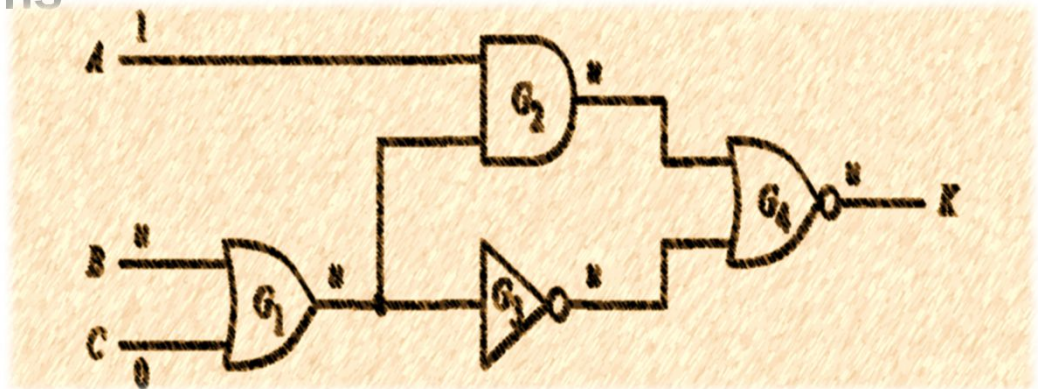


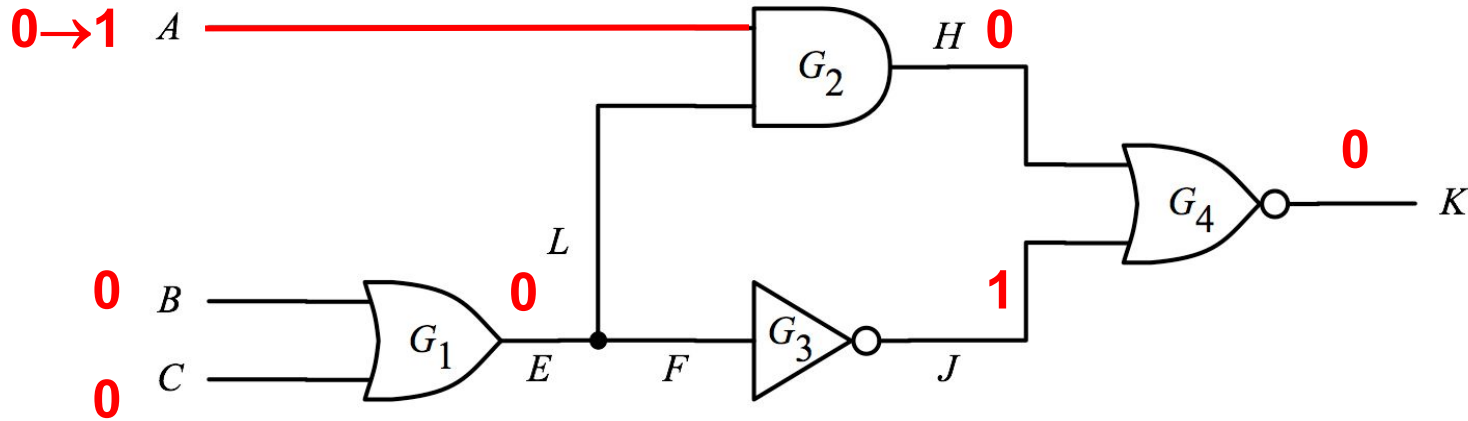
Logic Simulation

- Introduction
- Simulation Models
- **Logic Simulation Techniques**
 - ◆ Compiled-code simulation
 - ◆ **Event-driven simulation (1965)**
 - * Zero delay
 - * **Nominal delay**
 - * **False events**
 - ◆ Parallel Simulation
- Issues of Logic Simulations
- Conclusions



Compiled-code Simulation Problems

- 1. Gate delay model not considered
- 2. **Oblivious**: forgets results in previous cycle



```
while{true} do
    read(A,B,C);
    E  $\square$  OR(B,C);
    H  $\square$  AND(A,E);
    J  $\square$  NOT(E);
    K  $\square$  NOR(H,J);
end
```

1st

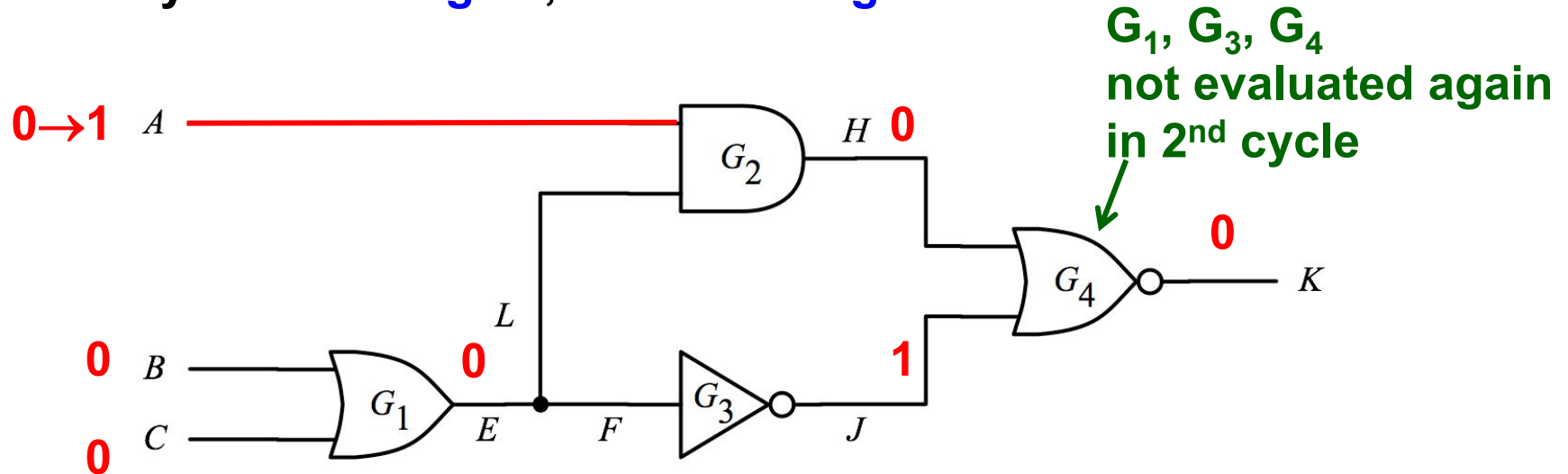
2nd

Forget 1st cycle results.

Rerun all codes for 2nd cycle.

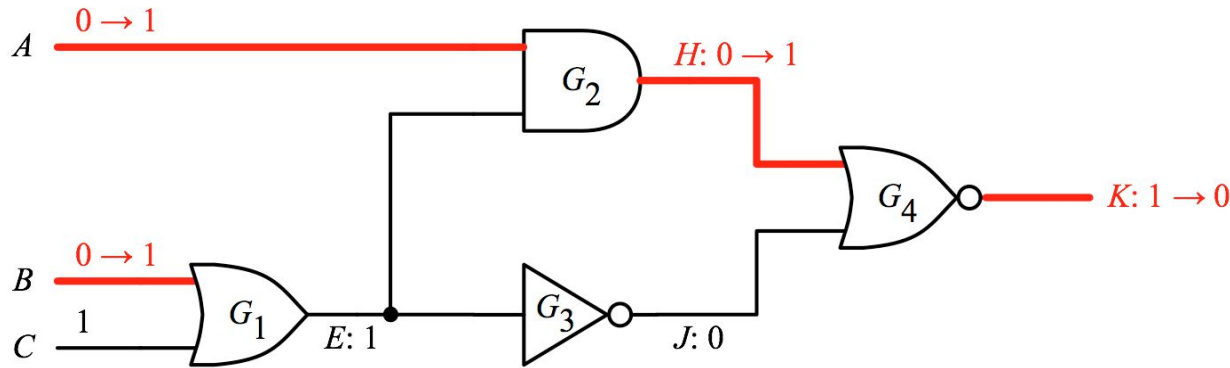
Event-Driven Simulation [Ulrich 1965]

- IDEA: evaluates a gate only if there is event(s) at its gate inputs
 - ♦ **Event** is a signal value change (at time t)
 - ♦ Gates with inputs changed are **activated**
- Example:
 - ♦ Event: **A 0→1**
 - ♦ Activated gate: **G₂**
 - ♦ Only evaluate **1 gate**, instead of **4 gates**



Event-driven Faster than CC

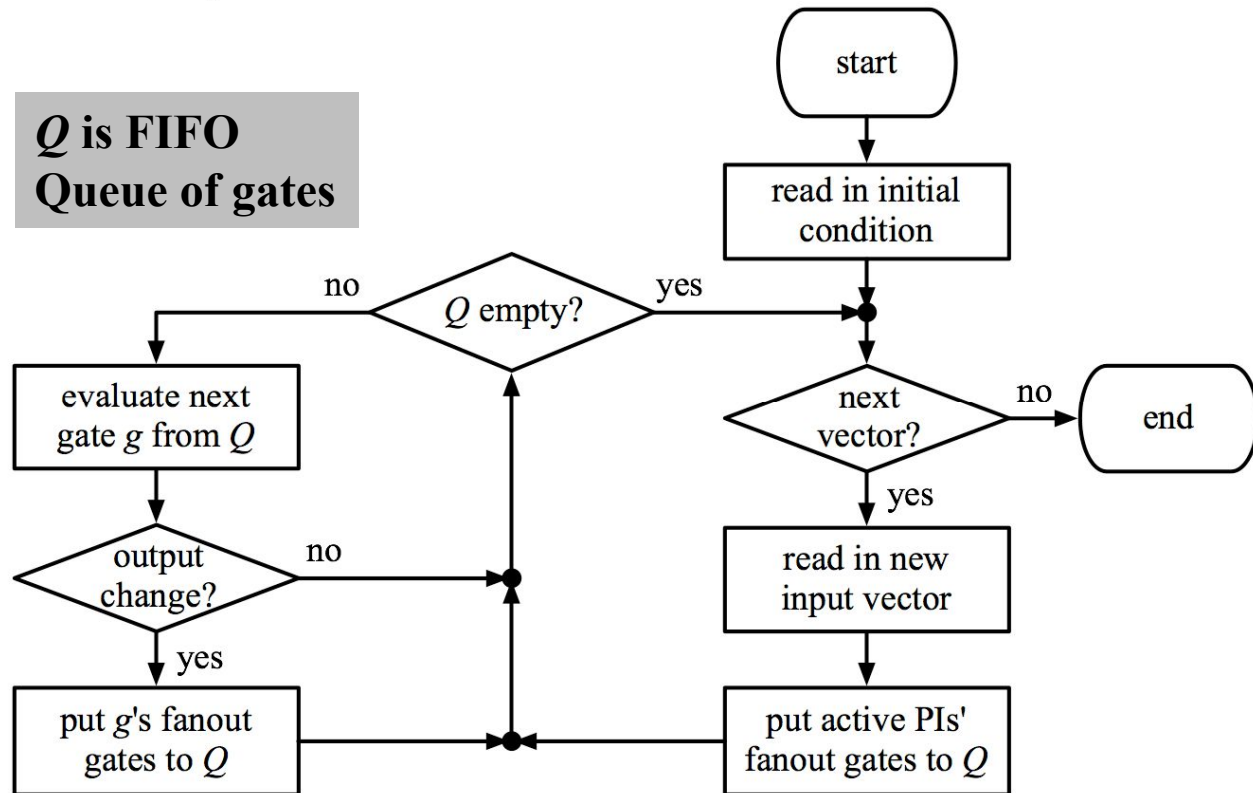
Zero-delay Event-driven Sim.



event (g, v_g^+) means
gate g changes to v_g^+

Q is FIFO
Queue of gates

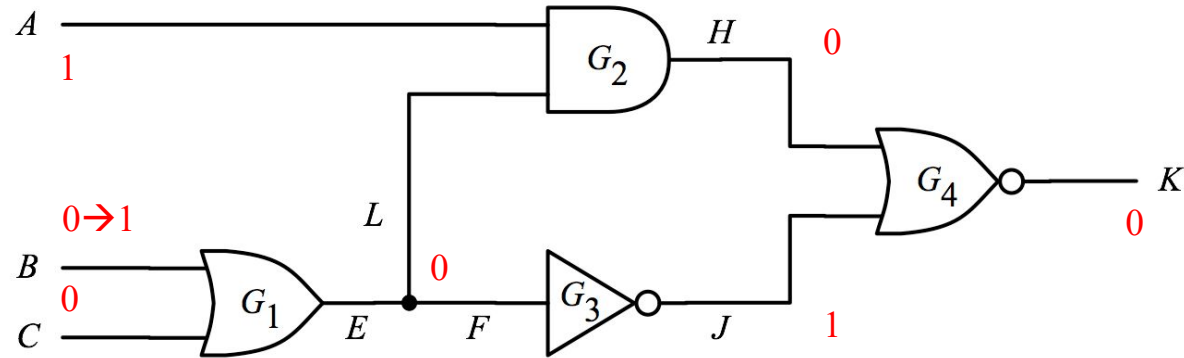
Executed events	Q
$(B,1)(A,1)$	G_1, G_2
$(G_1, 1) -$	G_2
$(G_2, 1)$	G_4
$(G_4, 0)$	-



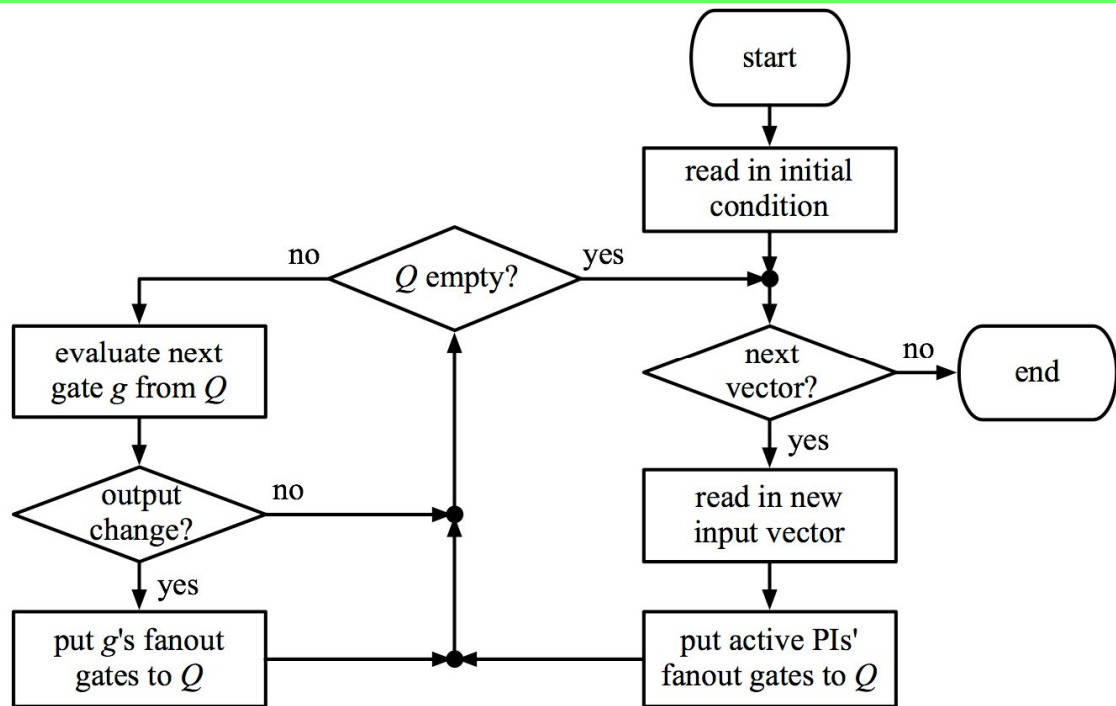
(WWW Fig 3.16)

Quiz

Please simulate this circuit

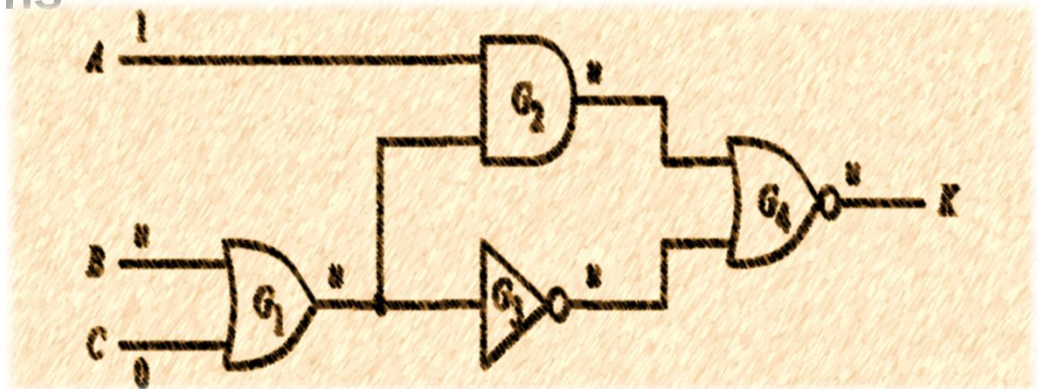


Executed Events	Q
$(B,1)$	G_1



Logic Simulation

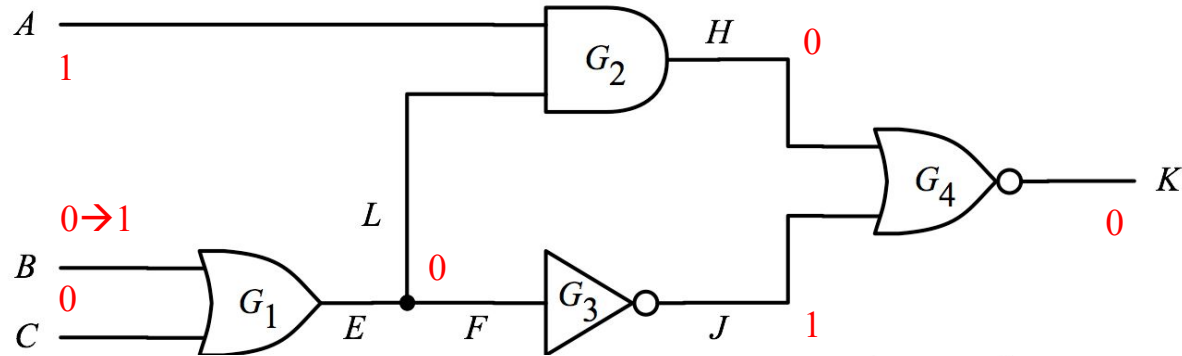
- Introduction
- Simulation Models
- Logic Simulation Techniques
 - ◆ Compiled-code simulation
 - ◆ Event-driven simulation (1965)
 - * Zero delay
 - * Nominal delay
 - * False events
 - ◆ Parallel Simulation
- Issues of Logic Simulations
- Conclusions



Nominal-delay Event-drive Sim.

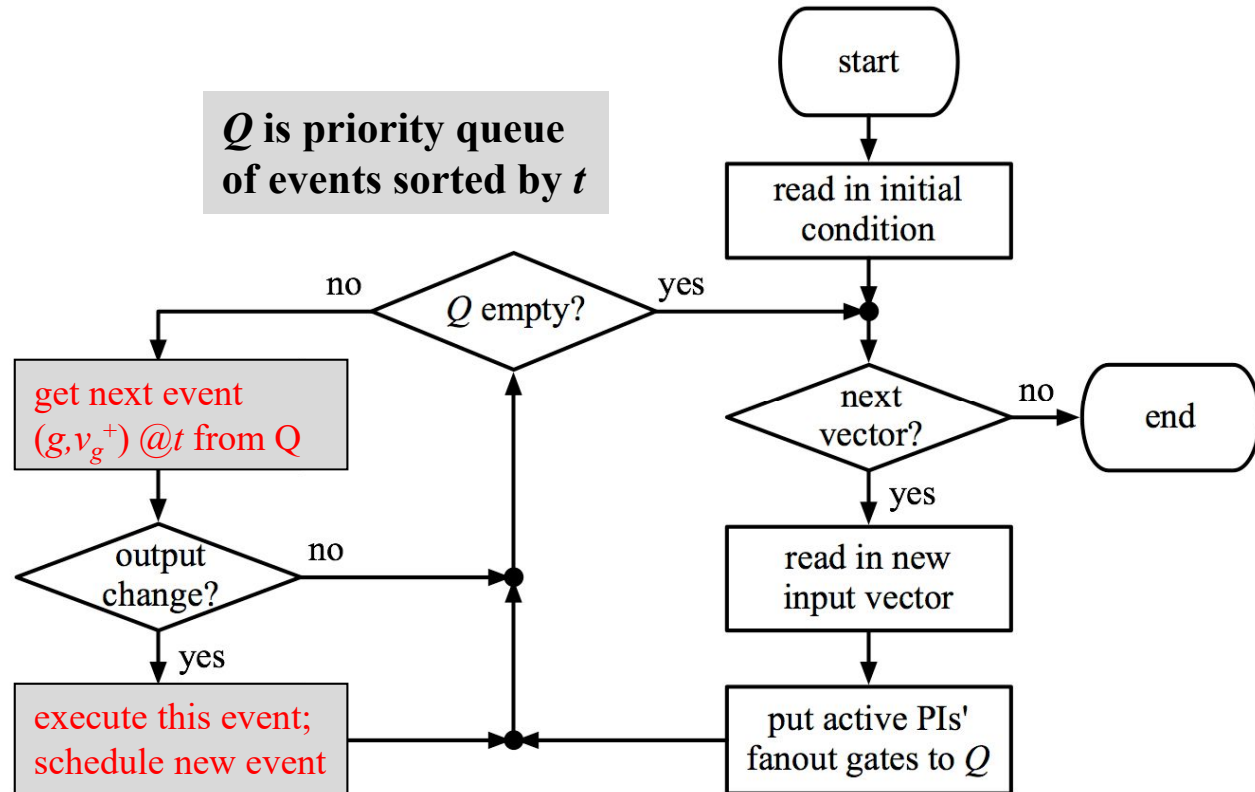
every gate delay = 1ns

event $(g, v_g^+) @ t$
gate g changes to v_g^+
at time stamp t



t	Executed events	sch'd events Q
0	$(B,1)$	$(G_1,1)@1$
1	$(G_1,1)$	$(G_3,0)@2$ $(G_2,1)@2$
2	$(G_3,0)$	$(G_2,1)@2$ $(G_4,1)@3$
2	$(G_2,1)$	$(G_4,1)@3$ $(G_4,0)@3$
3	$(G_4,1)$	$(G_4,0)@3$
3	$(G_4,0)$	

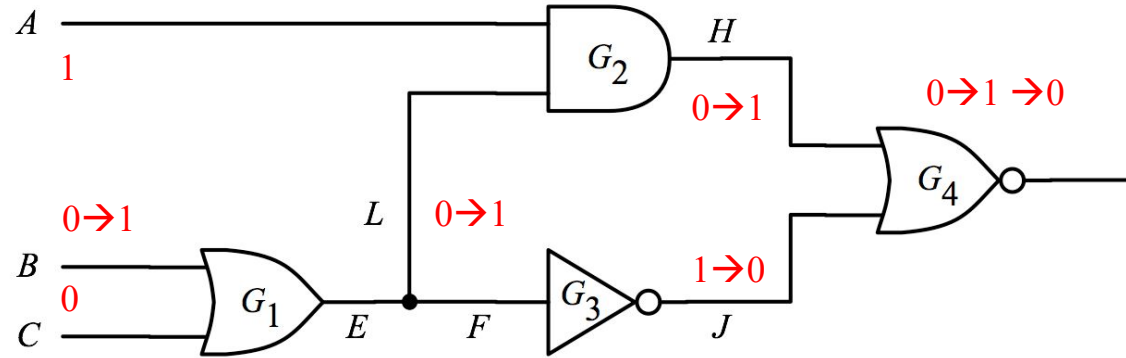
Q is priority queue
of events sorted by t



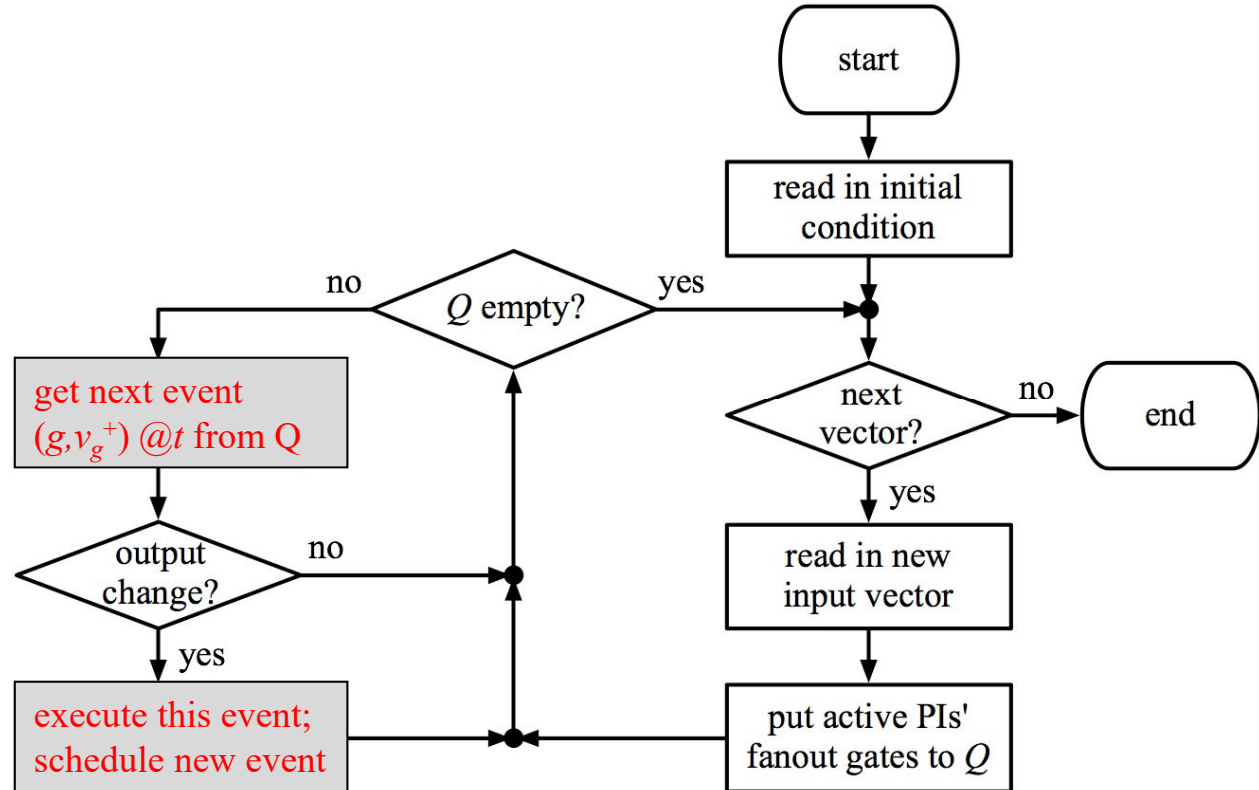
FFT

Simultaneous signal change
of G_4 @3 wastes CPU time.

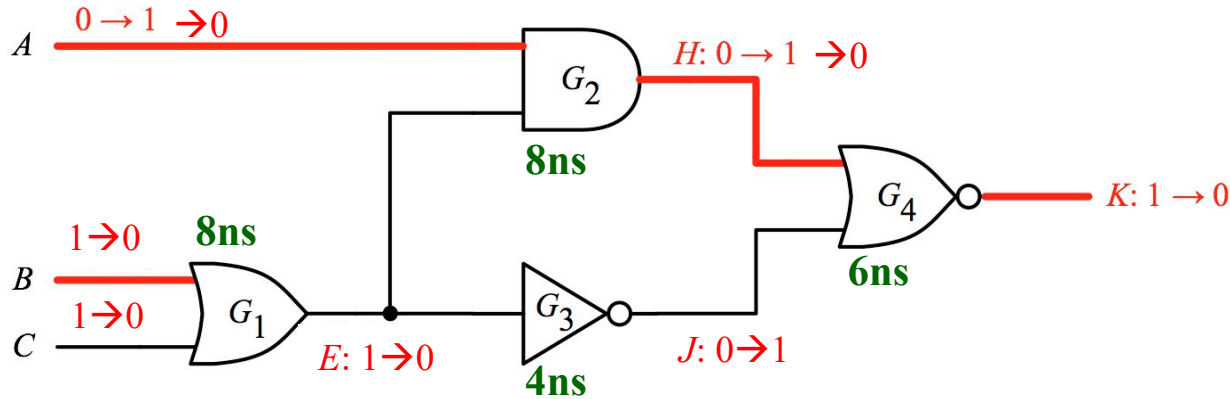
Q: Can we do better?



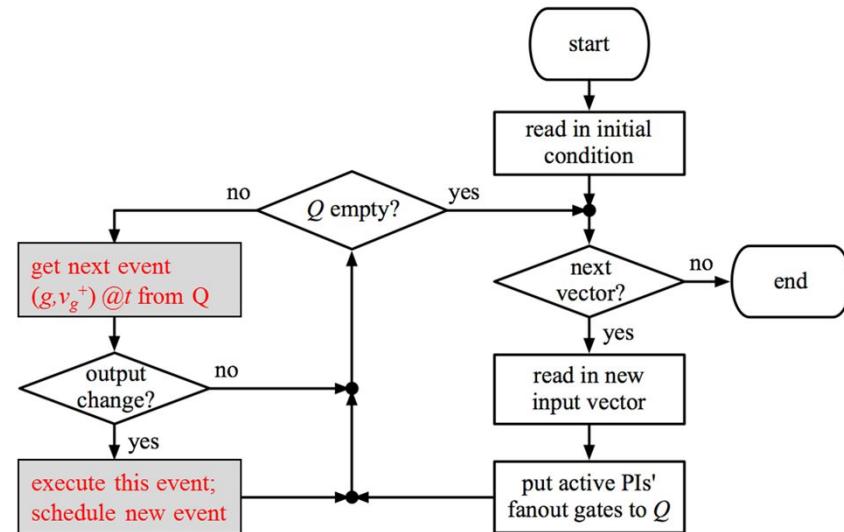
t	Executed events	sch'd events Q
0	(B,1)	($G_1,1$)@1
1	($G_1,1$)	($G_3,0$)@2 ($G_2,1$)@2
2	($G_3,0$)	($G_2,1$)@2 ($G_4,1$)@3
2	($G_2,1$)	($G_4,1$)@3 ($G_4,0$)@3
3	($G_4,1$)	($G_4,0$)@3
3	($G_4,0$)	



Quiz



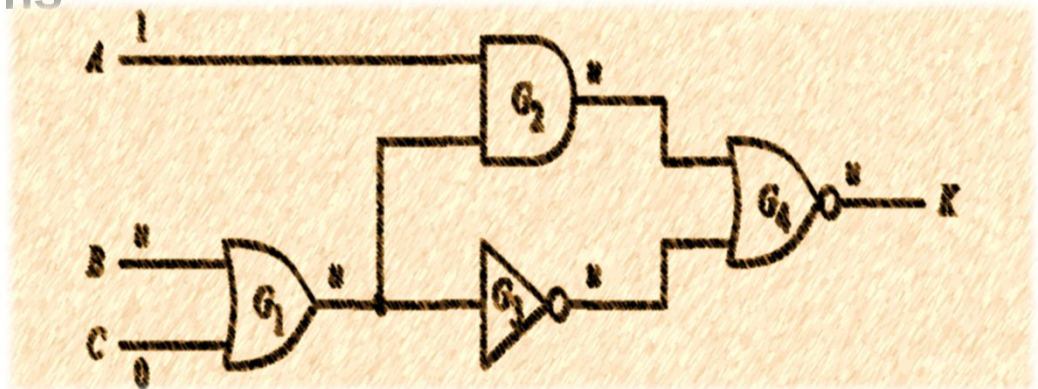
t	executed	scheduled events in Q
0	(A,1)	(G ₂ ,1)@8
2	(C,0)	(G ₂ ,1)@8; (G ₁ ,1)@10
4	(B,0)	(G ₂ ,1)@8; (G ₁ ,1)@10; (G ₁ ,0)@12
8	(A,0),(G ₂ ,1)	(G ₁ ,1)@10; (G ₁ ,0)@12 (G ₄ ,0)@14; (G ₂ ,0)@16
10	(G ₁ ,1)	(G ₁ ,0)@12;(G ₄ ,0)@14;(G ₂ ,0)@16
12	(G ₁ ,0)	(G ₄ ,0)@14; (G ₂ ,0)@16 (G ₃ ,1)@16; (G ₂ ,0)@20
14		
16		
20		
22	(G ₄ ,1)(G ₄ ,0)(G ₄ ,0)	-



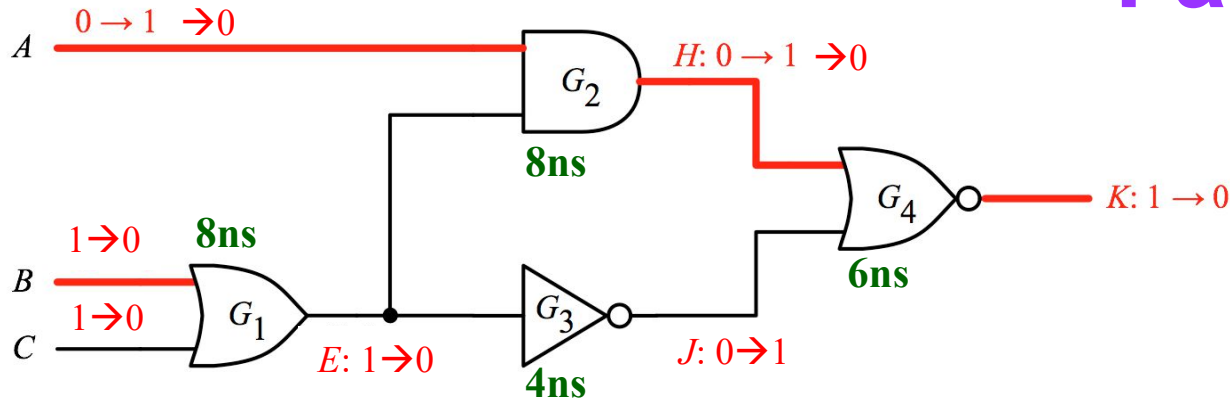
*Two events are shown in same row for simplicity

Logic Simulation

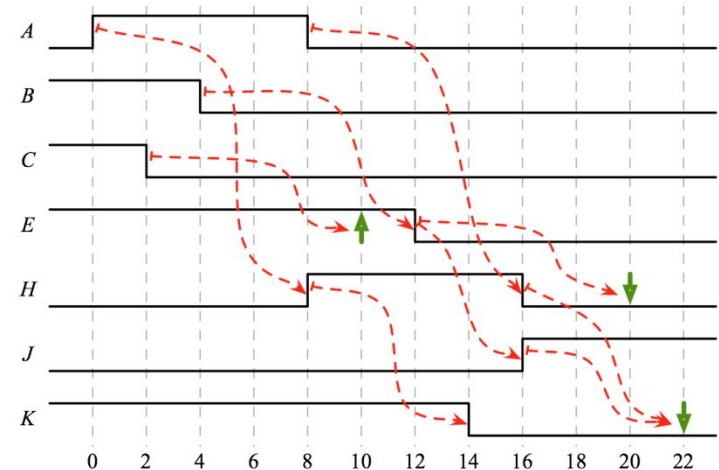
- Introduction
- Simulation Models
- **Logic Simulation Techniques**
 - ◆ Compiled-code simulation
 - ◆ **Event-driven simulation (1965)**
 - * Zero delay
 - * Nominal delay
 - * **False events**
 - ◆ Parallel Simulation
- Issues of Logic Simulations
- Conclusions



False Events



t	executed	scheduled events in Q
0	(A,1)	($G_2,1$)@8
2	(C,0)	($G_2,1$)@8; ($G_1,1$)@10
4	(B,0)	($G_2,1$)@8; ($G_1,1$)@10; ($G_1,0$)@12
8	(A,0),($G_2,1$)	($G_1,1$)@10; ($G_1,0$)@12 ($G_4,0$)@14; ($G_2,0$)@16
10	($G_1,1$) false	($G_1,0$)@12; ($G_4,0$)@14; ($G_2,0$)@16
12	($G_1,0$)	($G_4,0$)@14; ($G_2,0$)@16 ($G_3,1$)@16; ($G_2,0$)@20
14	($G_4,0$)	($G_2,0$)@16; ($G_3,1$)@16; ($G_2,0$)@20
16	($G_2,0$)($G_3,1$)	($G_2,0$)@20; ($G_4,1$)@22; ($G_4,0$)@22
20	($G_2,0$) false	($G_4,1$)@22; ($G_4,0$)@22; ($G_4,0$)@22
22	($G_4,1$)($G_4,0$)($G_4,0$)	-



**False Events cause
no change in signal**

Event-driven Alg. w/o False Events

- For each gate, record *last scheduled value* to avoid false events
- (BA Fig. 3.32)

event-driven-sim (t) /* t is current time stamp*/

1. **for every** event $(g, v_g^+) @ t$
2. $v_g = v_g^+$
3. **for every** j on the fauout list of g
4. update input values of j
5. $v_j^+ = \text{evaluate}(j)$
6. **if** $v_j^+ \neq \text{lsv}(j)$ /* lsv = last scheduled value */
7. schedule $(j, v_j^+) @ t+d_j$ /* d_j = gate delay of j */
8. $\text{lsv}(j) = v_j^+$

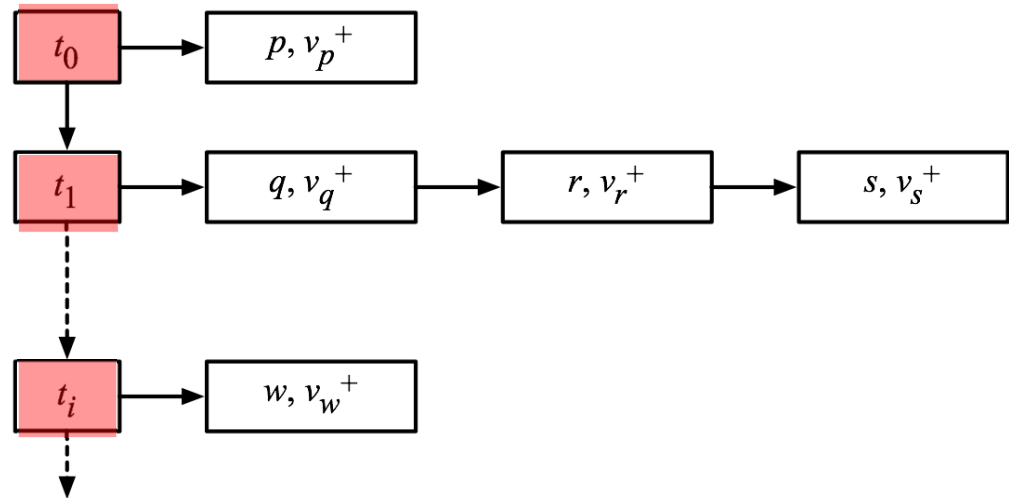
Trade off Memory for Speed

How to Implement Event List ?

1. Linked list

😊 t is flexible

😞 Slower to search

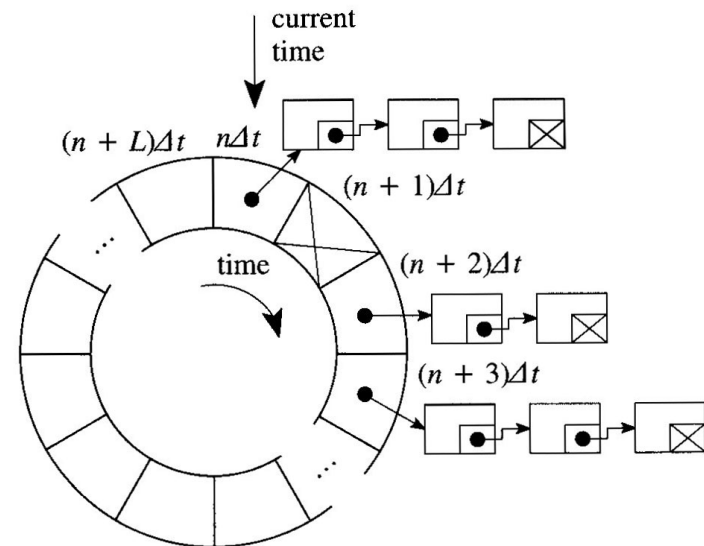


2. Cyclic Array (aka. **Timing Wheel**) [Ulrich 1969]

😊 Faster to search

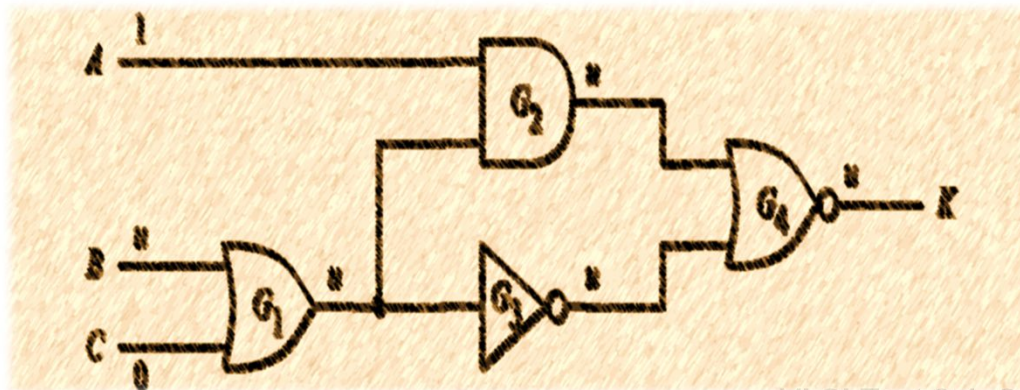
😞 Δt is fixed

😞 Limited L slots in a cycle



Summary

- **Event-driven simulation**
 - ♦ Consider **gate delay**
 - ♦ **Saves CPU time.** Only evaluate gates with events at input.
- **Two algorithms**
 - ♦ **Zero delay**
 - ♦ **Nominal delay**
- **False events** are redundant events that can be removed
- **Event list implementation**
 - ♦ **Linked list:** slow but flexible
 - ♦ **Cyclic Array (Timing wheel):** fast but fixed time slots



Comparison

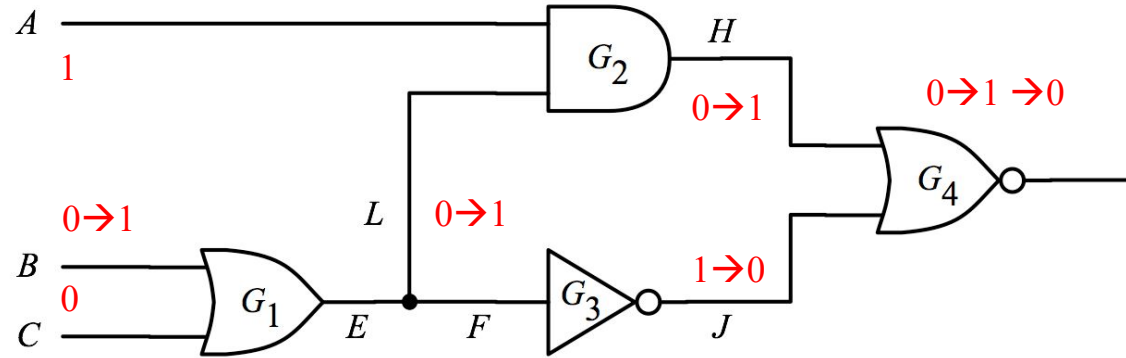
	Pros 😊	Cons 😞
Compiled-code	Simple implementation	No gate delay Oblivious Suitable for high activity
Event-driven	Consider gate delay Only simulate events Suitable for low activity	Complex algorithm

ED is Generally Better than CC

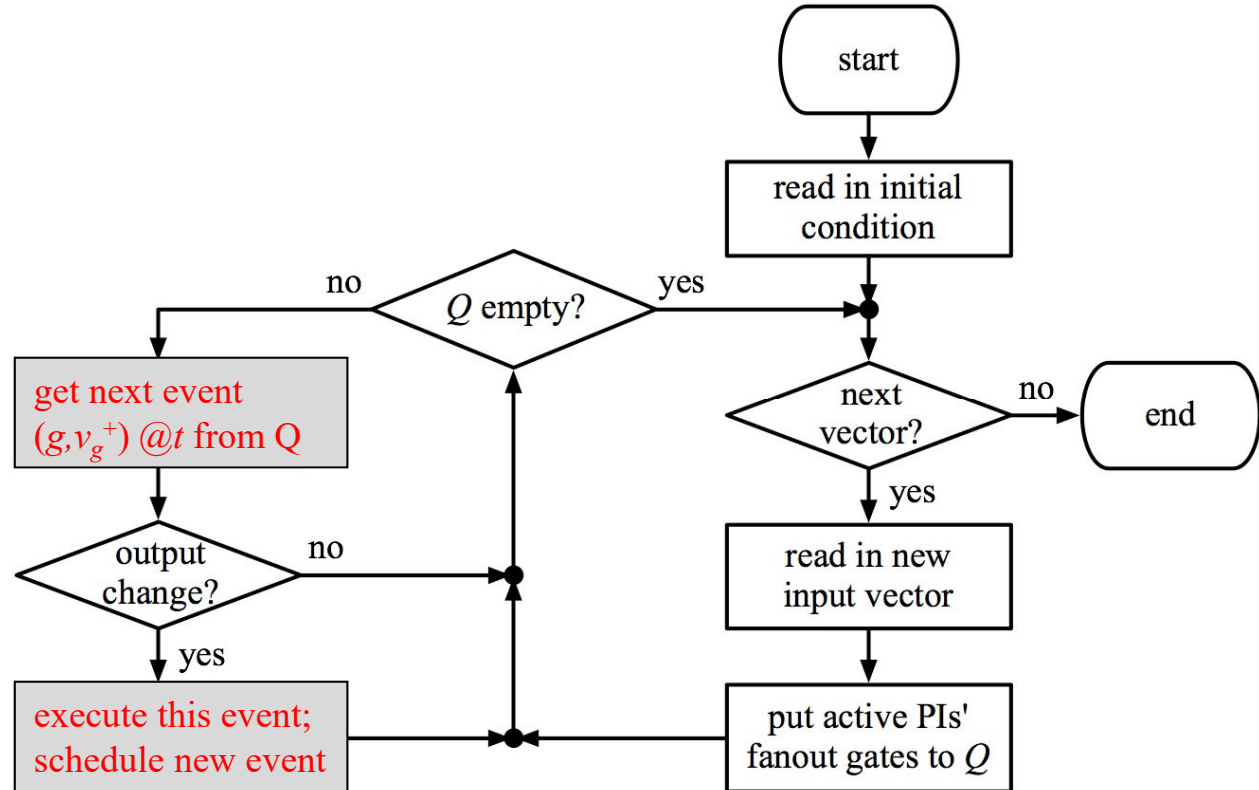
FFT

Simultaneous signal change
of G_4 @3 wastes CPU time.

Q: Can we do better?



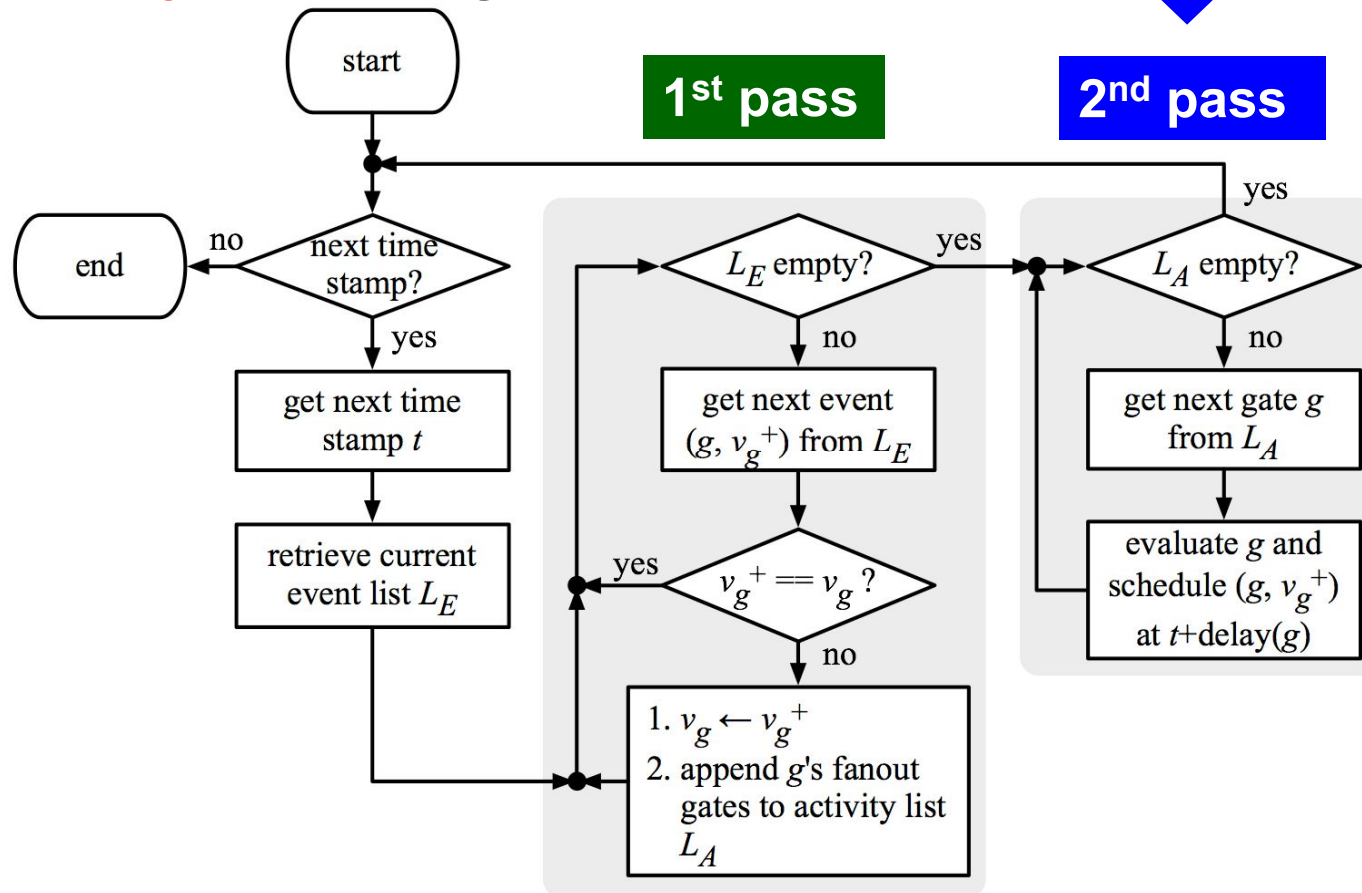
t	Executed events	sch'd events Q
0	$(B,1)$	$(G_1,1)@1$
1	$(G_1,1)$	$(G_3,0)@2$ $(G_2,1)@2$
2	$(G_3,0)$	$(G_2,1)@2$ $(G_4,1)@3$
2	$(G_2,1)$	$(G_4,1)@3$ $(G_4,0)@3$
3	$(G_4,1)$	$(G_4,0)@3$
3	$(G_4,0)$	



IDEA#1 Two-pass Algorithm [Ulrich 1965]

- 1st pass executes events
 - ♦ L_E **event list**, priority queue of events
- 2nd pass evaluates gates
 - ♦ L_A **activity list**, list of gates

All input events to a gate evaluated together.
Avoid simultaneous signal change



IDEA #2 One-pass Algorithm [Ulrich 1969]

- Add **last scheduled time** (lst)
- Cancel previously scheduled events when needed

event-driven-sim-new (t)

1. **for every** event $(g, v_g^+) @ t$
2. $v_g = v_g^+$
3. **for every** j on the fauout list of g
4. update input values of j
5. $v_j^+ = \text{evaluate}(j)$
6. **if** $v_j^+ \neq \text{lsv}(j)$
7. $t^+ = t + d_j$
8. **if** $t^+ == \text{lst}(j)$ /* simultaneous signal change*/
9. cancel event $(j, \text{lsv}(j)) @ t^+$
10. schedule $(j, v_j^+) @ t^+$
11. $\text{lsv}(j) = v_j^+$
12. $\text{lst}(j) = t^+$ /* lst = last scheduled time */

FFT

- Cyclic Array (aka. Timing Wheel)
 - ☹ Limited L slots in a cycle
- Q: what if **remote events** that is outside of $(n+L)\Delta t$?

