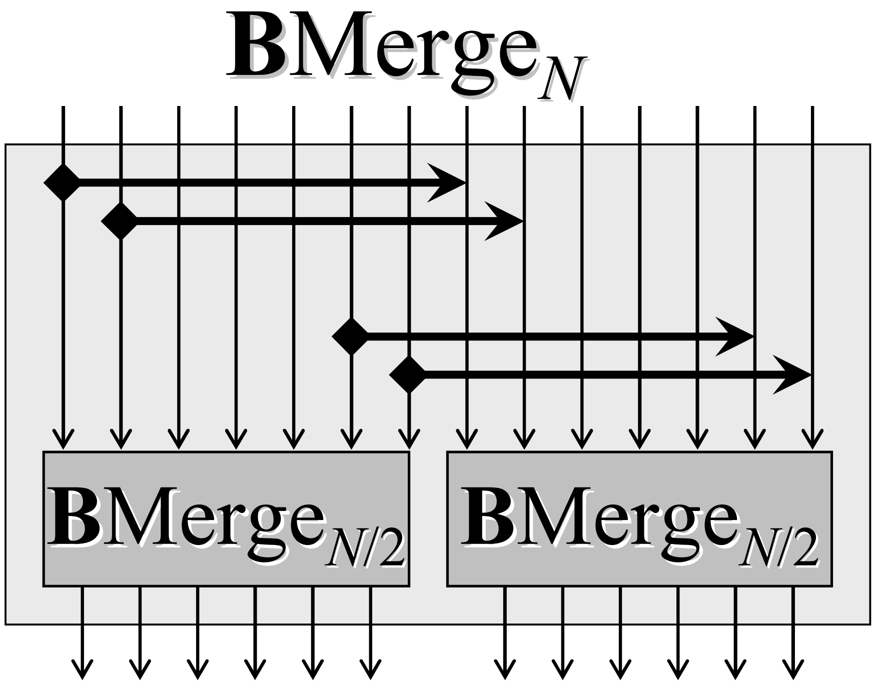
$x[1...M]$ non-*de*creasing & $x[M+1...N]$ non-*in*creasing

Batcher's Bitonic Sorter

Size = $O(N\cdot\log^2 N)$ gates → opt.time $O(N\cdot\log N)$ **?**

Depth = $O(\log^2 N)$ par.time → $O(\log N)$ **?**

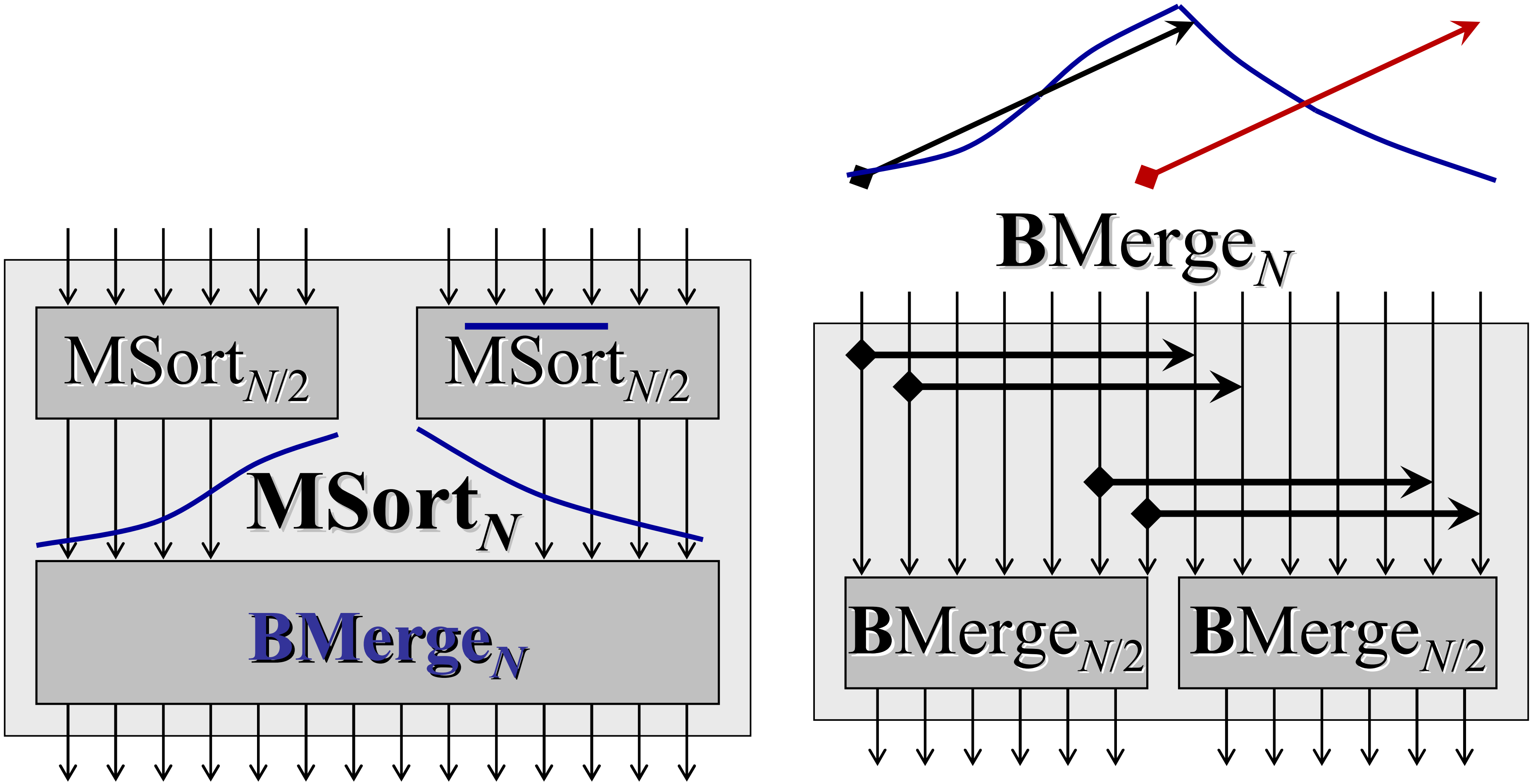




$$\text{DepthMSort}(N) = \mathbf{1}\cdot\text{DepthMSort}(N/2)+\text{DepthBMerge}(N)$$

$$\text{DepthBMerge}(N) = O(\mathbf{1})+\mathbf{1}\cdot\text{DepthBMerge}(N/2) = O(\log N)$$

Correctness of *Batcher*



Def: Call $x[1\ldots N]$ ***bitonic*** if, for some $M\leq N$, *or cyclic shift*

$x[1\ldots M]$ non-*de*creasing & $x[M+1\ldots N]$ non-*in*creasing

AKS Sorting Network

Batcher's Bitonic Sorting Network

Size = $O(N \cdot \log^2 N)$ gates → opt.time $O(N \cdot \log N)$ **?**

Depth = $O(\log^2 N)$ par.time → $O(\log N)$ **?**

Ajtai, Komlós, Szemerédi (1983):

Sorting network of

Size $O(N \cdot \log N)$

Depth $O(\log N)$

§9 Summary

- Classification
- Parallel Prefix
- Graph Reachability
- Carry-Lookahead
- Sorting Networks