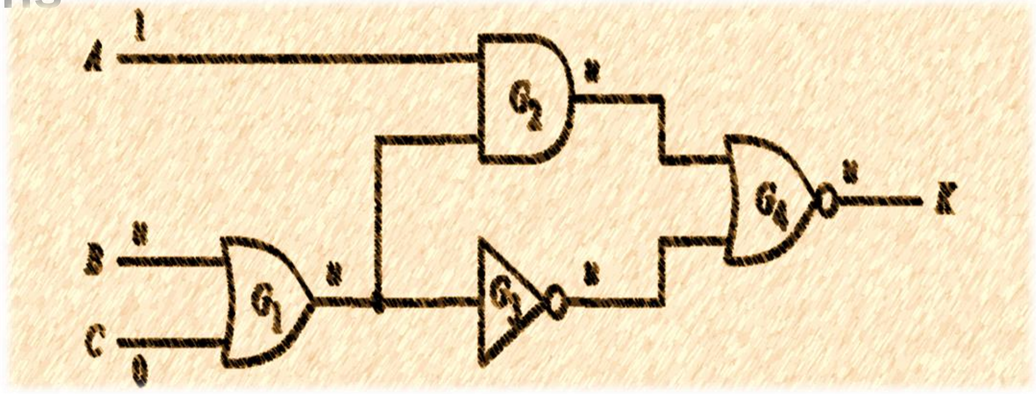
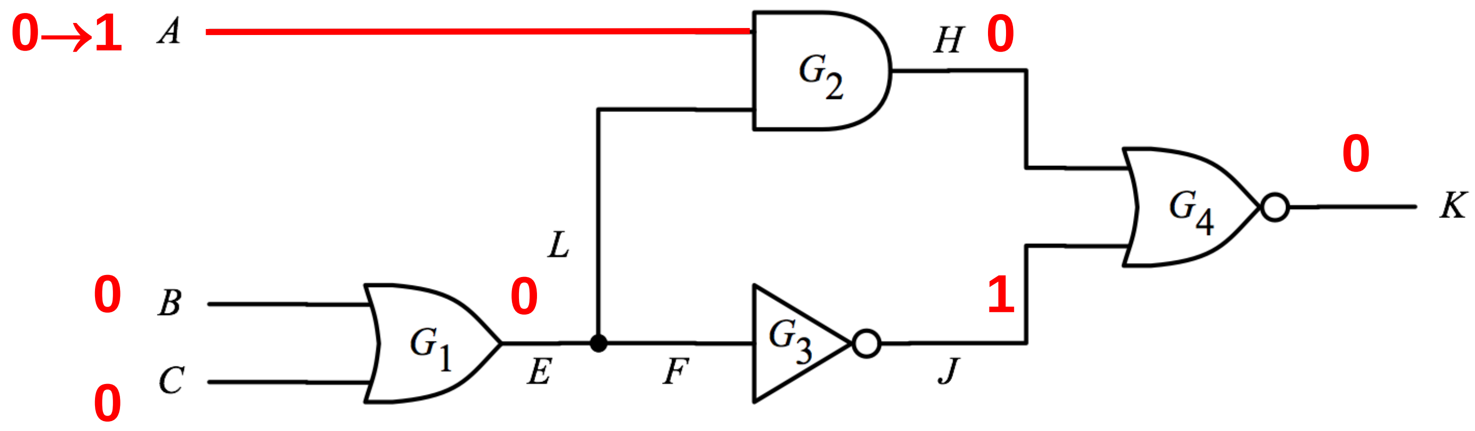
Logic Simulation

- Introduction
- Simulation Models
- <u>Logic Simulation Techniques</u>
  - Compiled-code simulation
  - <u>Event-driven simulation (1965)</u>
    - <u>Zero delay</u>
    - Nominal delay
    - Data structure
  - Parallel Simulation
- Issues of Logic Simulations
- Conclusions

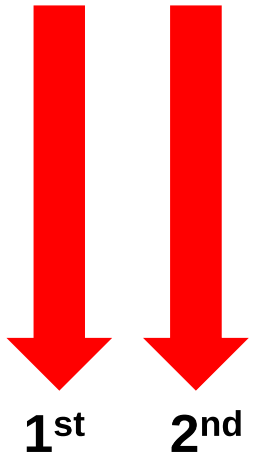


Compiled-code Simulation Problems

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



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



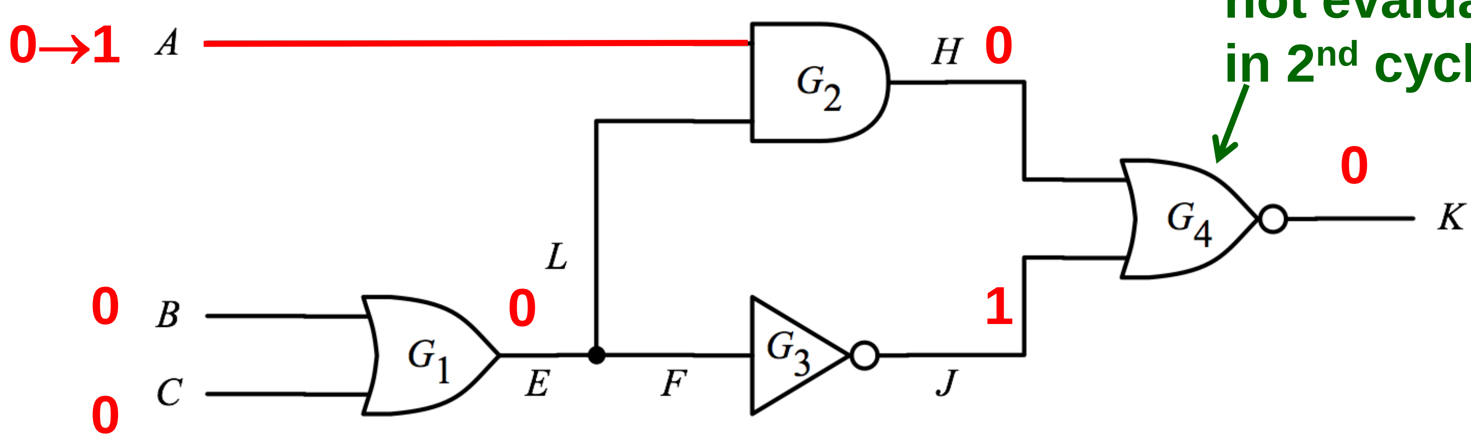
Forget 1st cycle results.

Rerun all codes for 2nd cycle.

Event-Driven Simulation [Ulrich 1965]

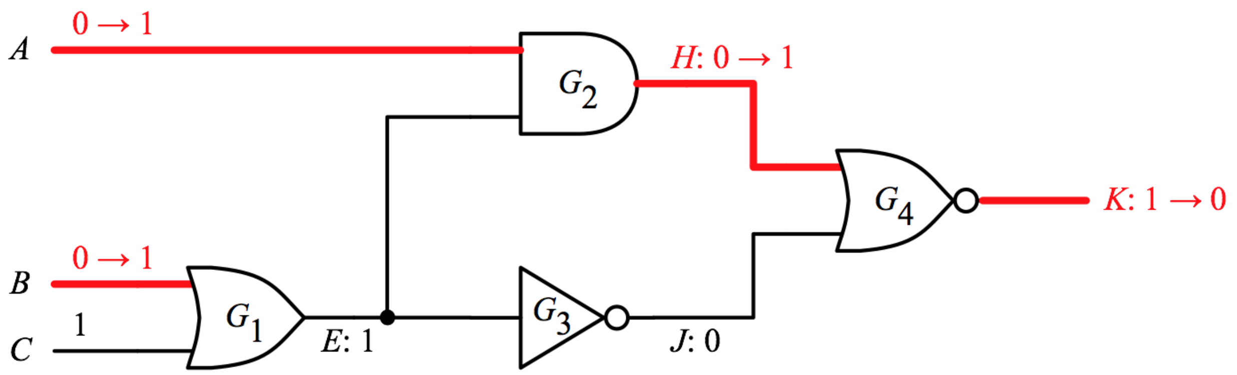
- IDEA: evaluates a gate only if there is event(s) at its gate inputs
  - *Event* is a signal value change (at time *t*)
  - Gates with inputs changed are *activated*
- Example:
  - Event: *A* 0→1
  - Activated gate: G_2
  - Only evaluate 1 gate, instead of 4 gates

G_1, G_3, G_4 not evaluated again in 2nd cycle



Event-driven Faster then CC

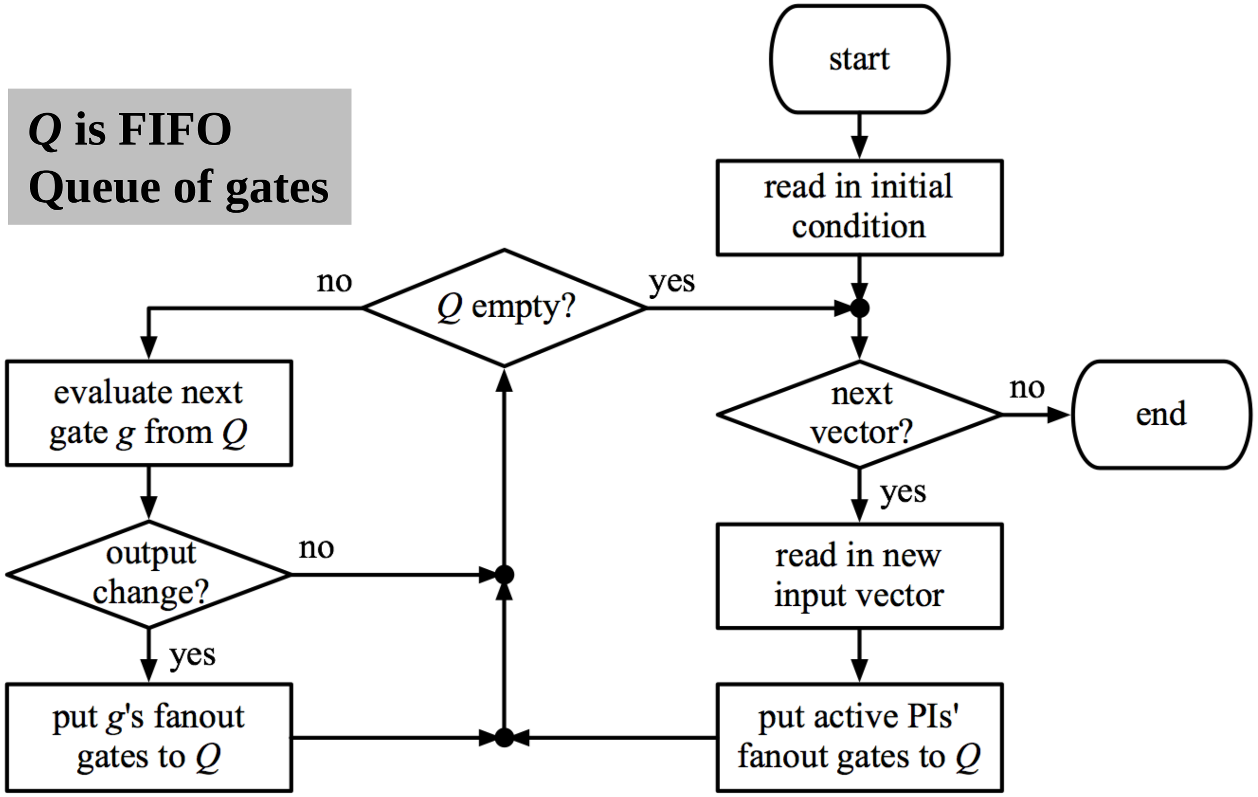
Zero-delay Event-driven Sim.



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
$(G_4,0)$	-

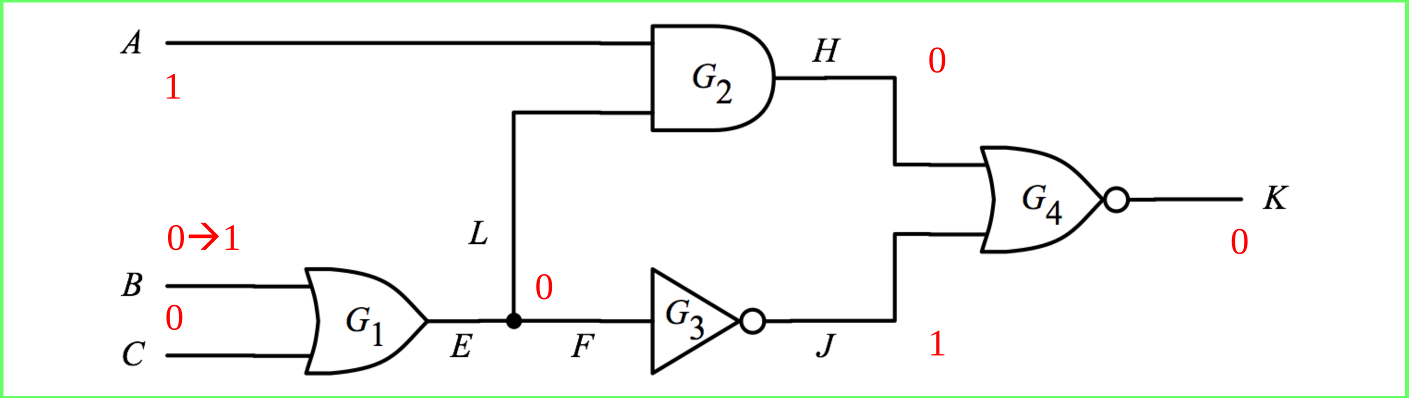


start
read in initial condition
Q empty?
no
yes
evaluate next gate g from Q
next vector?
no
end
yes
output change?
no
yes
read in new input vector
put g's fanout gates to Q
put active PIs' fanout gates to Q

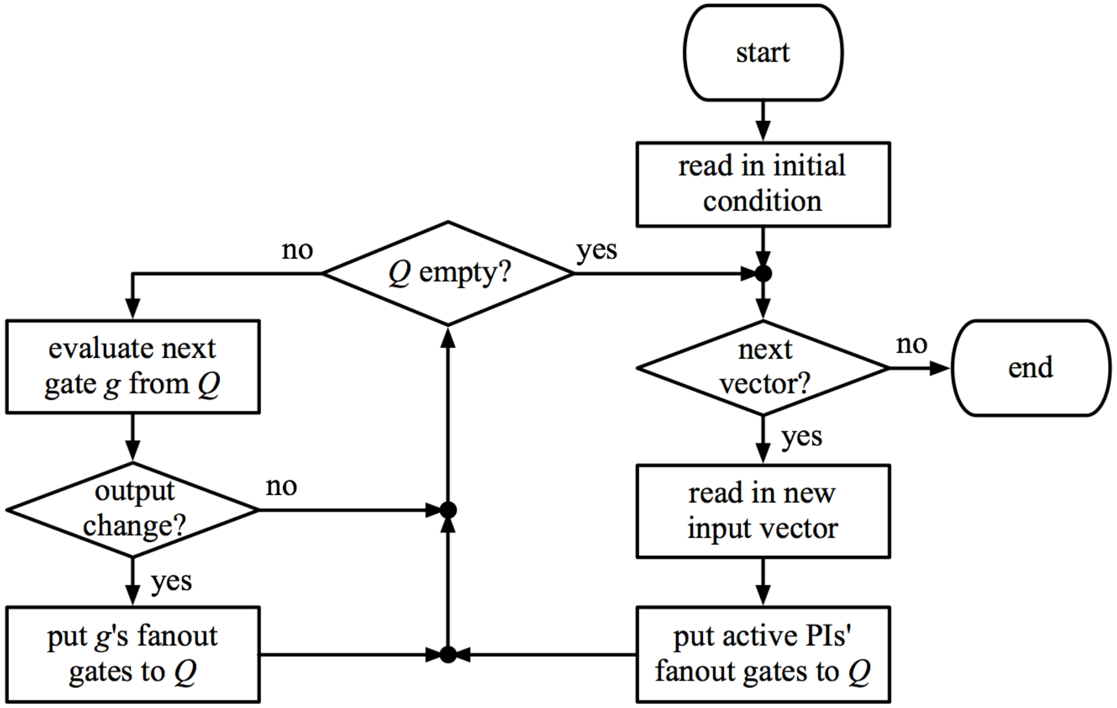
(WWW Fig 3.16)

Quiz

Please simulate this circuit

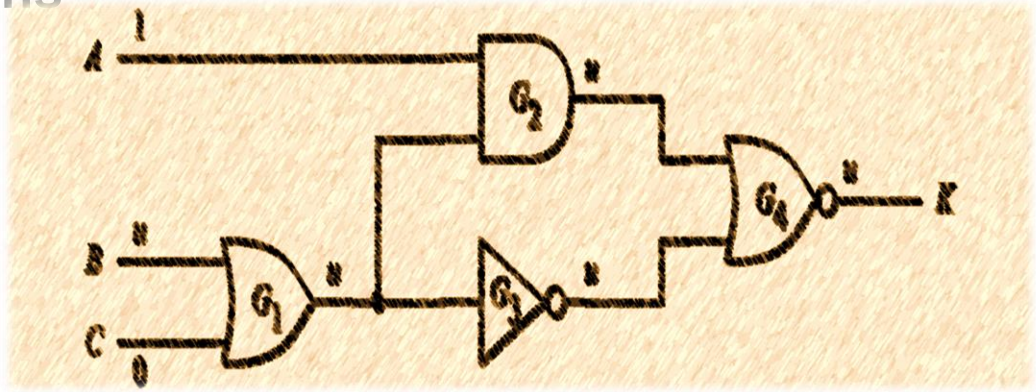


Executed Events	Q
(*B*,1)	*G_1*

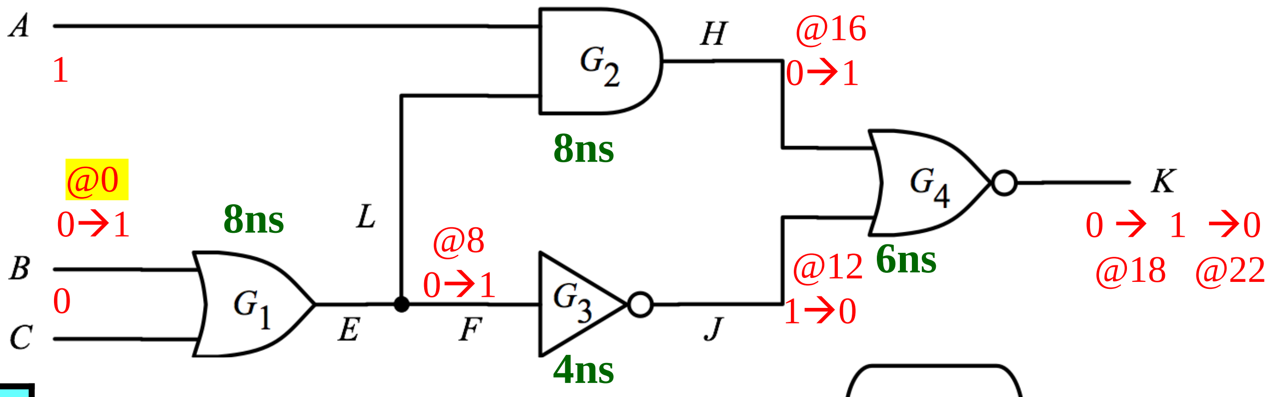

start
read in initial condition
Q empty?
no
yes
evaluate next gate g from Q
next vector?
no
end
yes
output change?
no
yes
read in new input vector
put g's fanout gates to Q
put active PIs' fanout gates to Q

Logic Simulation

- Introduction
- Simulation Models
- <u>Logic Simulation Techniques</u>
  - Compiled-code simulation
  - <u>Event-driven simulation (1965)</u>
    - Zero delay
    - <u>Nominal delay</u>
    - Data structure
  - Parallel Simulation
- Issues of Logic Simulations
- Conclusions



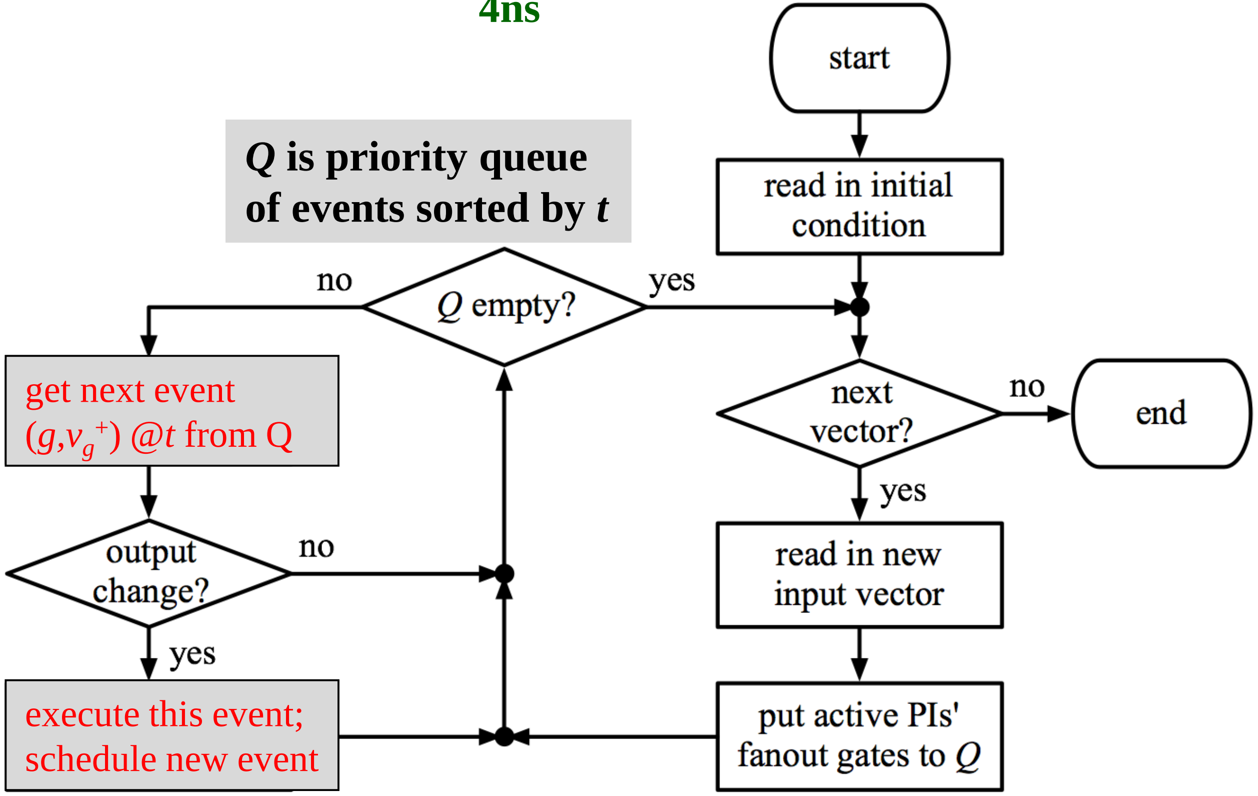
Can We Modify ZD Alg.?



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

t	Executed events	Q
-		(*B*,1)@0
0	(*B*,1)	(G_1,1)@8
8	(G_1,1)	(G_3,0)@12 (G_2,1)@16
12	(G_3,0)	(G_2,1)@16 (G_4,1)@18
16	(G_2,1)	(G_4,1)@18 (G_4,0)@22
18	(G_4,1)	(G_4,0)@22
22	(G_4,0)	--

Q is priority queue of events sorted by *t*


start
read in initial condition
Q empty?
no
yes
next vector?
no
end
yes
read in new input vector
put active PIs' fanout gates to Q
get next event (g,v_g^+) @t from Q
output change?
no
yes
execute this event; schedule new event

Problem! Simultaneous Input Change

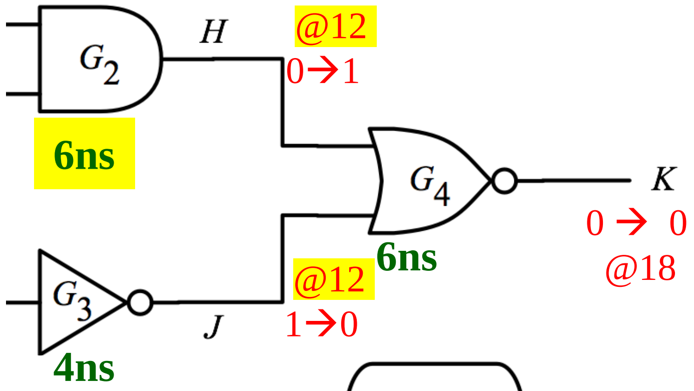
t	Executed events	Sch'd event
12	$(G_2,1)$	-- *(G4=0 output no change)*
12	$(G_3,0)$	-- *(G4=0 output no change)*

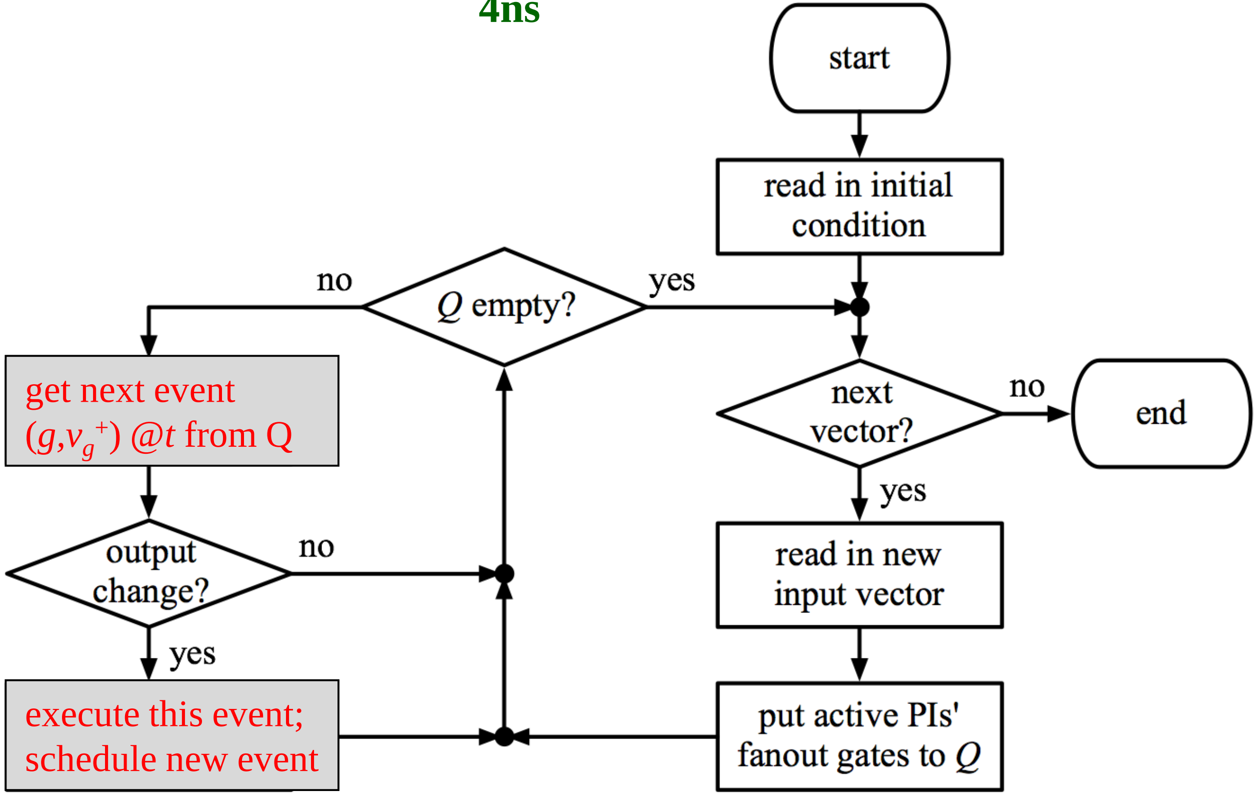
G_4=0 @18 Correct

t	Executed events	Sch'd event
12	$(G_3,0)$	$(G_4,1)$@18
12	$(G_2,1)$	-- *(G4=0 output no change)*
18	$(G_4,1)$	--

G_4=1 @18 Wrong!

SIC May Results in Different Outputs!



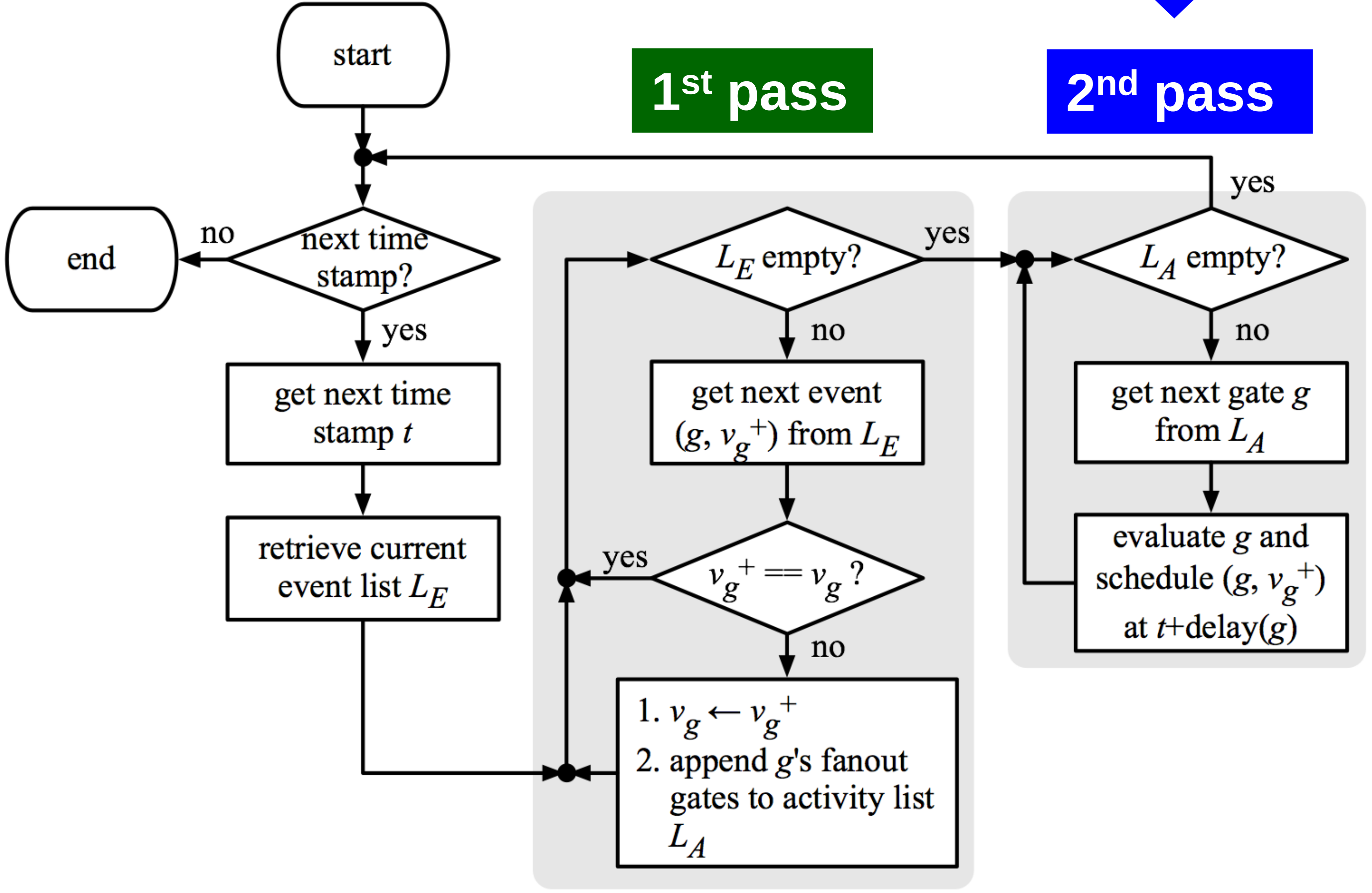


start
read in initial condition
Q empty?
no
yes
get next event (g,v_g^+) @t from Q
next vector?
no
end
yes
output change?
no
yes
read in new input vector
execute this event; schedule new event
put active PIs' fanout gates to Q

Sol: 2-pass Alg. [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

1. Evaluate gate inputs of a gate together (for SIC).
2. Schedule events no matter v_g changes or not (for hazards).

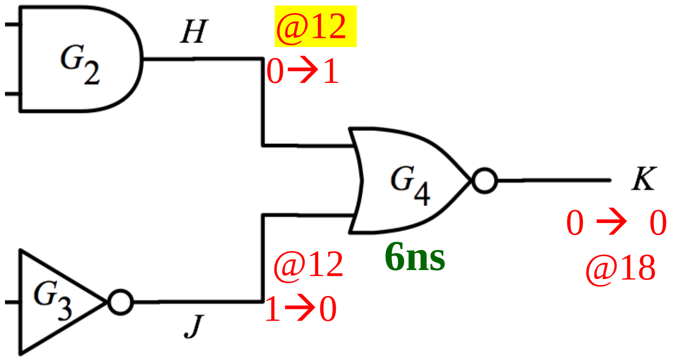


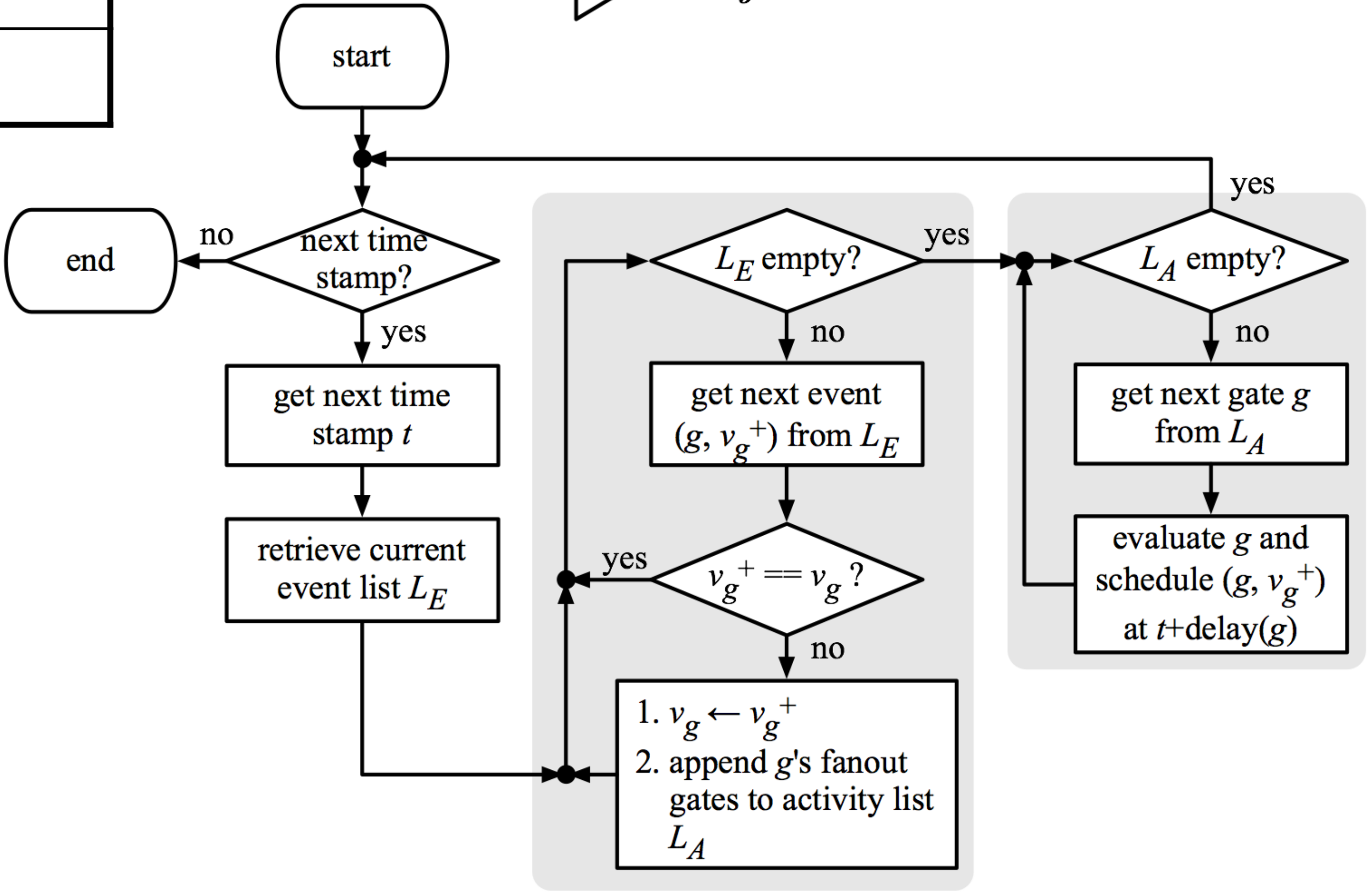
start
1st pass
2nd pass
yes
end
no
next time stamp?
yes
get next time stamp t
retrieve current event list L_E
L_E empty?
yes
no
get next event (g, v_g^+) from L_E
yes
$v_g^+ == v_g$?
no
1. $v_g \leftarrow v_g^+$
2. append g's fanout gates to activity list L_A
L_A empty?
no
get next gate g from L_A
evaluate g and schedule (g, v_g^+) at t+delay(g)

SIC Problem Solved

t	L_E	L_A	sch'd event
12	$(G_3,0)$ $(G_2,1)$	G_4	$(G_4,0)$@18
18	$(G_4,0)$	-	-

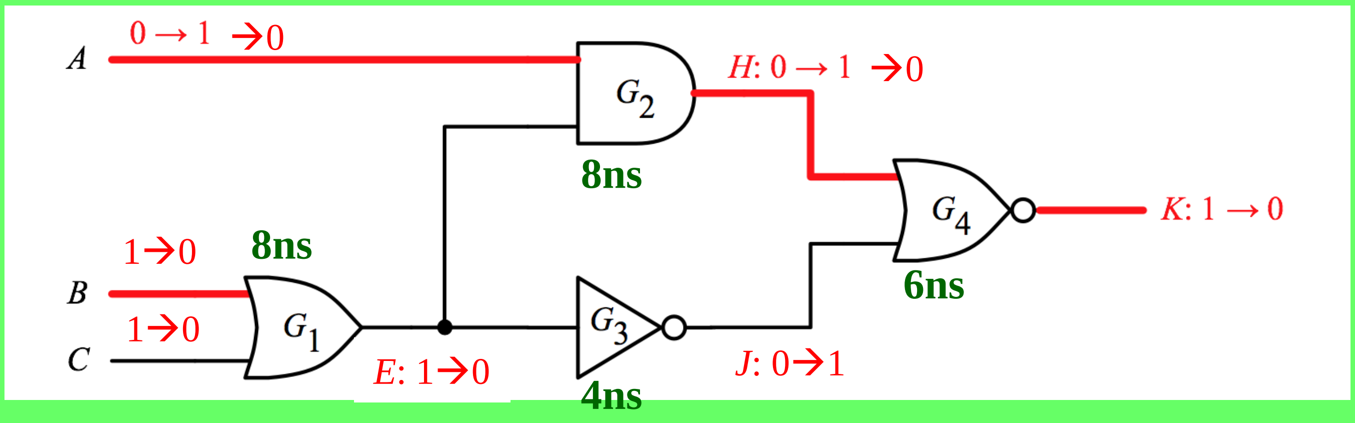
G_4=0 @18 Correct



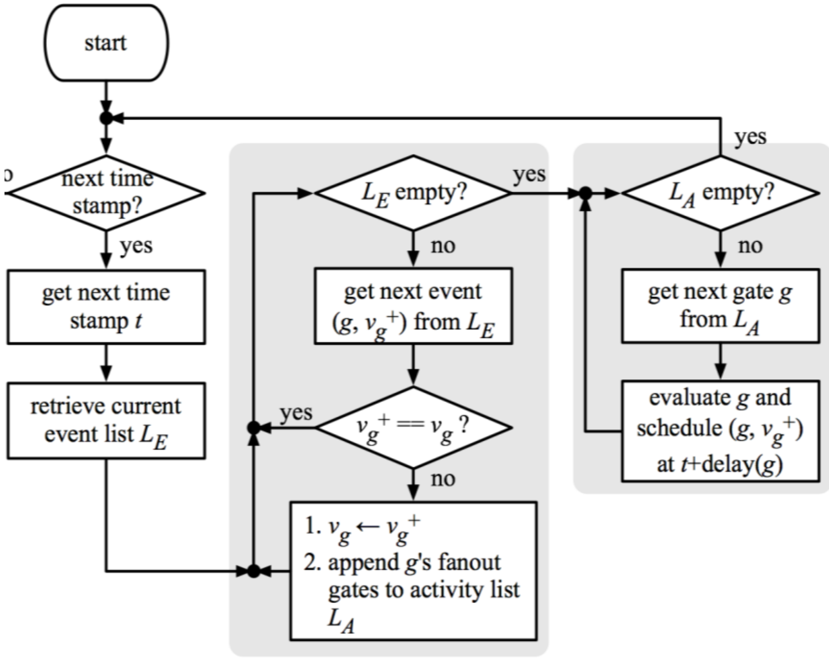


start
end
no
next time stamp?
yes
get next time stamp t
retrieve current event list L_E
L_E empty?
yes
no
get next event (g, v_g^+) from L_E
$v_g^+ == v_g$?
yes
no
1. $v_g \leftarrow v_g^+$
2. append g's fanout gates to activity list L_A
L_A empty?
yes
no
get next gate g from L_A
evaluate g and schedule (g, v_g^+) at t+delay(g)

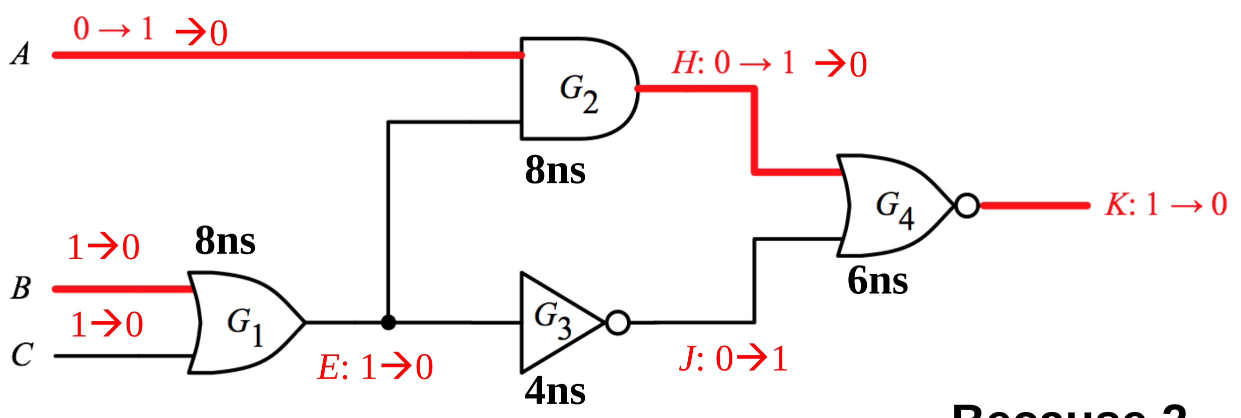
Quiz



t	L_E	L_A	Scheduled events
-			*(A,1)@0 (C,0)@2 (B,0)@4 (A,0)@8*
0	*(A,1)*	*G_2*	*$(G_2,1)$@8*
2	[redacted]	[redacted]	[redacted]
4	*(B,0)*	*G_1*	*$(G_1,0)$@12*
8	*$(A,0)(G_2,1)$*	*G_2, G_4*	*$(G_4,0)$@14 $(G_2,0)$@16*
10	[redacted]	[redacted]	[redacted]
12	[redacted]	[redacted]	[redacted]
14	*$(G_4,0)$*		
16	*$(G_2,0)(G_3,1)$*	*G_4*	*$(G_4,0)$@22*
20	*$(G_2,0)$*		
22	*$(G_4,0)$*		

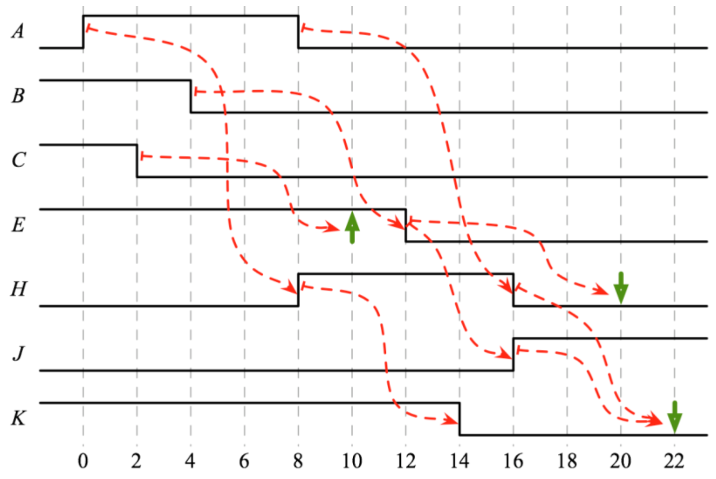

start
next time stamp?
yes
get next time stamp t
retrieve current event list L_E
L_E empty?
yes
no
get next event (g, v_g^+) from L_E
$v_g^+ == v_g$?
yes
no
1. $v_g \leftarrow v_g^+$
2. append g's fanout gates to activity list L_A
L_A empty?
yes
no
get next gate g from L_A
evaluate g and schedule (g, v_g^+) at t+delay(g)

FFT: How to Remove False Events?



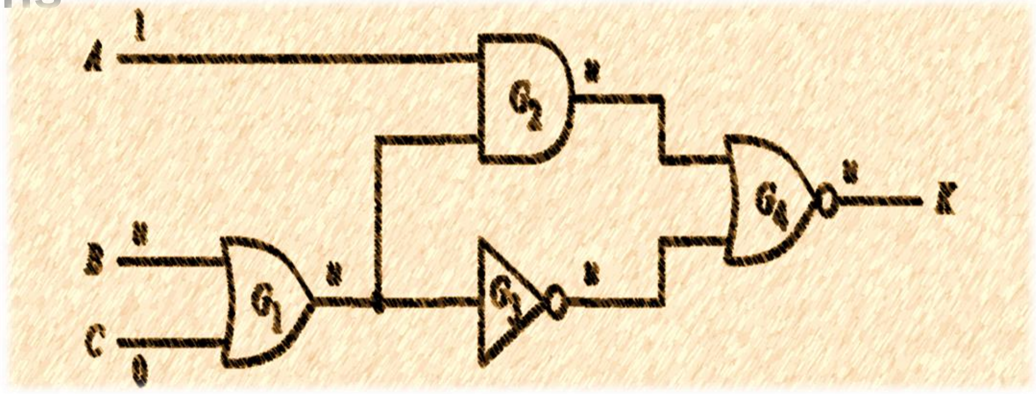
t	L_E	L_A	Scheduled events
-			*(A,1)@0 (C,0)@2 (B,0)@4 (A,0)@8*
0	*(A,1)*	*G_2*	*(G_2,1)@8*
2	*(C,0)*	*G_1*	*(G_1,1)@10*
4	*(B,0)*	*G_1*	*(G_1,0)@12*
8	*(A,0)(G_2,1)*	*G_2, G_4*	*(G_4,0)@14 (G_2,0)@16*
10	*(G_1,1)*		
12	*(G_1,0)*	*G_2, G_3*	*(G_3,1)@16 (G_2,0)@20*
14	*(G_4,0)*		
16	*(G_2,0)(G_3,1)*	*G_4*	*(G_4,0)@22*
20	*G_2,0)*		
22	*(G_4,0)*		

Because 2_{nd} pass schedule events no matter v_g changes or not, many *False Events* waste CPU time.



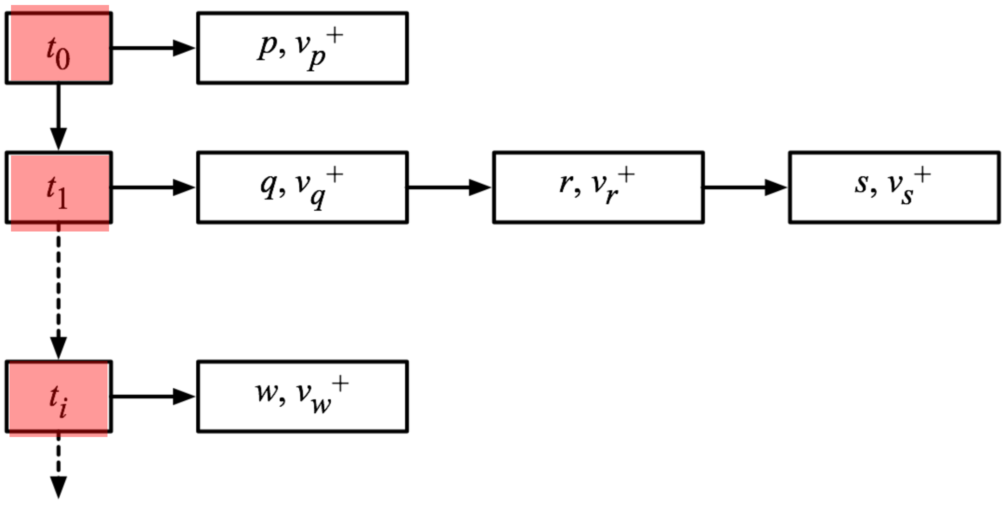
Logic Simulation

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

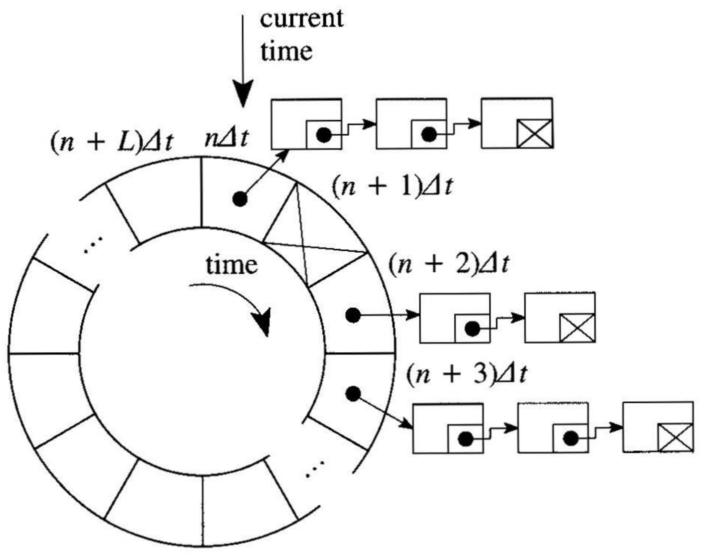


How to Implement Event List ?

1. Linked list
   - ☺ *t* is flexible
   - ☹ Slower to search

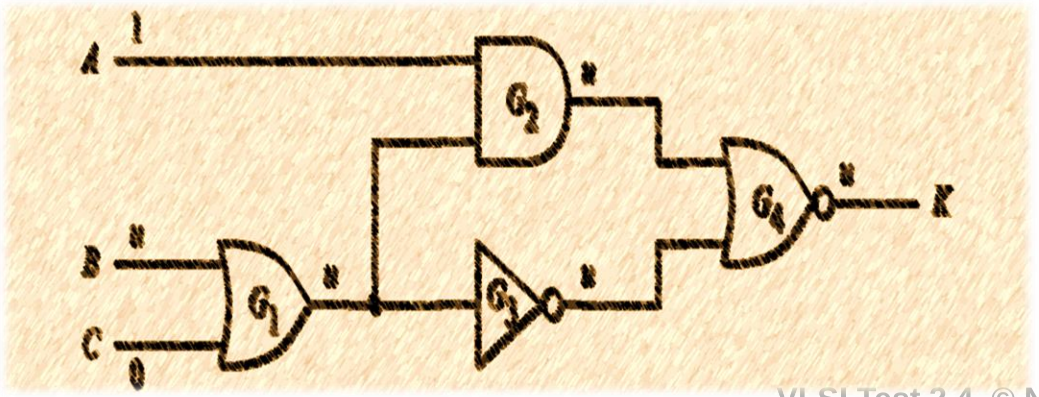


2. Cyclic Array *(aka. Timing Wheel*) [Ulrich 1969]
   - ☺ Faster to search
   - ☹ Δ*t* is fixed
   - ☹ Limited *L* slots in a cycle



Summary

- Event-driven simulation
  - Consider gate delay
  - Saves CPU time. Only evaluate gates with events at input.
- Two scenarios
  - Zero delay: 1-pass
  - Nominal delay: 2-pass (w/ false events)
- Event list implementation
  - Linked list: slow but flexible
  - Cyclic Array (Timing wheel): fast but fixed time slots



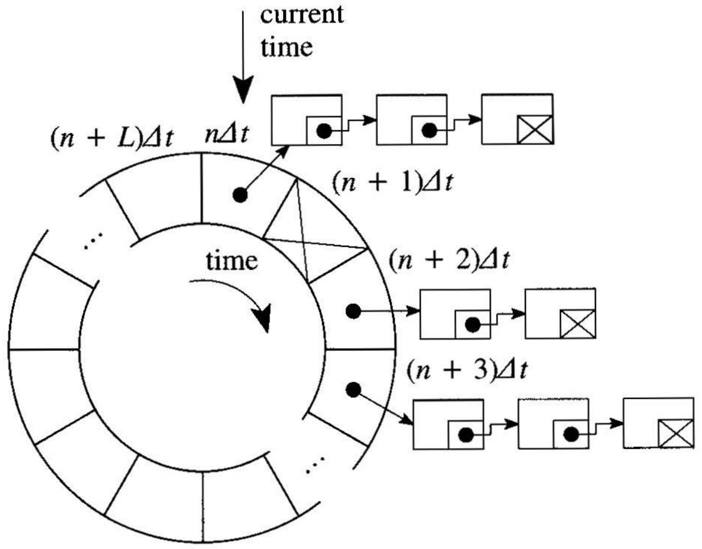
Comparison

	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

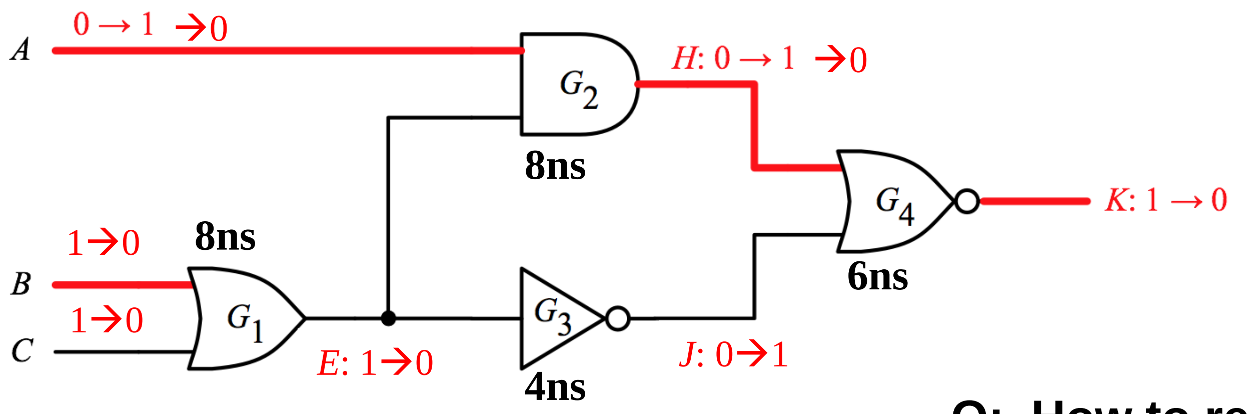
ED is Generally Better than CC

FFT #1

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

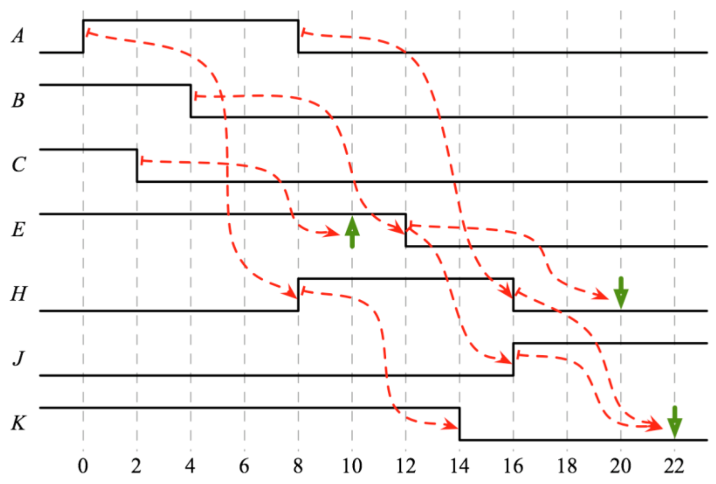


FFT#2: False Events



Q: How to remove false events?

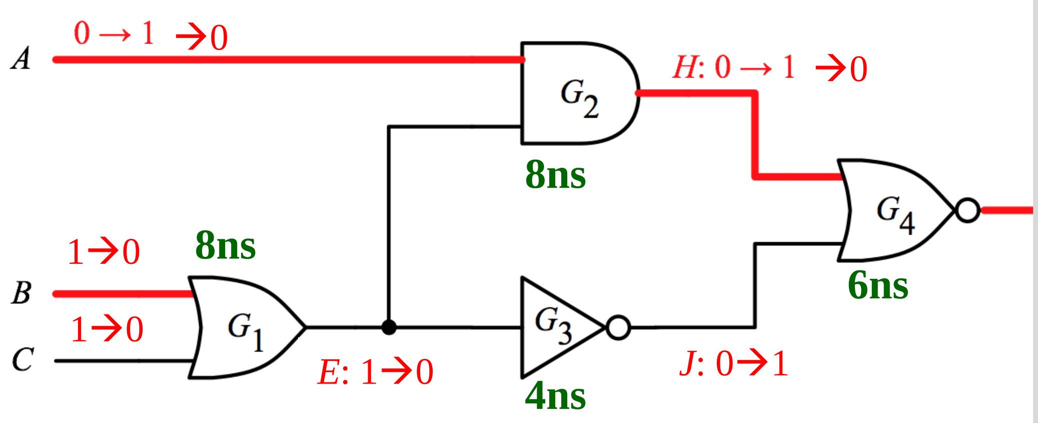
t	L_E	L_A	Scheduled events
-			$(A,1)@0$ $(C,0)@2$ $(B,0)@4$ $(A,0)@8$
0	$(A,1)$	G_2	$(G_2,1)@8$
2	$(C,0)$	G_1	$(G_1,1)@10$
4	$(B,0)$	G_1	$(G_1,0)@12$
8	$(A,0)(G_2,1)$	G_2, G_4	$(G_4,0)@14$ $(G_2,0)@16$
10	$(G_1,1)$		
12	$(G_1,0)$	G_2, G_3	$(G_3,1)@16$ $(G_2,0)@20$
14	$(G_4,0)$		
16	$(G_2,0)(G_3,1)$	G_4	$(G_4,0)@22$
20	$G_2,0)$		
22	$(G_4,0)$		



Sol: Modified 1-pass Alg. [Ulrich 1969]

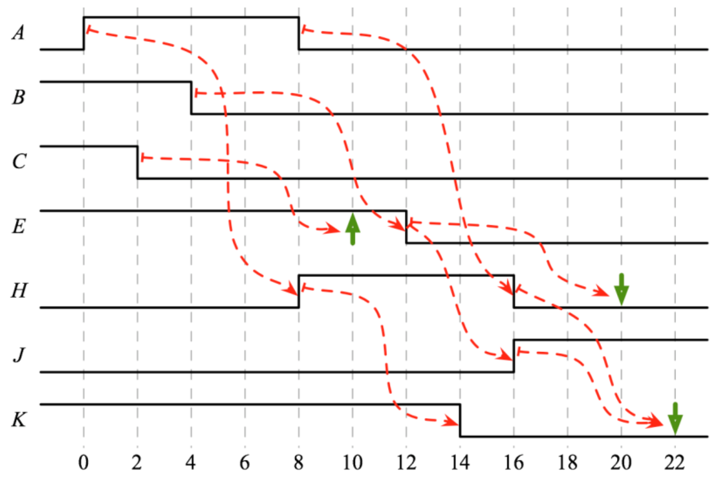
- Add last scheduled value (*lsv*), last scheduled time (*lst*)
- Cancel previously scheduled events when needed

```
1-pass-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.          $v_j^+$ = evaluate ($j$)
 6.          if  $v_j^+ \neq$ lsv($j$)        /*only schedule events different from lsv */
 7.              $t^+ = t + d_j$
 8.              if  $t^+$ == lst($j$)   /* simultaneous  opposite events */
 9.                  cancel event ($j$, lsv($j$)) @ $t^+$
10.              schedule ($j$, $v_j^+$) @ $t^+$
11.              lsv($j$) = $v_j^+$     /* remember lav and lst */
12.              lst($j$) = $t^+$
```



t	L_E	L_A	Scheduled events
-			$(A,1)@0$ $(C,0)@2$ $(B,0)@4$ $(A,0)@8$
0	$(A,1)$	G_2	$(G_2,1)@8$
2	$(C,0)$	G_1	~~$(G_1,1)@10$~~ assume *lsv=1*
4	$(B,0)$	G_1	$(G_1,0)@12$
8	$(A,0)(G_2,1)$	G_2, G_4	$(G_4,0)@14$ $(G_2,0)@16$
10	~~$(G_1,1)$~~		
12	$(G_1,0)$	G_2, G_3	$(G_3,1)@16$ ~~$(G_2,0)@20$~~
14	$(G_4,0)$		
16	$(G_2,0)(G_3,1)$	G_4	~~$(G_4,0)@22$~~
20	~~$(G_2,0)$~~		
22	~~$(G_4,0)$~~		

```
1-pass-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.         $v_j^+$ = evaluate ($j$)
 6.         if $v_j^+ \neq \text{lsv}(j)$        /* compare with lsv */
 7.             $t^+ = t + d_j$
 8.             if $t^+ == \text{lst}(j)$   /* remove opposite events */
 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 */
```



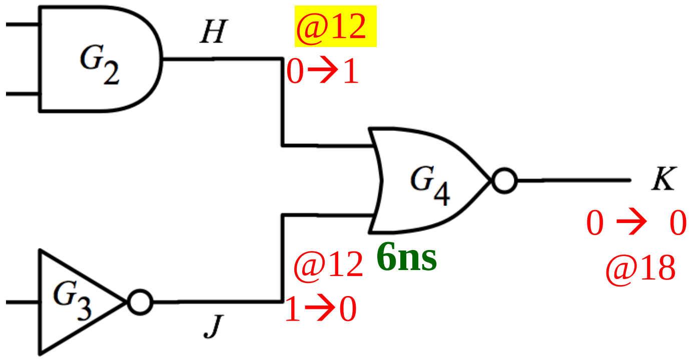
SIC Cancelled

t	Executed events	Sch'd event
12	$(G_2,1)$	$v_{G4}^+ = lsv(G_4) = 0$ *no event sch'd*
12	$(G_3,0)$	$v_{G4}^+ = lsv(G_4) = 0$ *no event sch'd*

G_4=0 @18 Correct

t	Executed events	Sch'd event
12	$(G_3,0)$	$(G_4,1)$@18 $lsv(G_4)=1$
12	$(G_2,1)$	$v_{G4}^+ \neq lsv(G_4)$ ***Cancelled***
18	--	--

G_4=0 @18 Correct



```
1-pass-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.          $v_j^+$ = evaluate (j)
6.          if $v_j^+ \neq$ lsv(j)        /* compare with lsv */
7.              $t^+ = t + d_j$
8.              if $t^+$ == lst(j)   /* simultaneous  opposite events */
9.                  cancel event (j, lsv(j)) @ $t^+$
10.             schedule (j, $v_j^+$) @ $t^+$
11.             lsv(j) = $v_j^+$
12.             lst(j) = $t^+$        /* lst = last scheduled time */
```