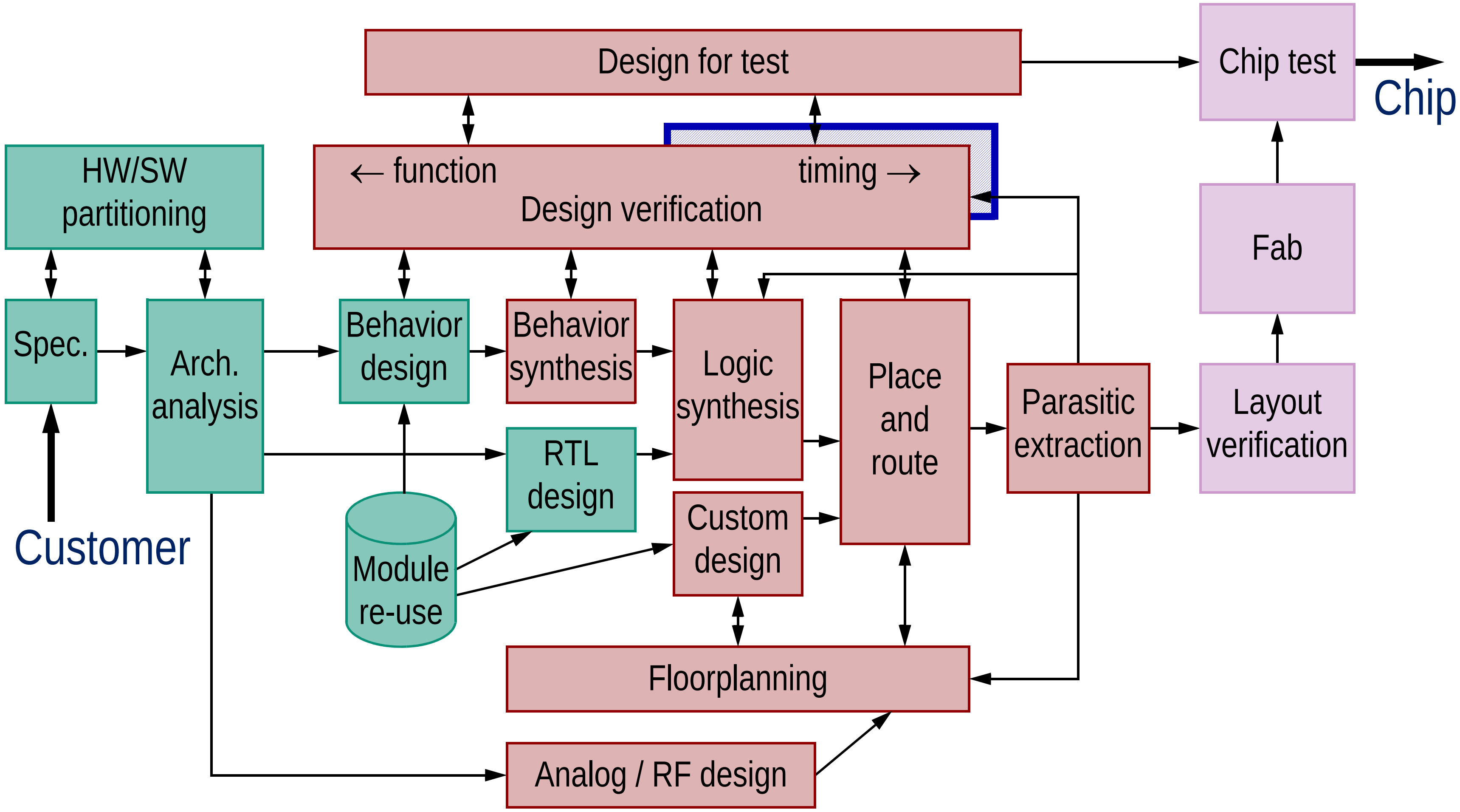
DAT110
Methods for Electronic System Design and Verification

Per Larsson-Edefors
VLSI Research Group

Lecture 5: Timing.

Present Scenario: Timing Verification



Design Flow

- *Conventional design flow for ASICs [from Lecture 1].*
  1. *RTL-specification $\rightarrow$ Generic gate netlist.*
  2. *Generic gate netlist $\rightarrow$ Cell library.*
  3. *Cell library $\rightarrow$ Placement & routing.*
- Verification flow.
  1. Functional verification of VHDL; testbench + RTL code.
  2. Functional verification of netlist that results from synthesis of RTL VHDL to cell library.
  3. **Timing verification of netlist.**
  4. Place&route and further verification...

Delay Models for Event-Driven Simulation

- ♦ Unit-delay models ...
  - make all gates have the same delay; a delay of 1 unit.
  - are slower than zero-delay models,
    but faster than full-delay models.
  - can uncover logic mismatches between gates and RTL,
    and problems with, for example, reset signals.
- ♦ Zero-delay models ...
  - yield something similar to cycle-based simulation;
    the difference being that only nodes with events are updated.
- ♦ Full-delay models ...

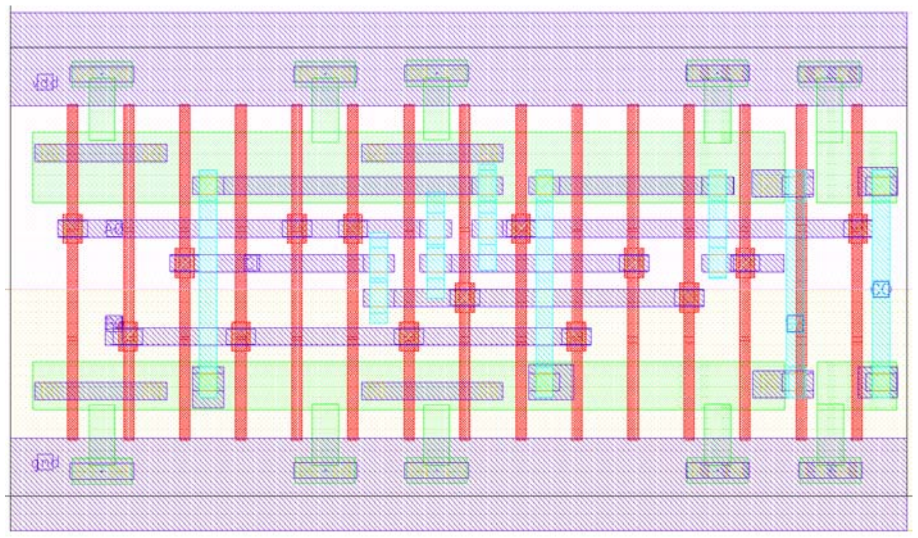
UNIT DELAY MODEL

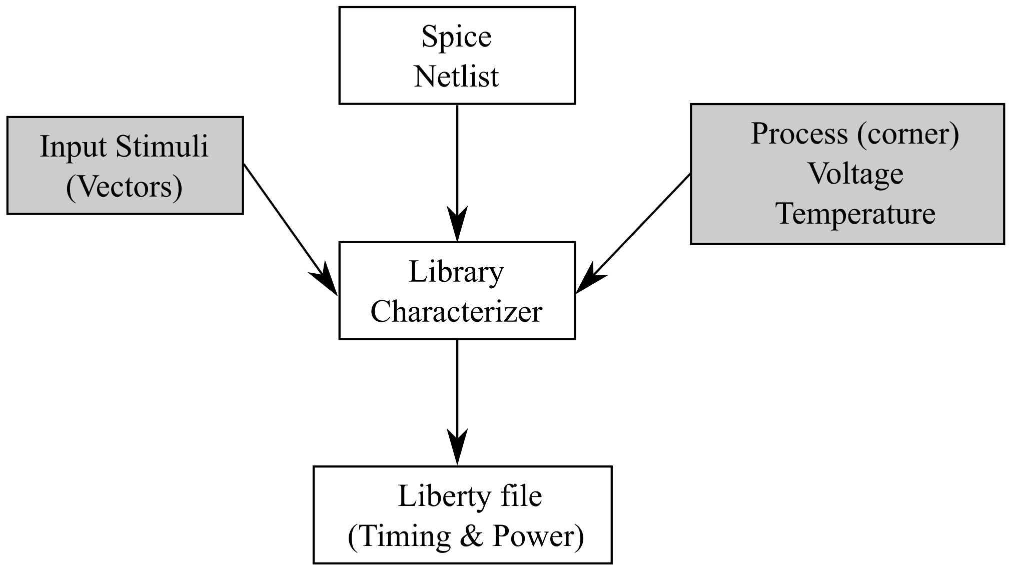
```
entity HS65_LSS_DFPHQNX18 is
   port( QN : out STD_LOGIC ; CP : in STD_LOGIC ... );
end HS65_LSS_DFPHQNX18;

architecture VHDL_FUNCT of HS65_LSS_DFPHQNX18 is
...
begin
PIQ : Process (CP)
   begin
      if rising_edge(CP) Then
      ...
   end Process;
   QN <= to_X01(not IQ)   after 1 ns;
end VHDL_FUNCT;
```

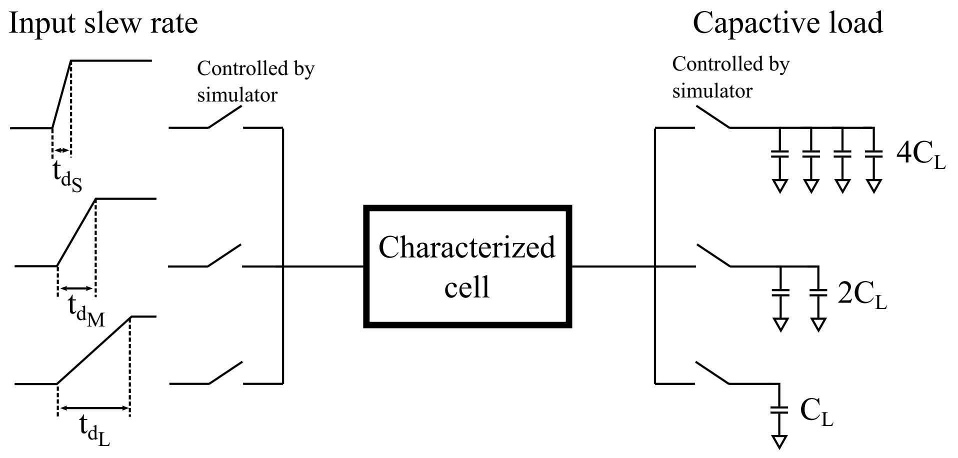
FULL-DELAY MODELS

- A more accurate delay model should consider input signal slope and output load capacitance: The full-delay model.
- The full-delay models are obtained from circuit characterization and model fitting.

Characterization Flow



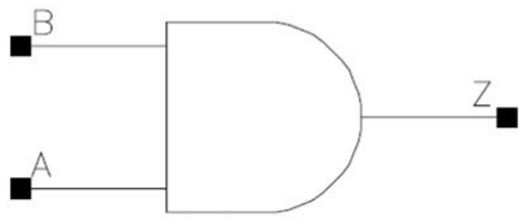

Spice Netlist
Input Stimuli (Vectors)
Process (corner) Voltage Temperature
Library Characterizer
Liberty file (Timing & Power)



Cell-Library Model Example

HS65_LS_AND2

Logical Symbol



Intrinsic delay - no output load.

Kload multiplier - load dependent delay.

(Table assumes input slew of 0.020 ns)

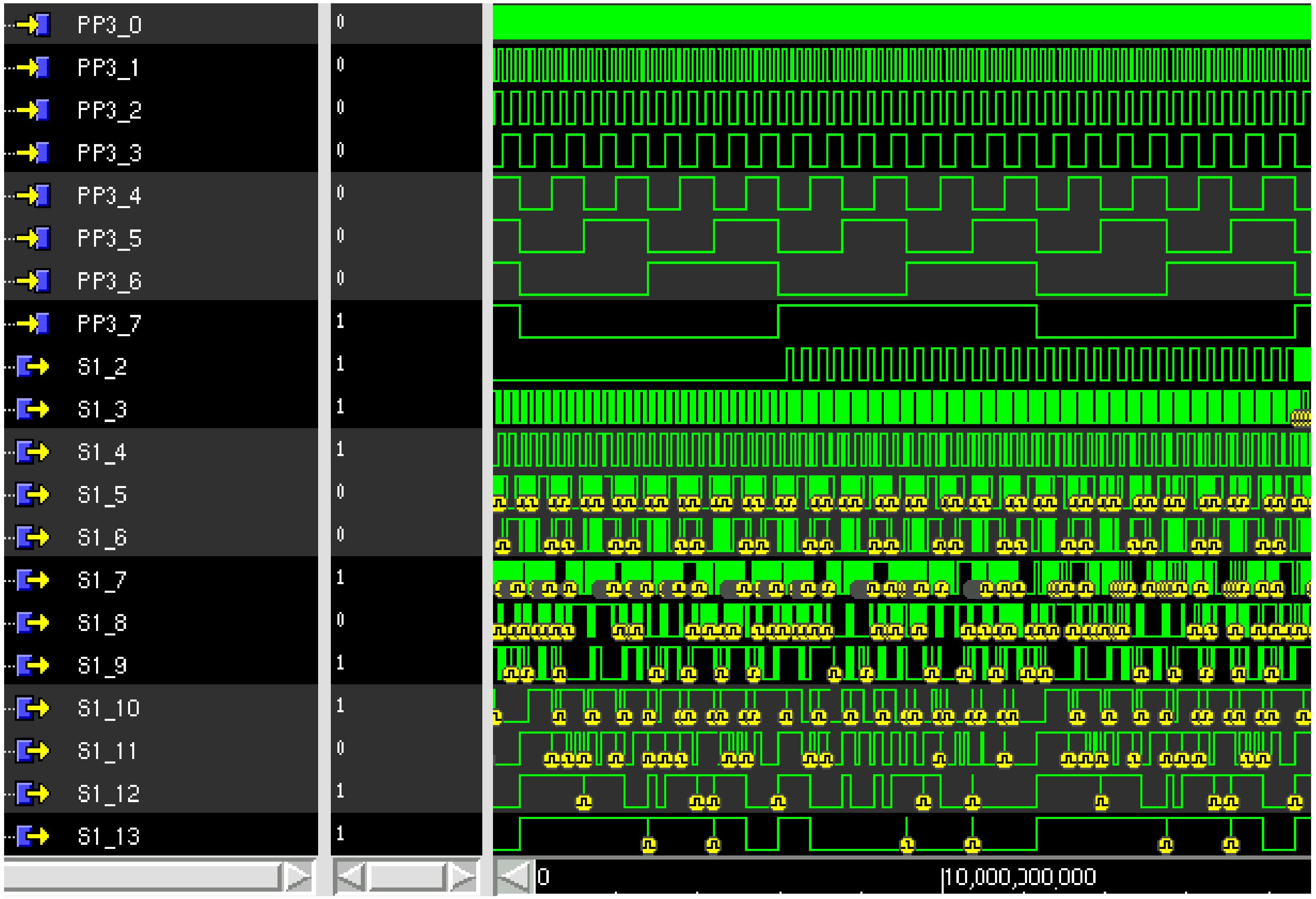
Propagation Delay at 25C, 1.20V Typ process

Description	Intrinsic Delay (ns)					Kload (ns/pf)				
	X4	X9	X18	X27	X35	X4	X9	X18	X27	X35
A to Z↓	0.0413	0.0360	0.0338	0.0332	0.0331	2.3369	1.1767	0.5785	0.3870	0.2878
A to Z↑	0.0454	0.0409	0.0368	0.0364	0.0359	3.2251	1.6479	0.8227	0.5452	0.4093
B to Z↓	0.0391	0.0341	0.0317	0.0307	0.0308	2.3337	1.1753	0.5776	0.3858	0.2872
B to Z↑	0.0441	0.0392	0.0349	0.0339	0.0335	3.2272	1.6489	0.8234	0.5456	0.4095

Two Sorts of Timing Analysis

- Static timing analysis (STA).
- Simulation-based timing analysis
  [Lab exercise 2.3] using SDF information.

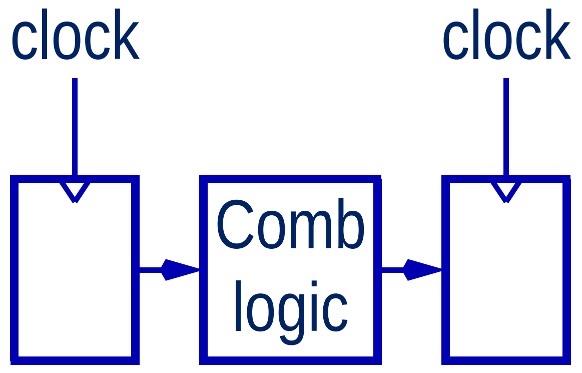
Timing Information also at Logic Level



PP3_0	0
PP3_1	0
PP3_2	0
PP3_3	0
PP3_4	0
PP3_5	0
PP3_6	0
PP3_7	1
S1_2	1
S1_3	1
S1_4	1
S1_5	0
S1_6	0
S1_7	1
S1_8	0
S1_9	1
S1_10	1
S1_11	0
S1_12	1
S1_13	1

Finding the Delay of a Circuit

- We want to know the critical path delay between register boundaries (combinational logic).



- Two options.
  - Simulation (dynamic and iterative).
  - Static timing analysis (static and deterministic).

Simulation vs Static Timing Analysis (STA)

Simulation	STA
Limited coverage (node wise)	High coverage (node wise)
Very high accuracy	Limited accuracy
Long run-time	Short run-time
Limited capacity	High capacity
Logic functions are evaluated	Logic functions are not evaluated
Test vectors are required	No test preparation

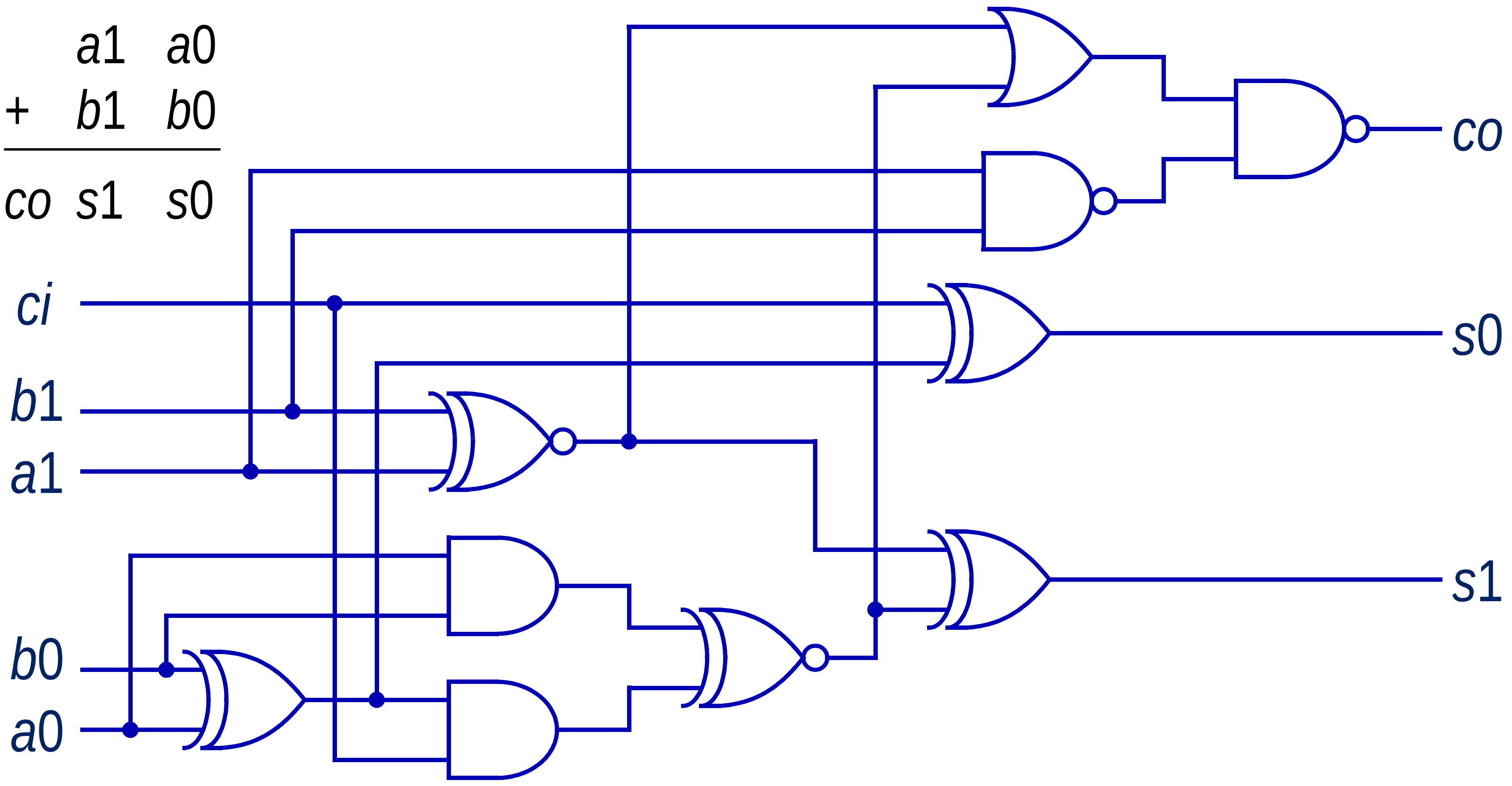
Basic Static Timing Analysis

- Priority 1 is to obtain information on ...
  - the critical path delay.
  - the critical path (which are the gates on this path).
- How is the critical path identified? What algorithm can be used?

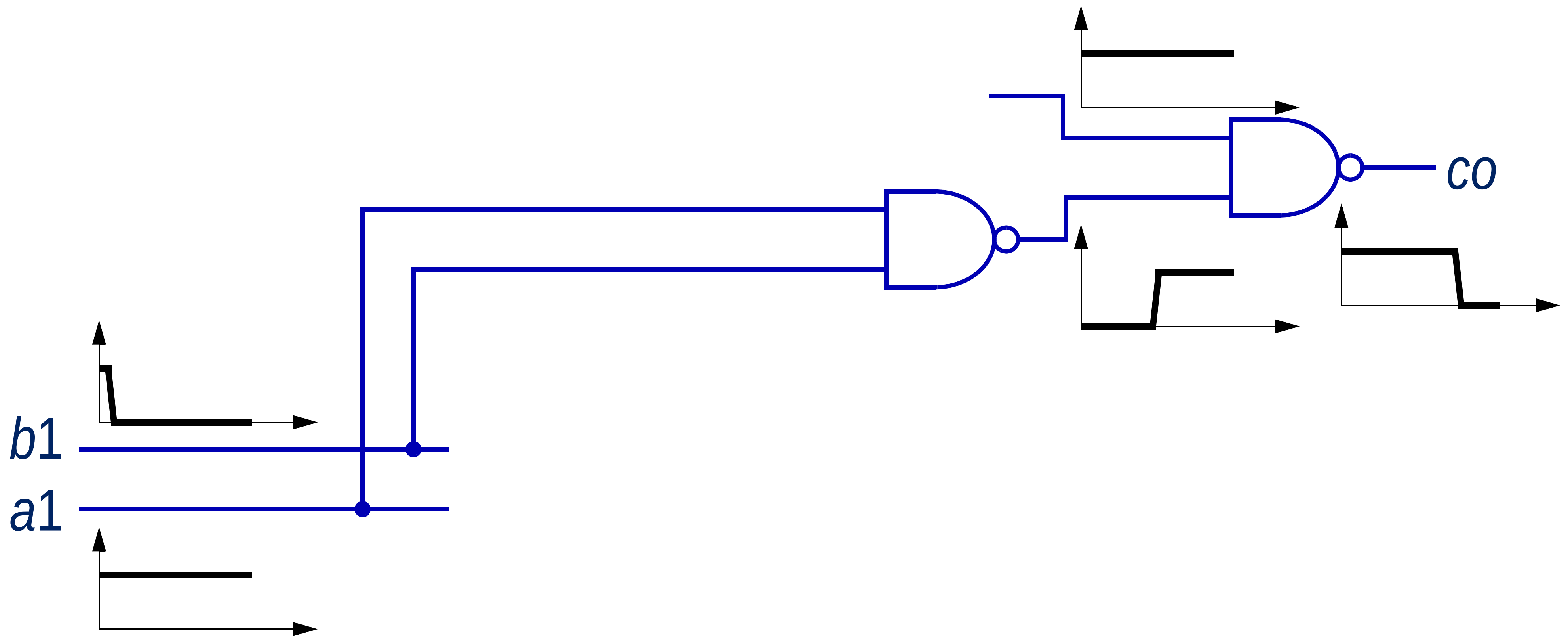
The following STA netlist example was adopted from a course from Technion: EE-046880 “CAD of VLSI systems”.

A Circuit for Radix-4 Addition

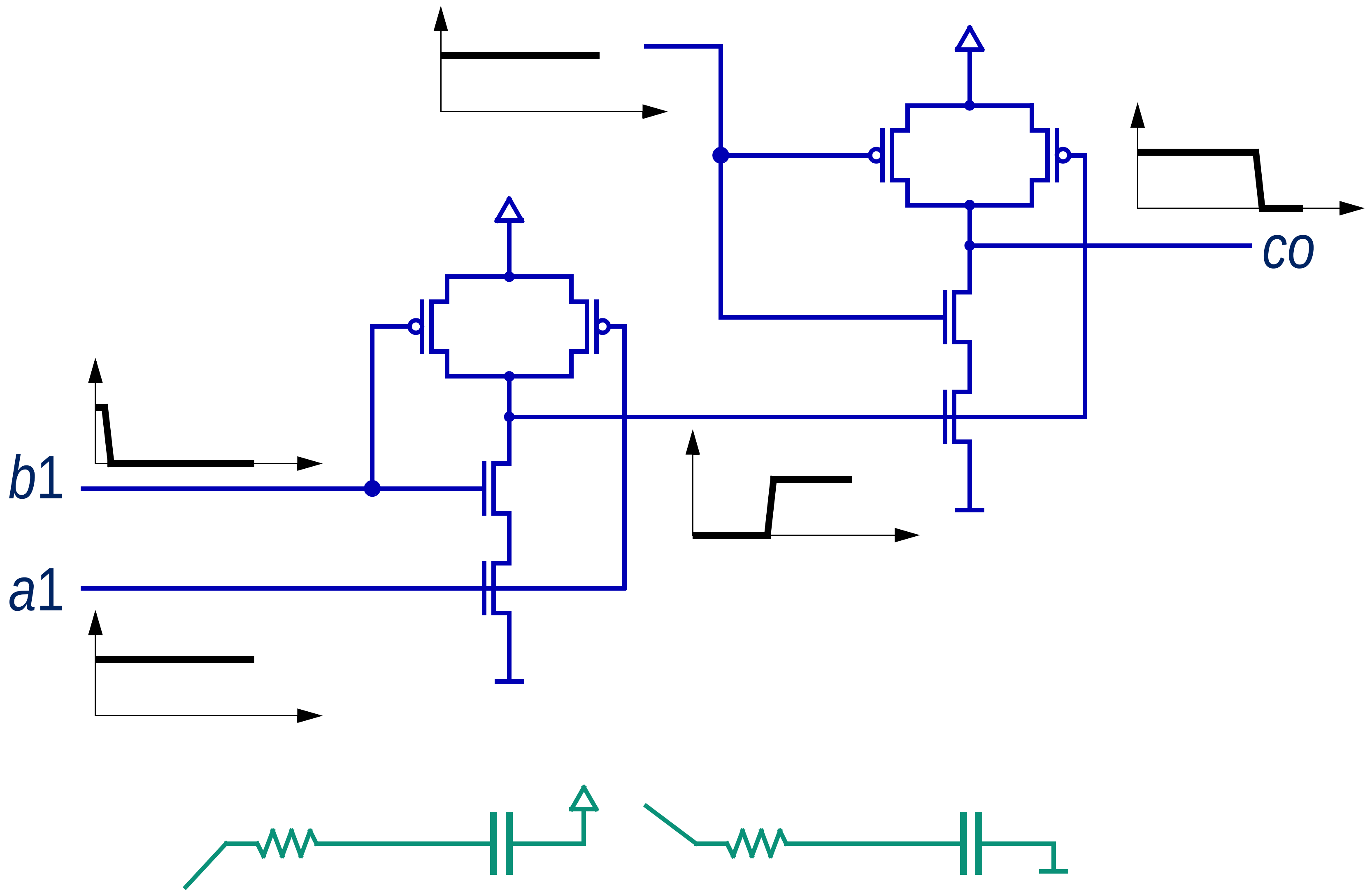
$$\begin{array}{ccc} & & ci \\ & a1 & a0 \\ + & b1 & b0 \\ \hline co & s1 & s0 \end{array}$$



Example of Signal Propagation

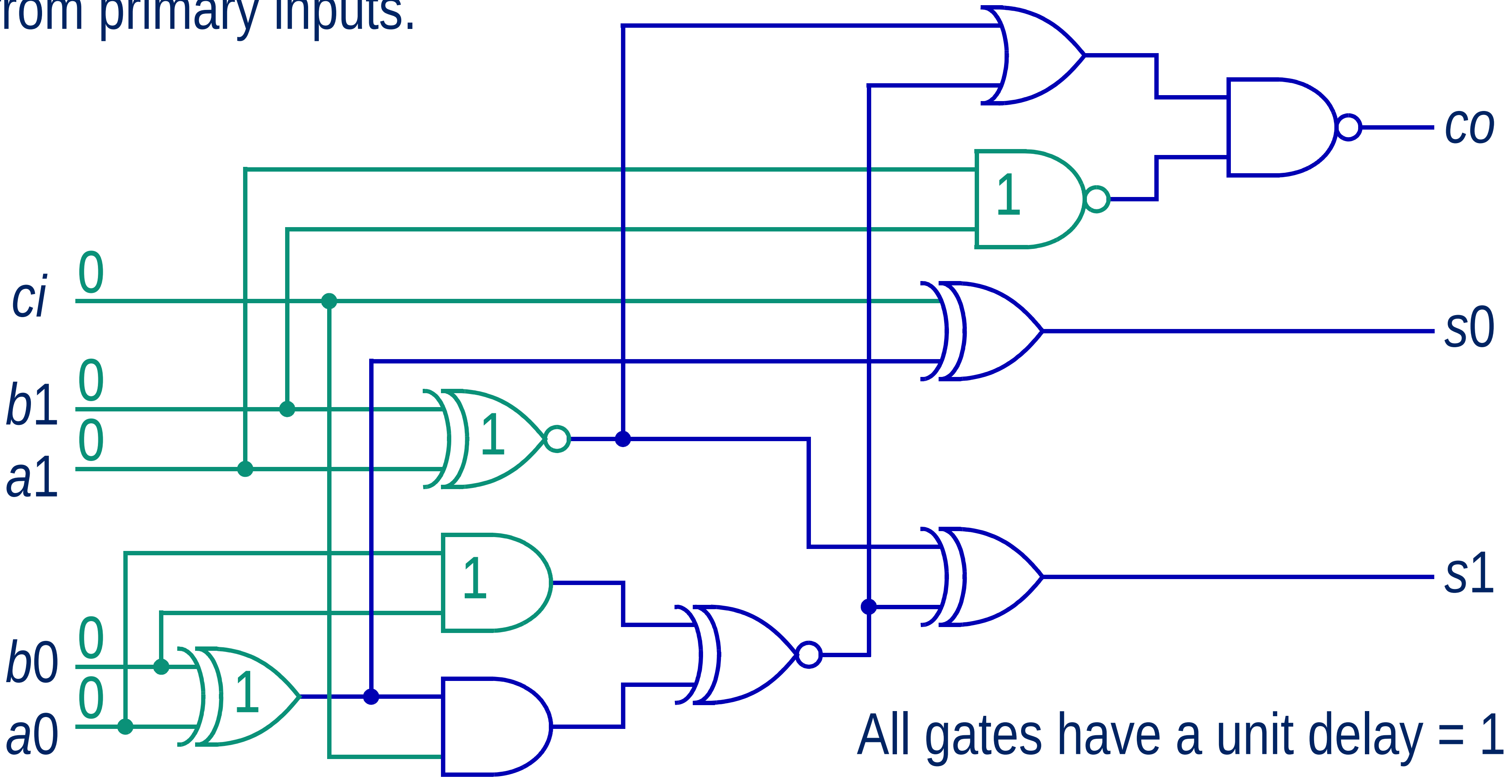


Transistor-Level View of Propagation



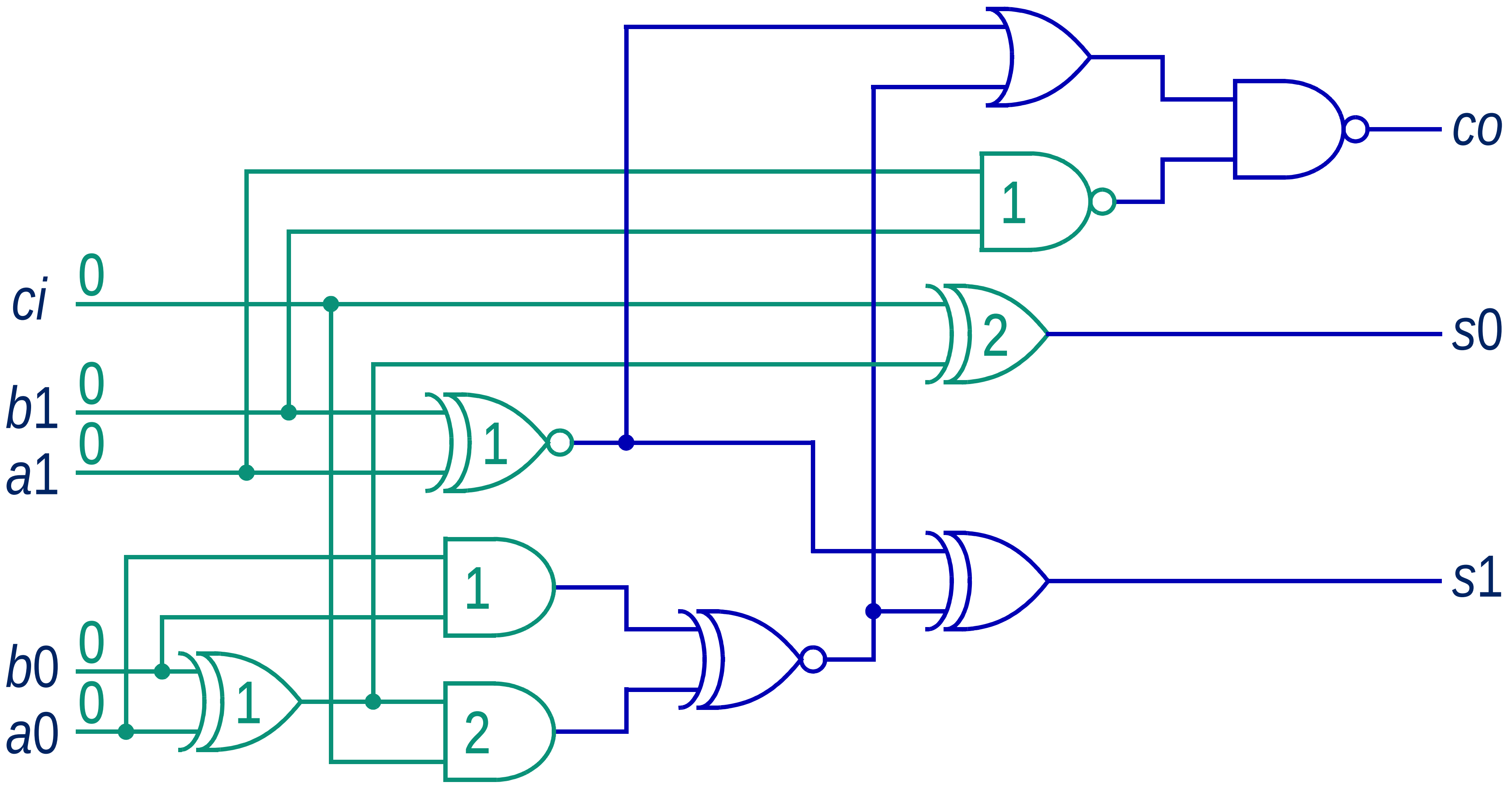
The Critical Path (Unit Delays) 1(10)

The number inside a gate symbol represents the accumulated delay from primary inputs.

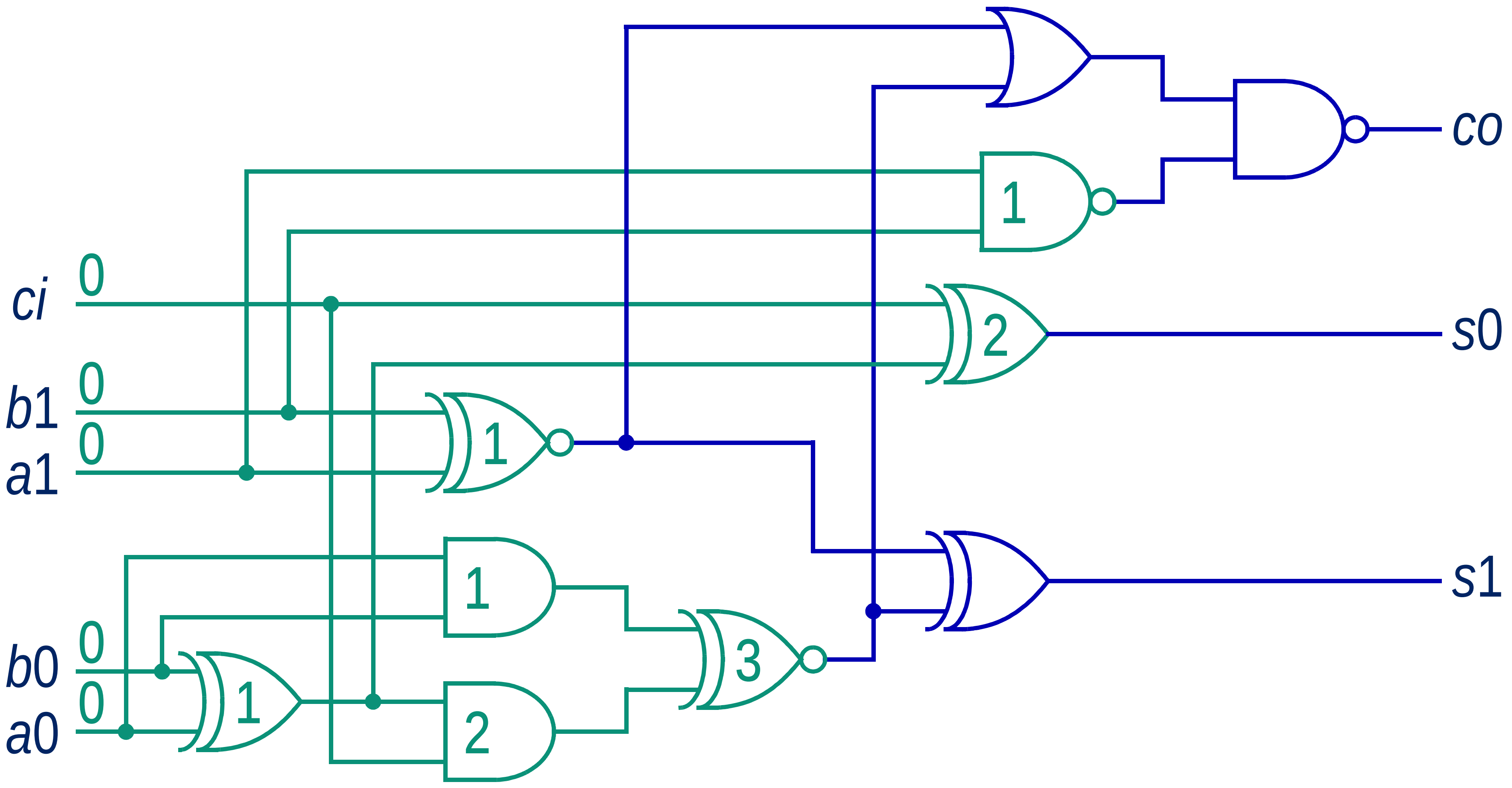


All gates have a unit delay = 1

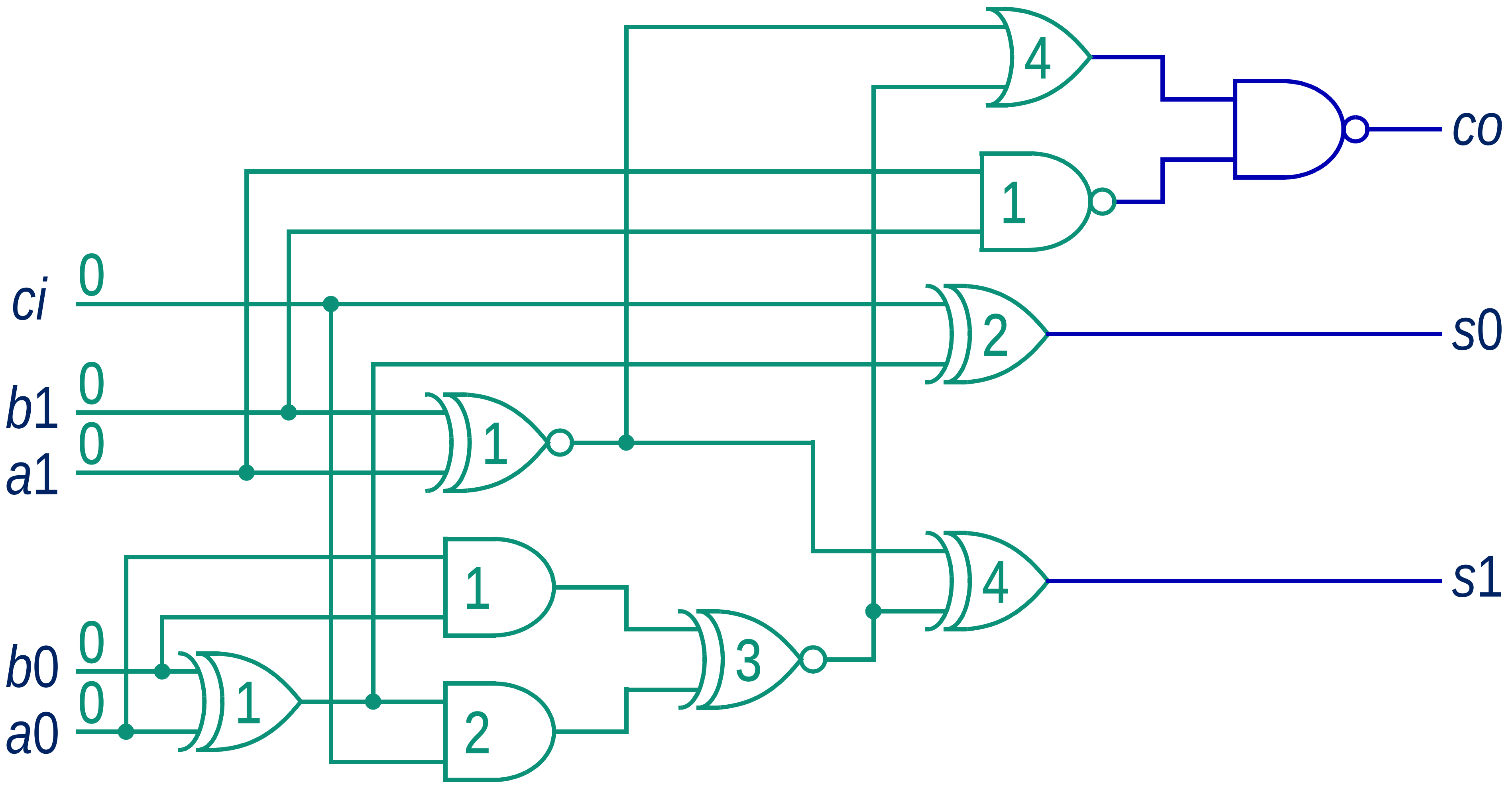
The Critical Path (Unit Delays) 2(10)



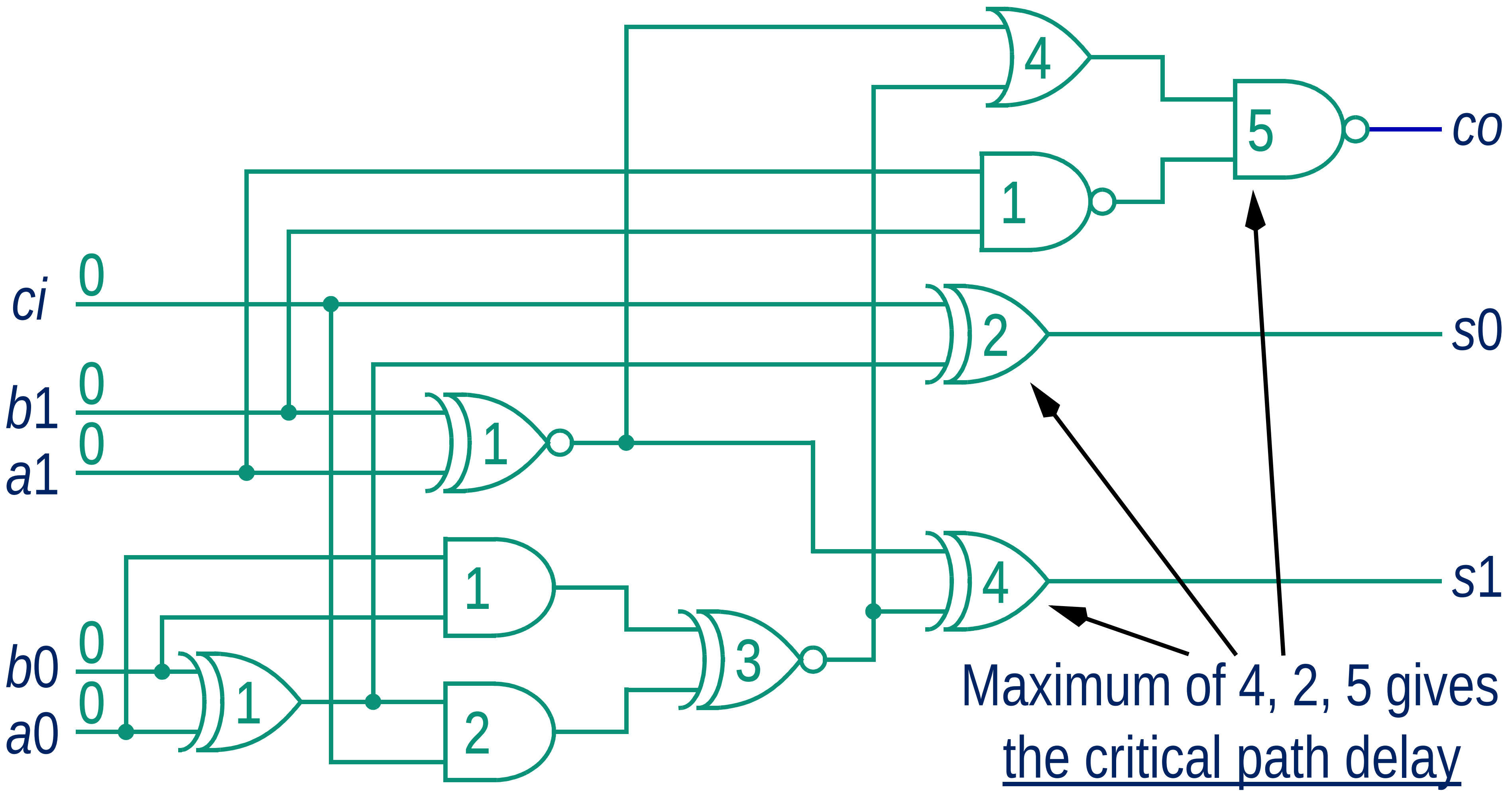
The Critical Path (Unit Delays) 3(10)



The Critical Path (Unit Delays) 4(10)



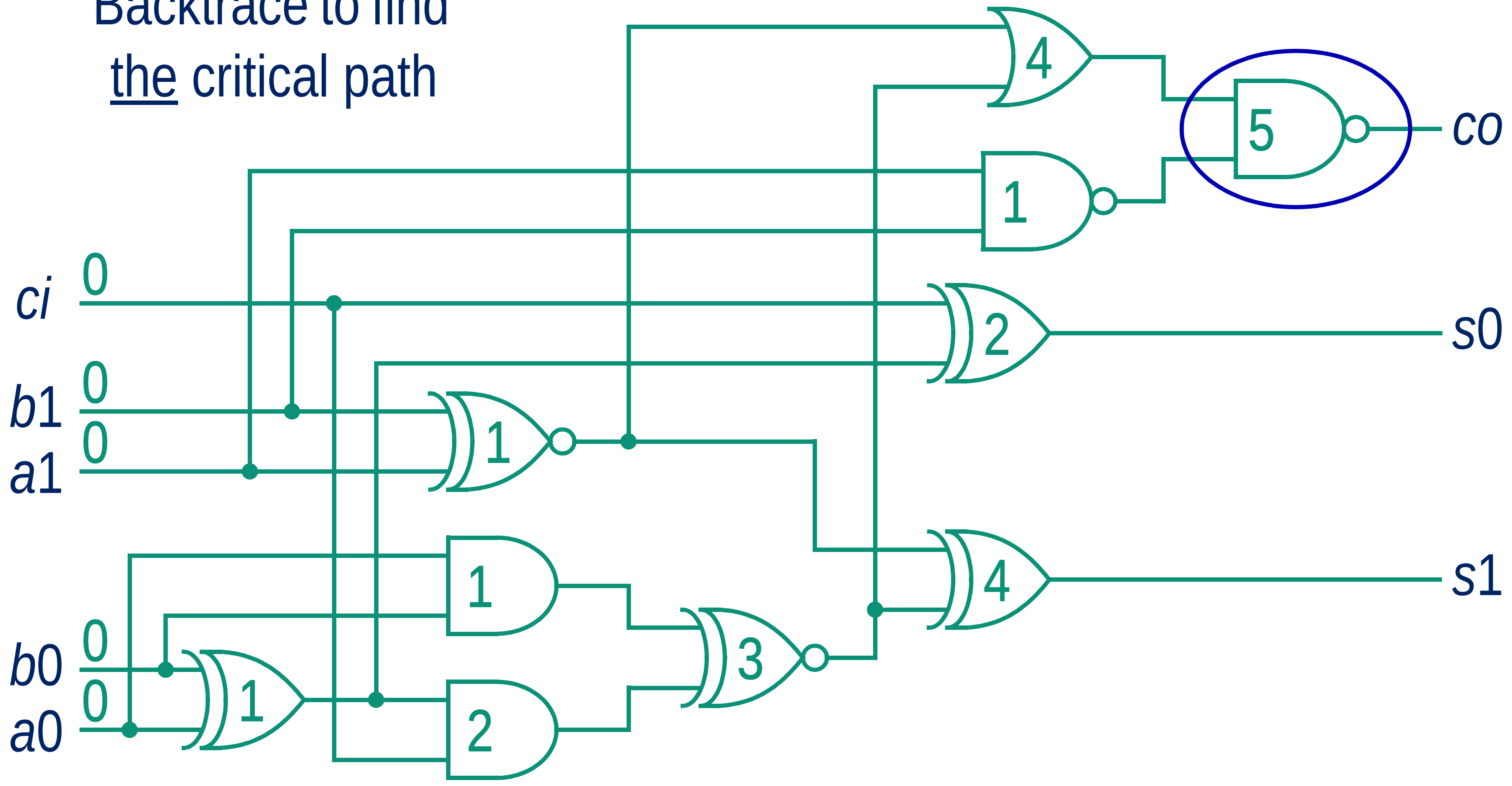
The Critical Path (Unit Delays) 5(10)



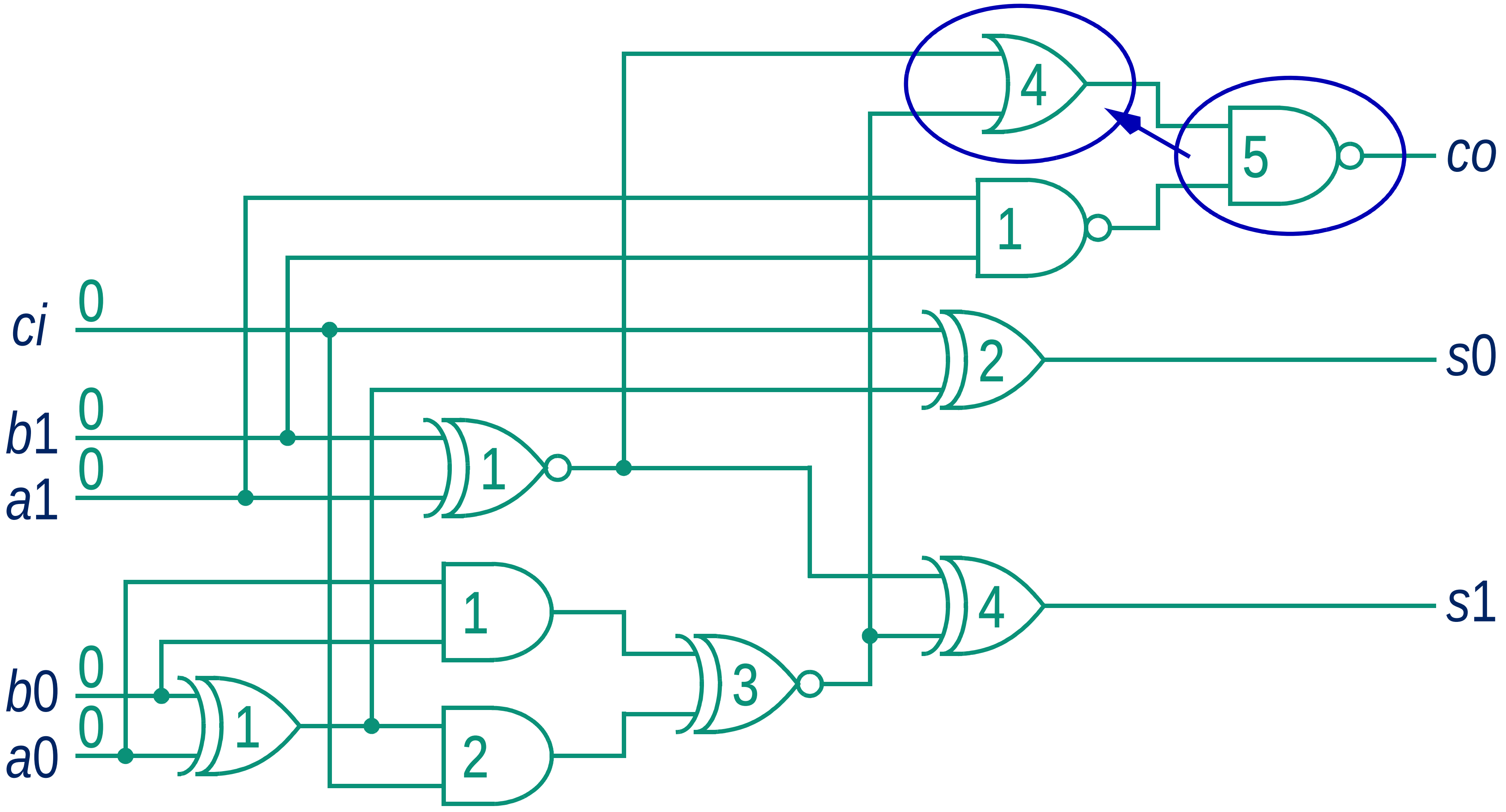
Maximum of 4, 2, 5 gives
<u>the critical path delay</u>

The Critical Path (Unit Delays) 6(10)

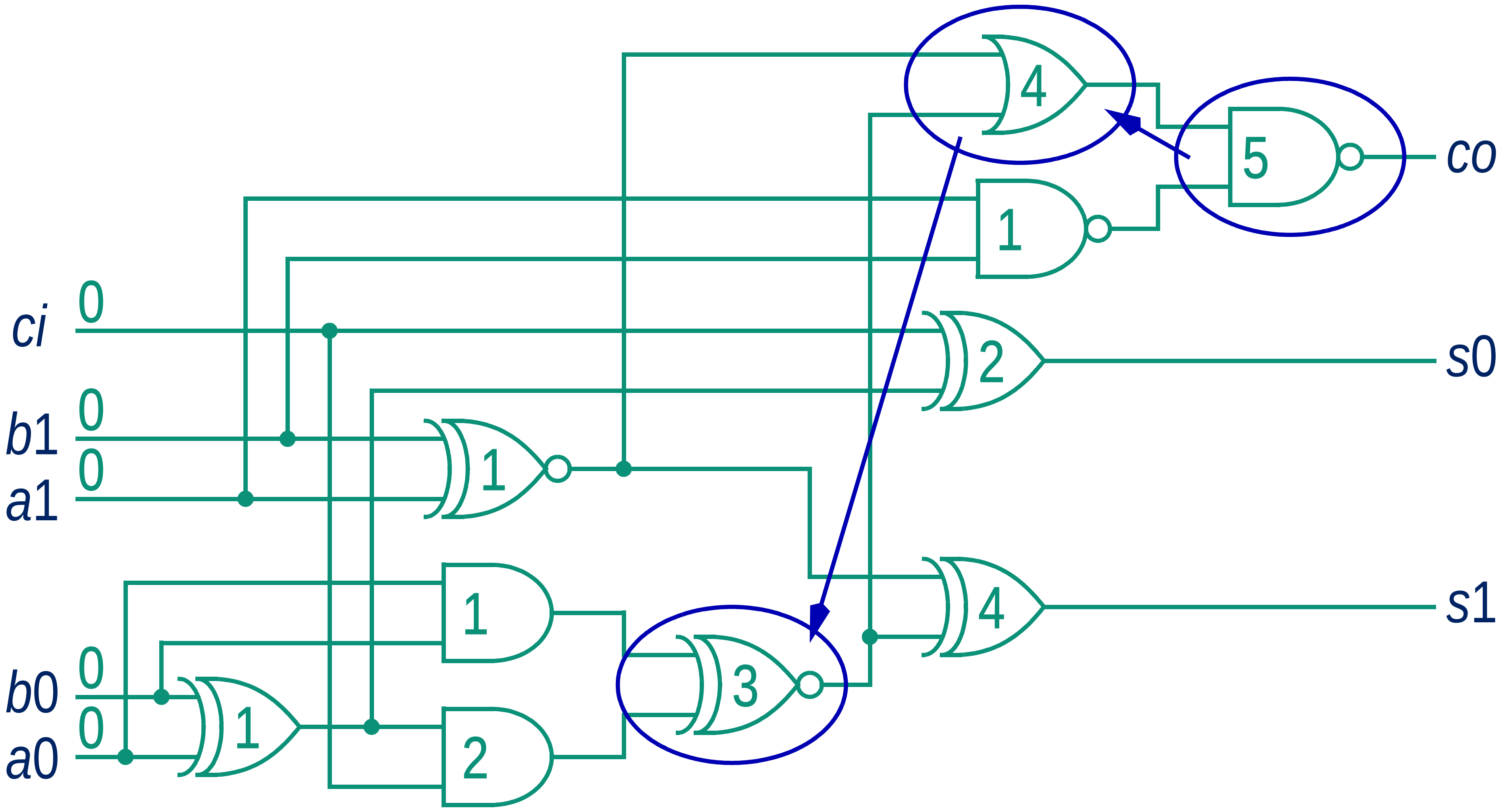
Backtrace to find <u>the</u> critical path



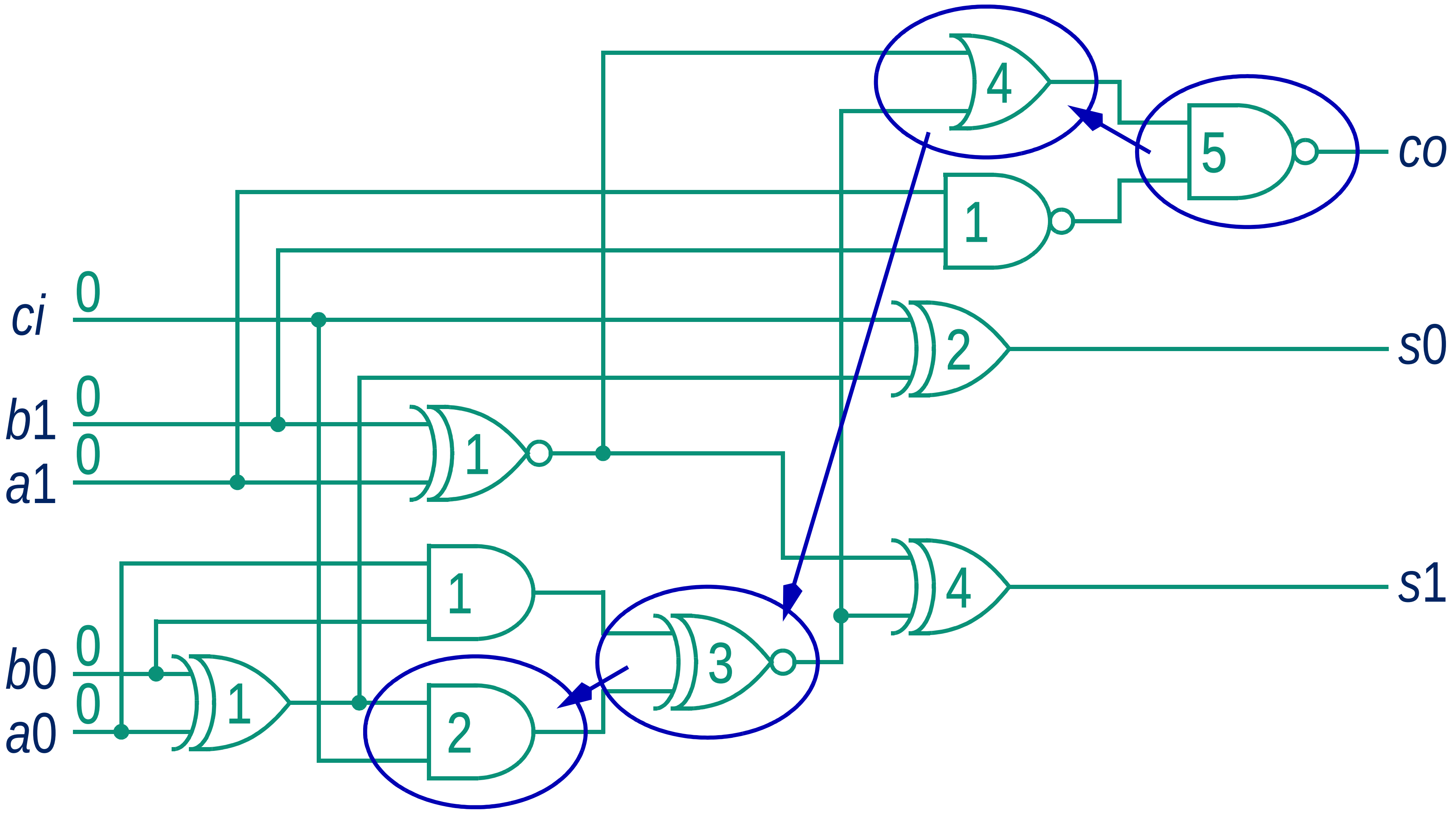
The Critical Path (Unit Delays) 7(10)



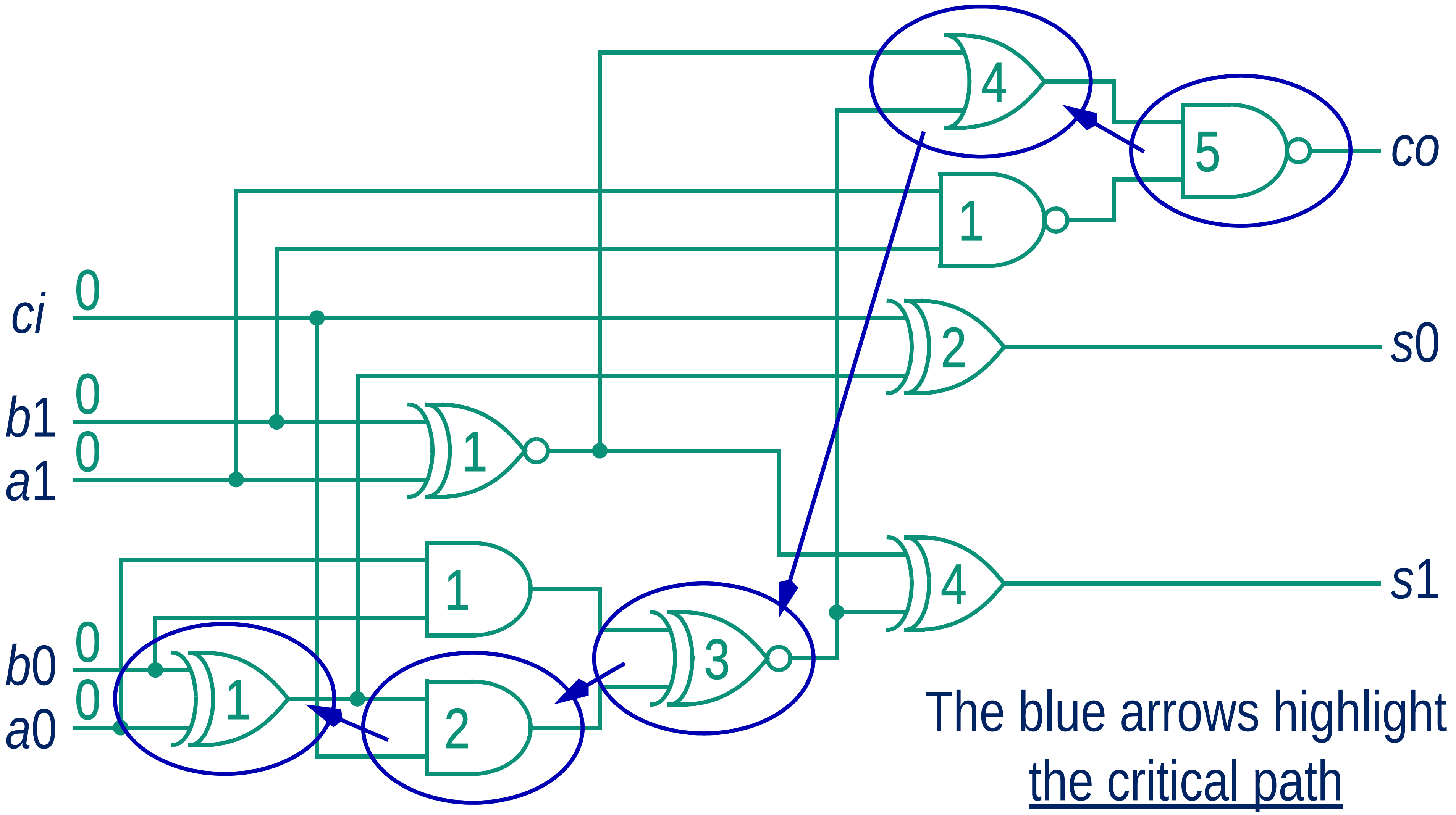
The Critical Path (Unit Delays) 8(10)



The Critical Path (Unit Delays) 9(10)



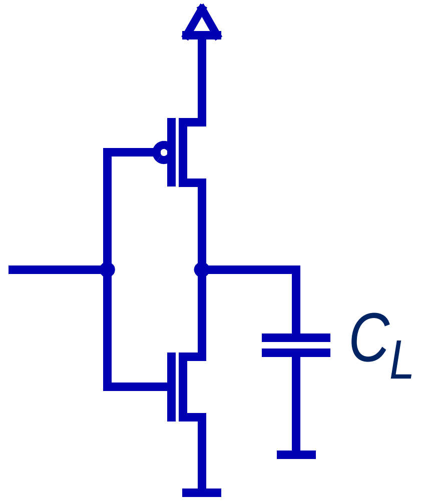
The Critical Path (Unit Delays) 10(10)



The blue arrows highlight the critical path

Back to Basics: Gate Delay

- Simple delay model: $t_d \approx \frac{C_L}{k \cdot V_{DD}}$, where $k = \frac{W}{L} \mu C_{ox}$.



- Empirical models can look like this:

$$t_d = k_0 + k_1 C_L + k_2 C_L^{\,3} + (k_3 + k_4 C_L) t_{rf}$$

- The constants k_0 - k_4 can be obtained by simulation for the different cells in our library.
- Notice: <u>intrinsic</u> delay + <u>input slope</u> dependent delay + <u>load cap</u> dependent delay.

GATES DO NOT HAVE UNIT DELAYS!

First-order 130-nm delay model:

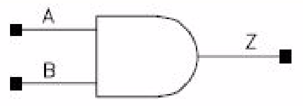
$$k_0 + k_1 C_L + k_2 t_{rf}$$

Cell libraries contain more complex models: See ECSM (Effective Current Source Model) or CCS (Composite Current Source).

AN2HS
AN2HSP
AN2HSX4
AN2HSX6
AN2HSX8

Function: Function = 2 Input AND

Boolean Expression: Z = A●B



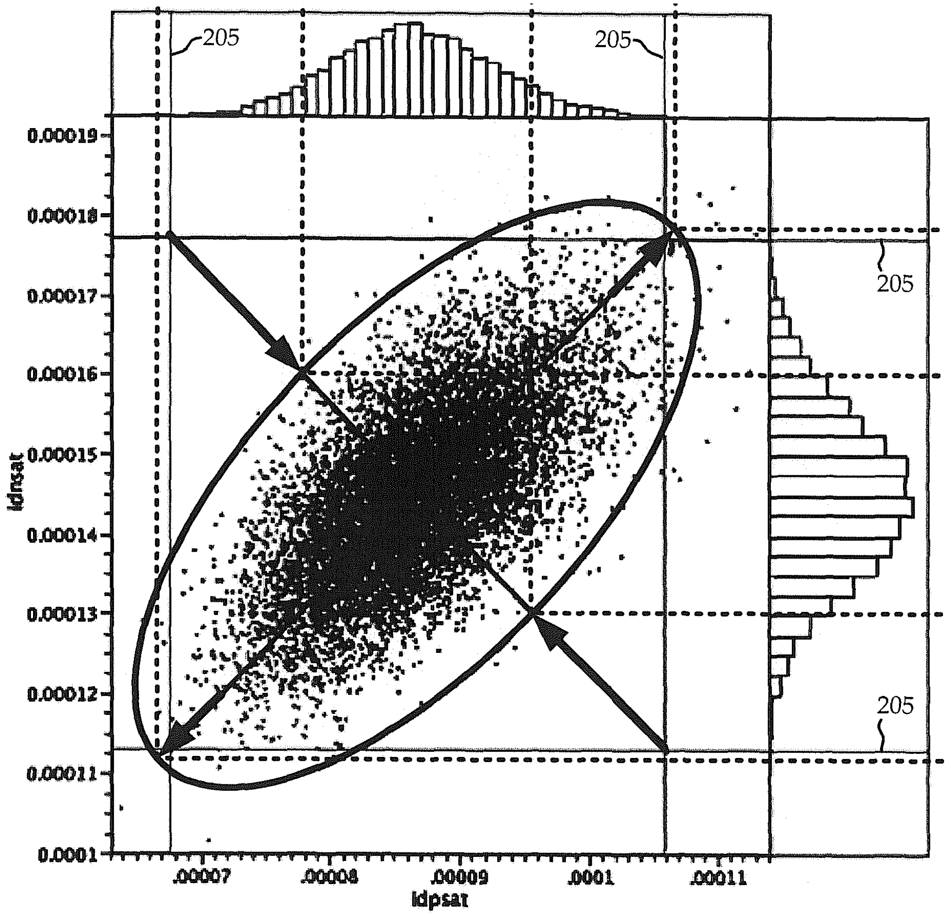
Propagation Delay

nanoSeconds, as a function of C (load in pF) and Tr (input transition time in nS)

Cell	Path	Event	Best 1.32V - 40C	Worst 1.08V 125C	Nominal 1.2V 25C
AN2HS	A-Z	A_Z (fall)	0.045 + 0.277*Tr + 0.995*C	0.106 + 0.270*Tr + 2.065*C	0.067 + 0.273*Tr + 1.360*C
AN2HS	A-Z	A_Z (rise)	0.038 + 0.111*Tr + 1.127*C	0.091 + 0.146*Tr + 2.490*C	0.056 + 0.125*Tr + 1.649*C
AN2HS	B-Z	B_Z (fall)	0.041 + 0.252*Tr + 0.997*C	0.096 + 0.250*Tr + 2.064*C	0.061 + 0.251*Tr + 1.362*C

VARIATIONS CALL FOR CORNER-BASED DESIGN

- Worst case timing necessary to qualify an implementation.



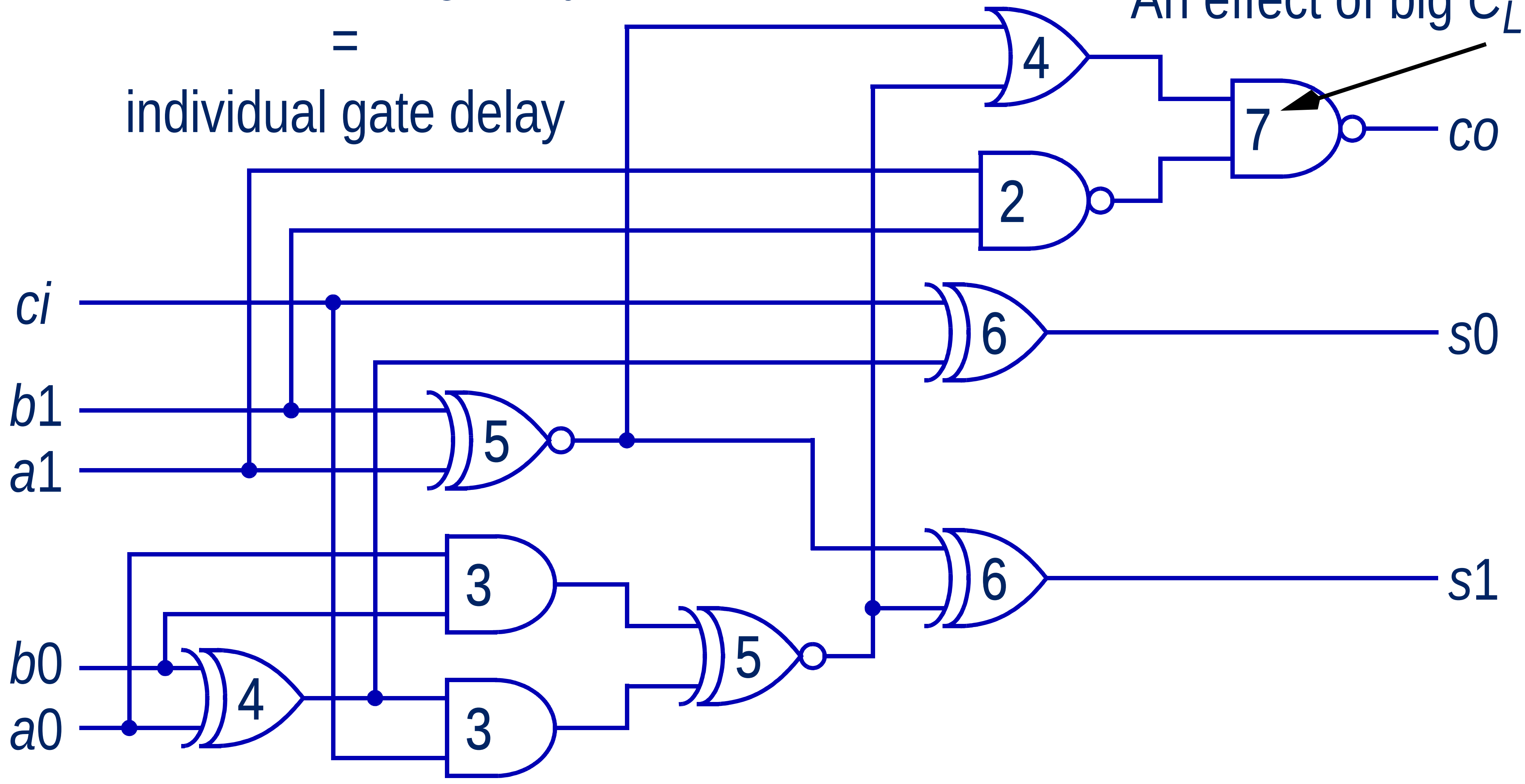
Source: Deriving effective corners for complex correlations, US patent 8819605, 2013

- WC: Slow NMOS + Slow PMOS + High T + Low VDD [Lecture 6].

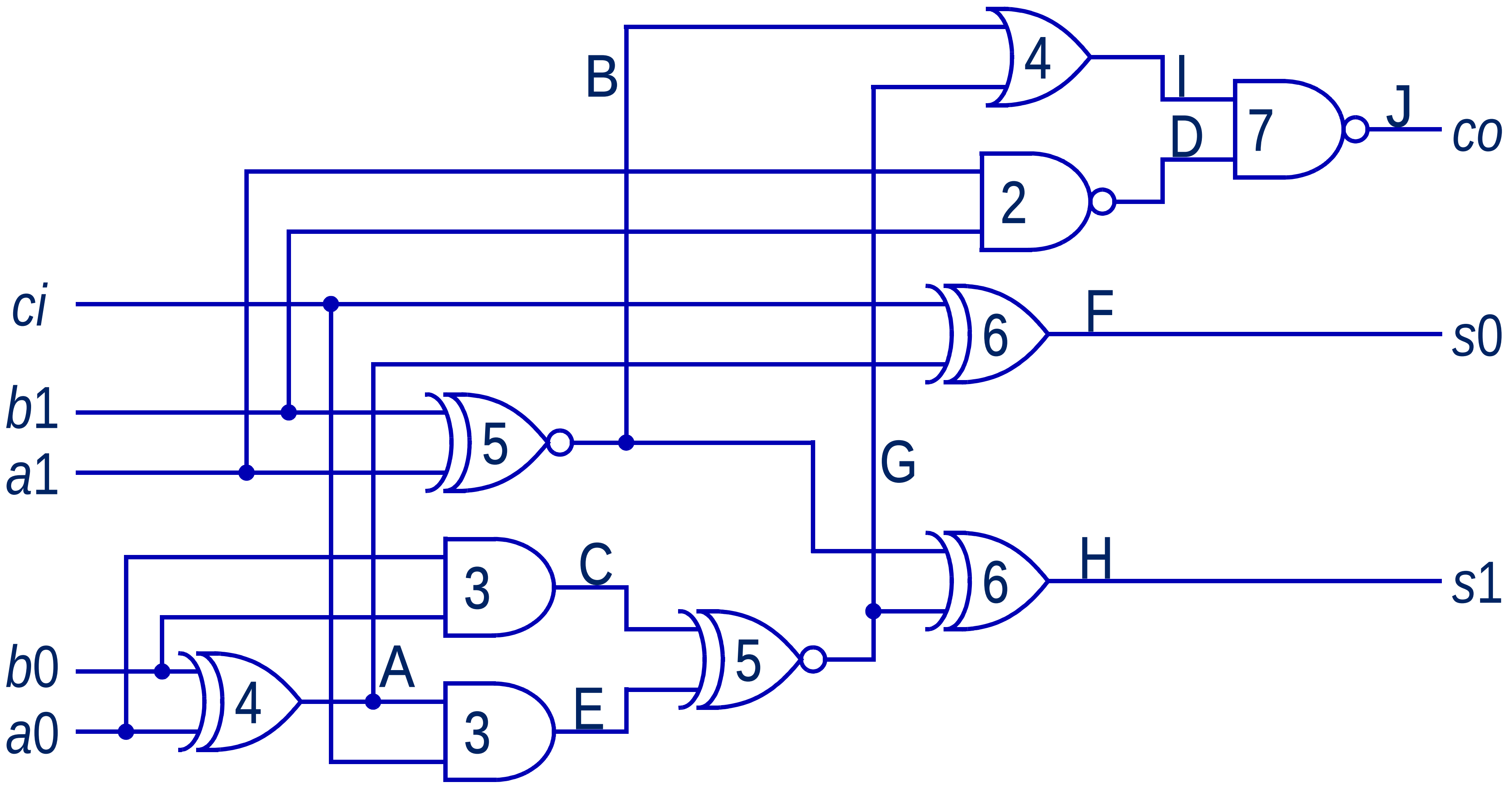
The Critical Path ("Realistic" Circuit) 1(9)

The number inside a gate symbol
=
individual gate delay

An effect of big C_L !



The Critical Path ("Realistic" Circuit) 2(9)

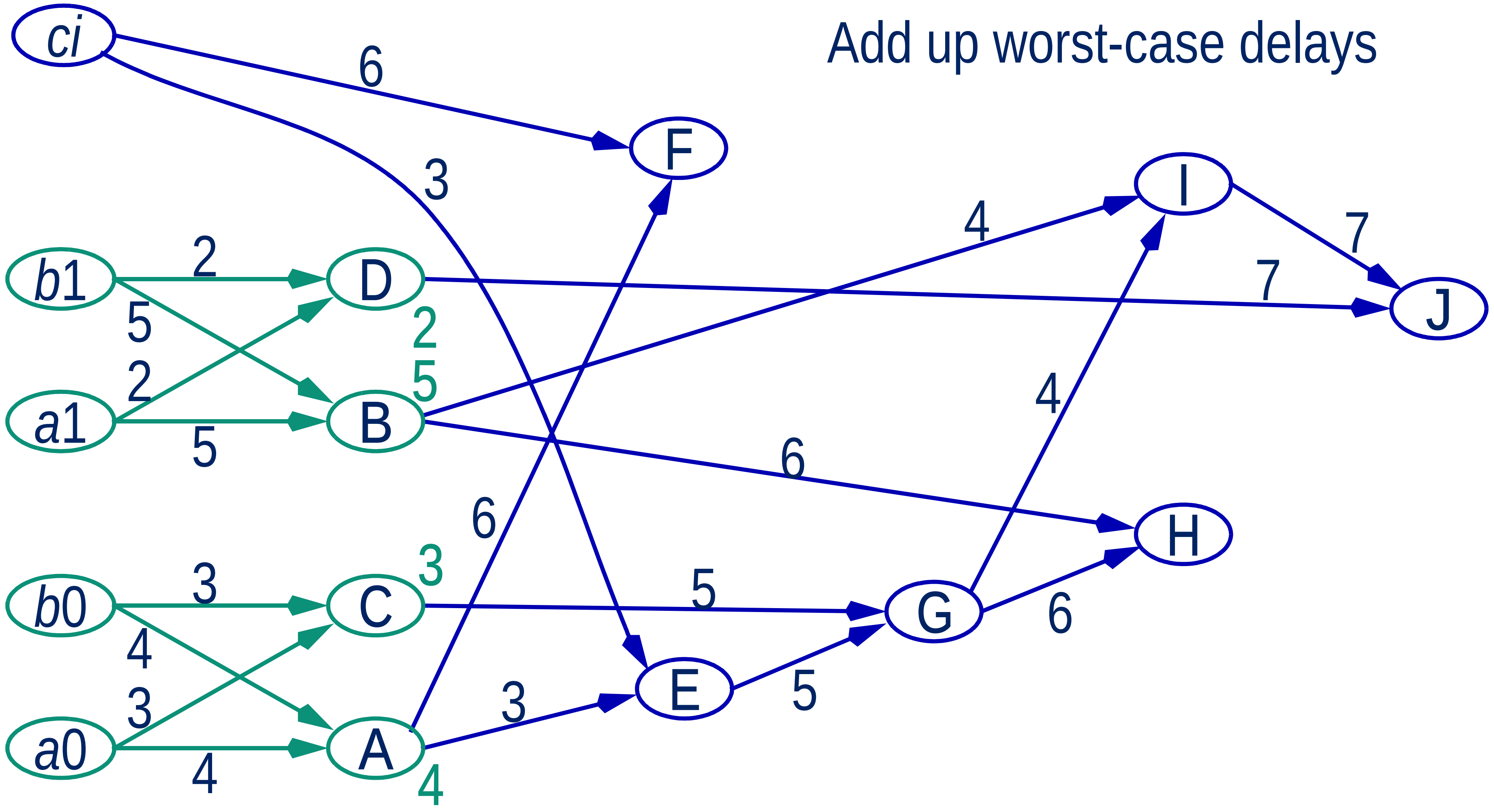


The Critical Path (“Realistic” Circuit) 3(9)

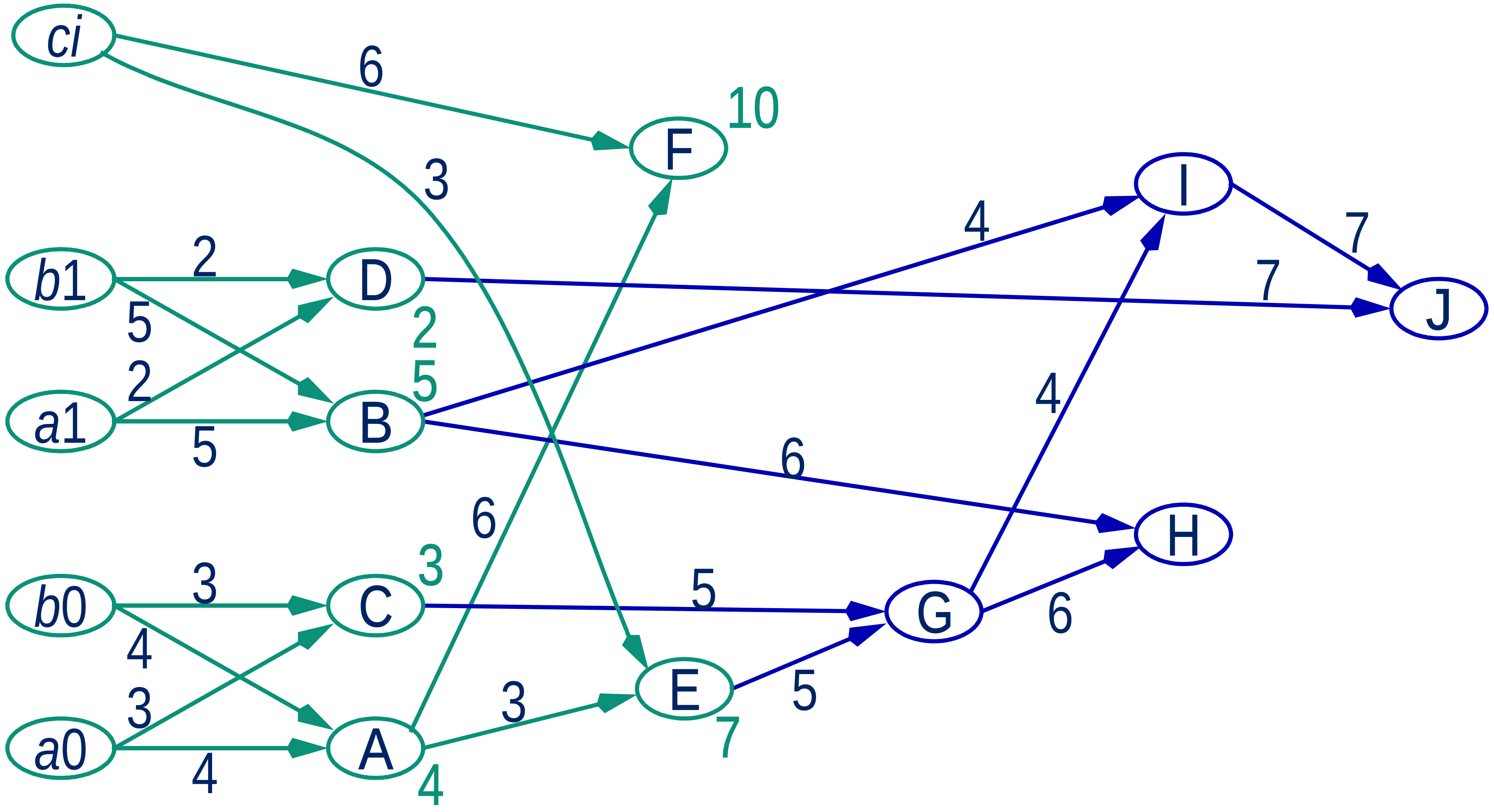
$$\begin{tikzcd}
ci \arrow[rr, "6"] \arrow[ddddrr, "3"] & & F & & I \arrow[r, "7"] & J \\
b1 \arrow[r, "2"] \arrow[dr, "5"] & D \arrow[urrrr, "7"] & & & & \\
a1 \arrow[ur, "2"] \arrow[r, "5"] & B \arrow[uurrr, "4"] \arrow[rrr, "6"] & & & H & \\
b0 \arrow[r, "3"] \arrow[dr, "4"] & C \arrow[rr, "5"] & & G \arrow[uuur, "4"] \arrow[ur, "6"] & & \\
a0 \arrow[ur, "3"] \arrow[r, "4"] & A \arrow[uuuur, "6"] \arrow[r, "3"] & E \arrow[ur, "5"] & & &
\end{tikzcd}$$

The Critical Path ("Realistic" Circuit) 4(9)

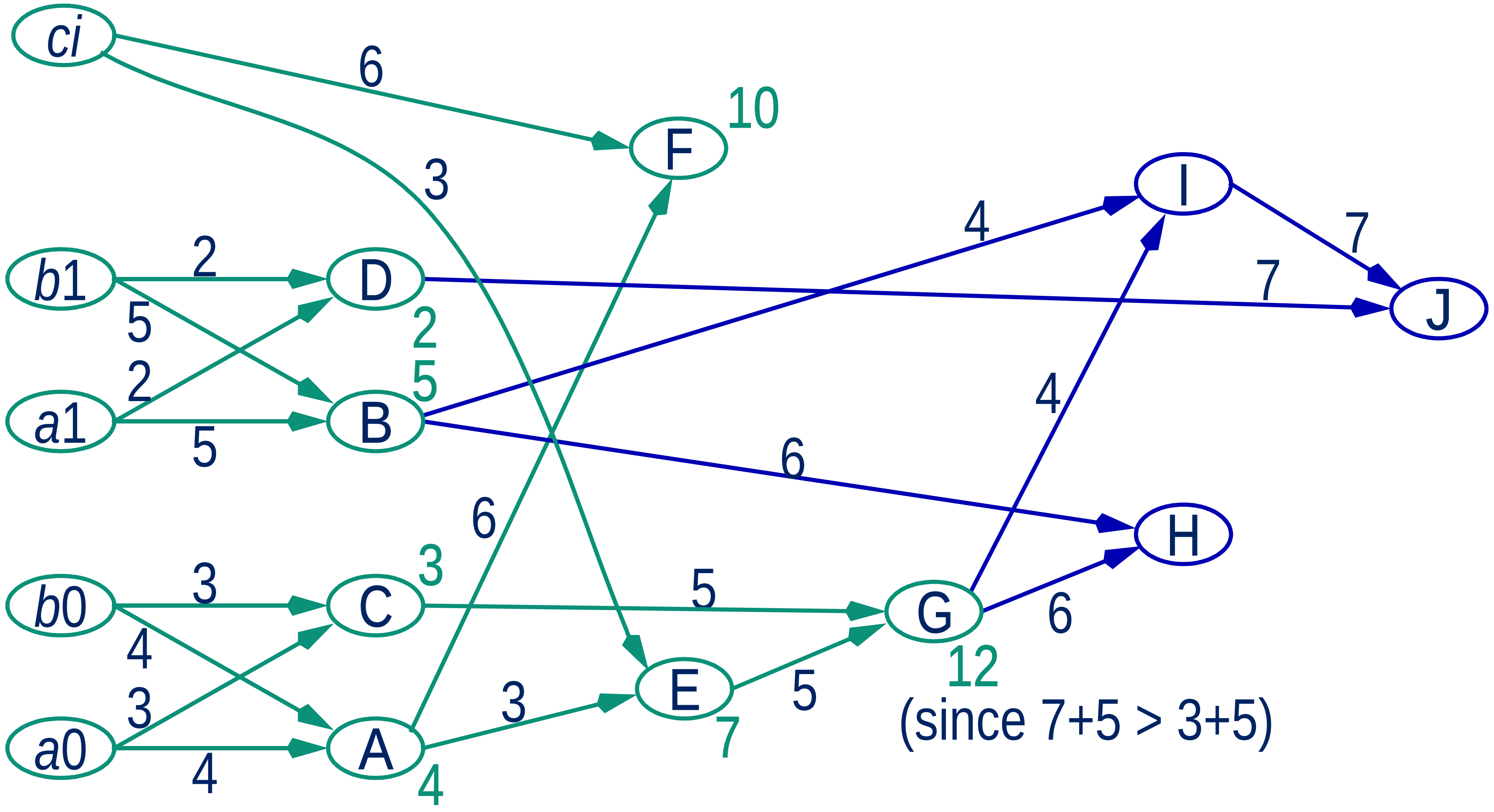
Add up worst-case delays



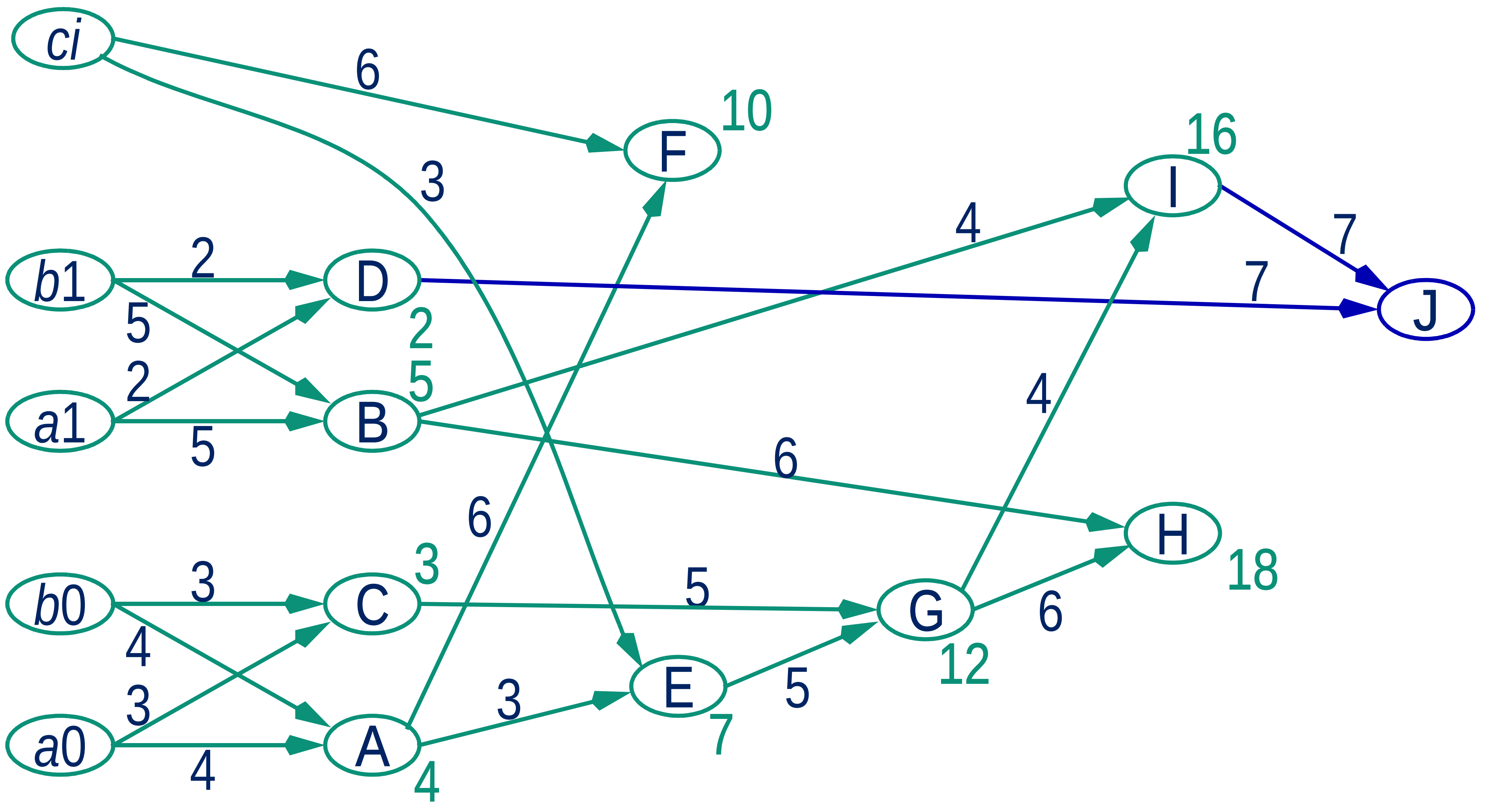
The Critical Path ("Realistic" Circuit) 5(9)



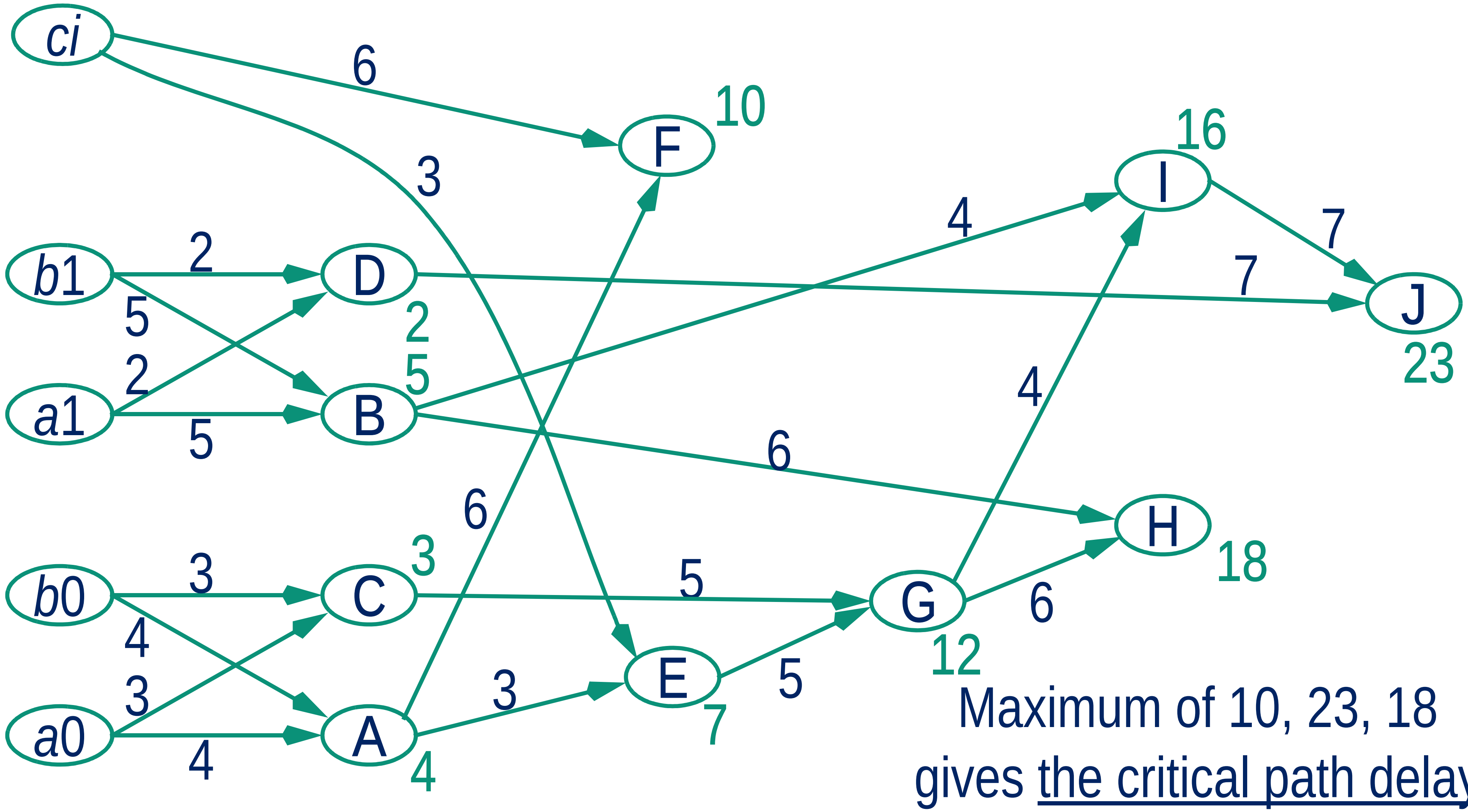
The Critical Path ("Realistic" Circuit) 6(9)



The Critical Path ("Realistic" Circuit) 7(9)

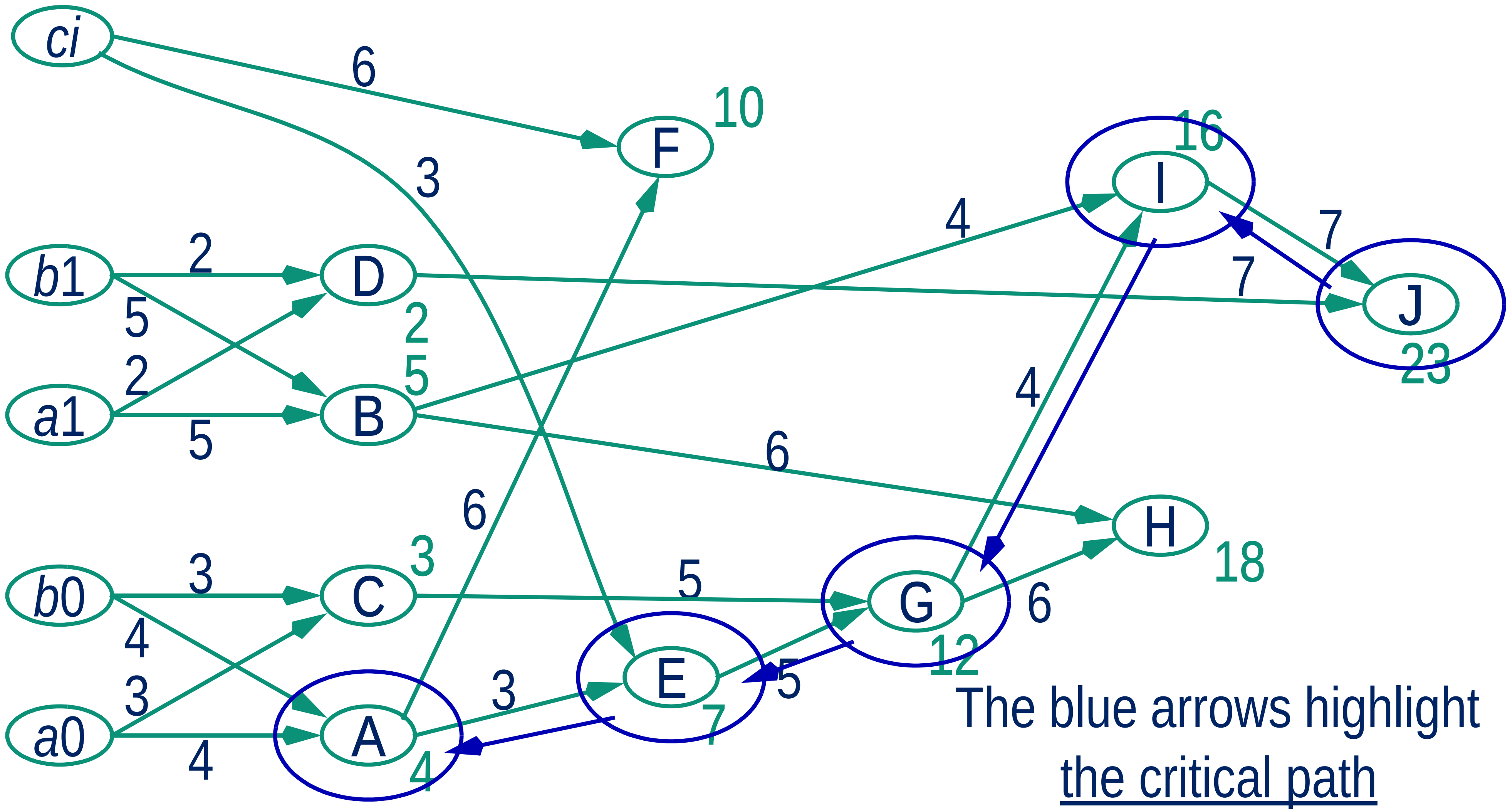


The Critical Path ("Realistic" Circuit) 8(9)



Maximum of 10, 23, 18 gives the critical path delay

The Critical Path ("Realistic" Circuit) 9(9)

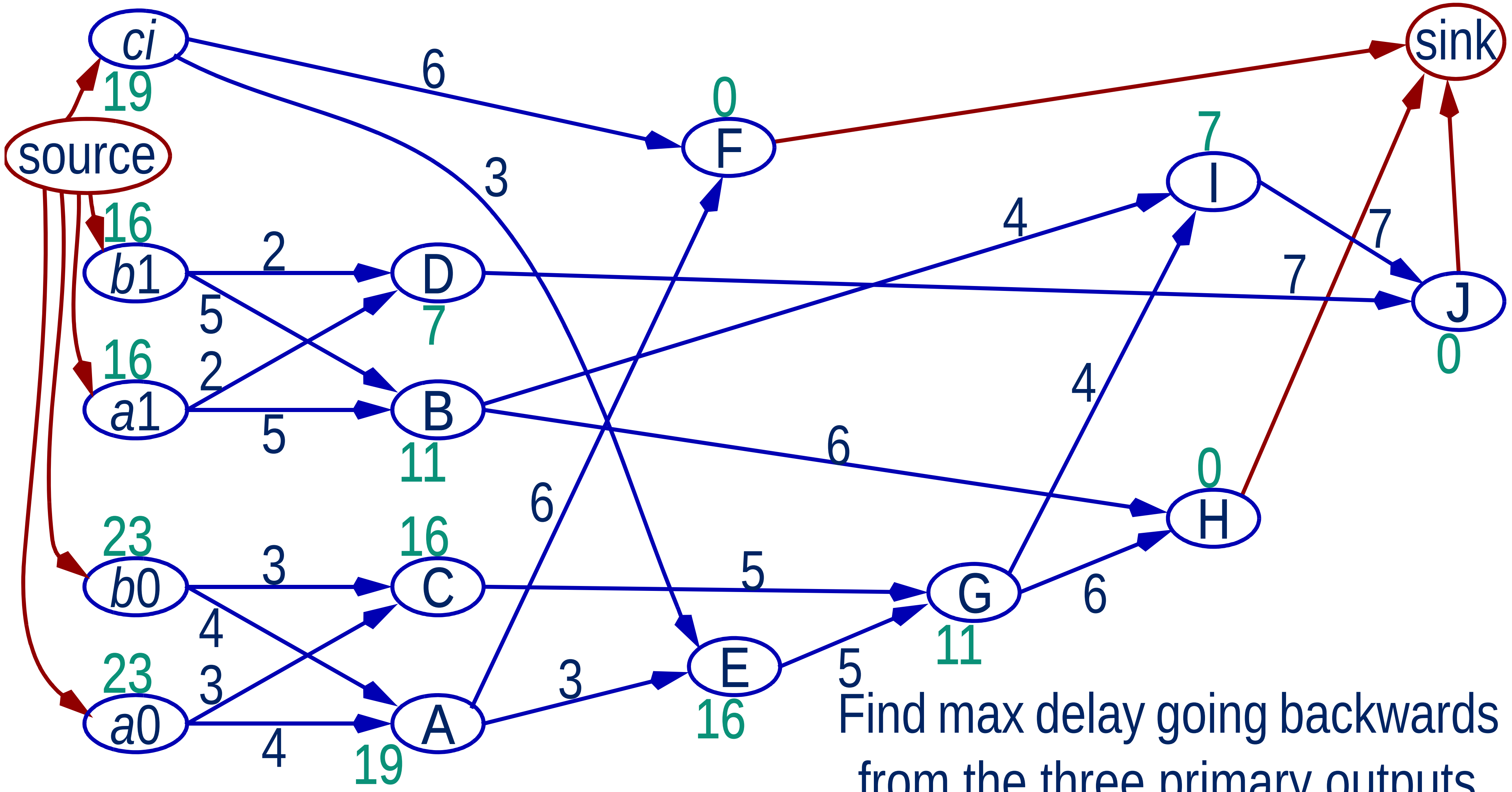


The blue arrows highlight <u>the critical path</u>

One or Several Long Paths?

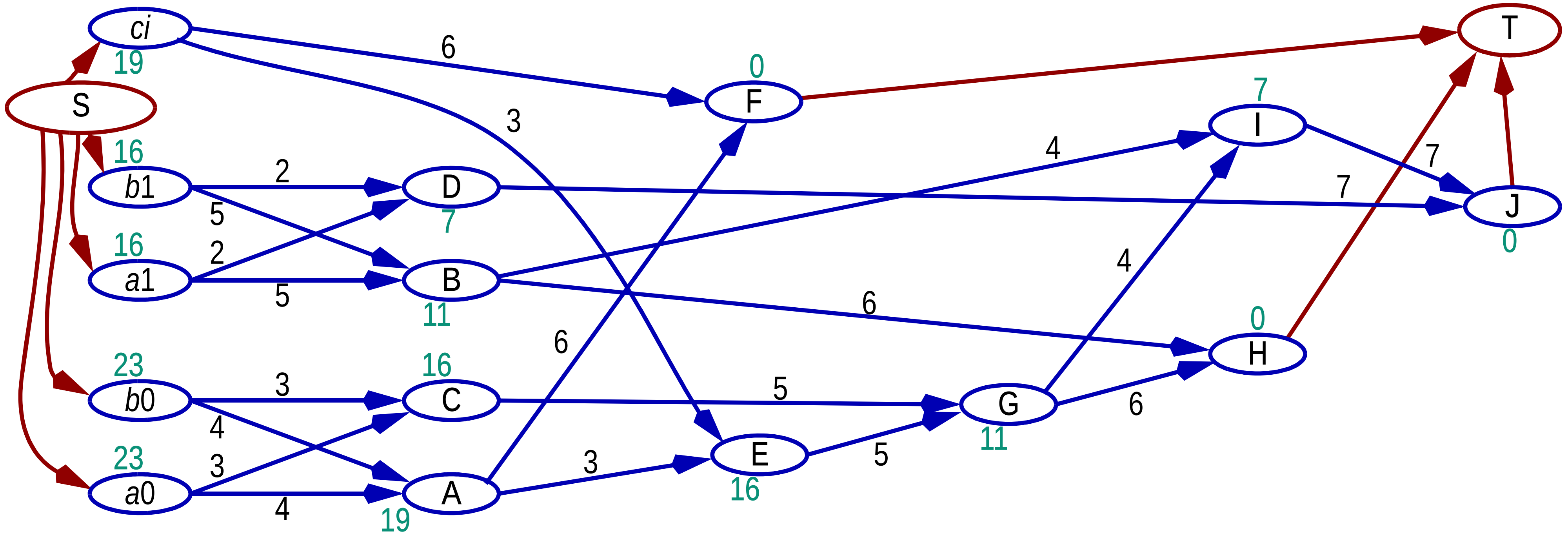
- The simple method shown gives ...
  - the critical path delay.
  - the critical path.
- In a real design situation, information on only one critical path is not useful.
  - Actually, well-designed circuits should have many “almost” critical paths.
- To find a list of the *N* most critical paths (ordered according to delay) we need to employ a procedure called path enumeration.

Path Enumeration 1(9)



Find max delay going backwards from the three primary outputs.

PATH ENUMERATION 2(9)

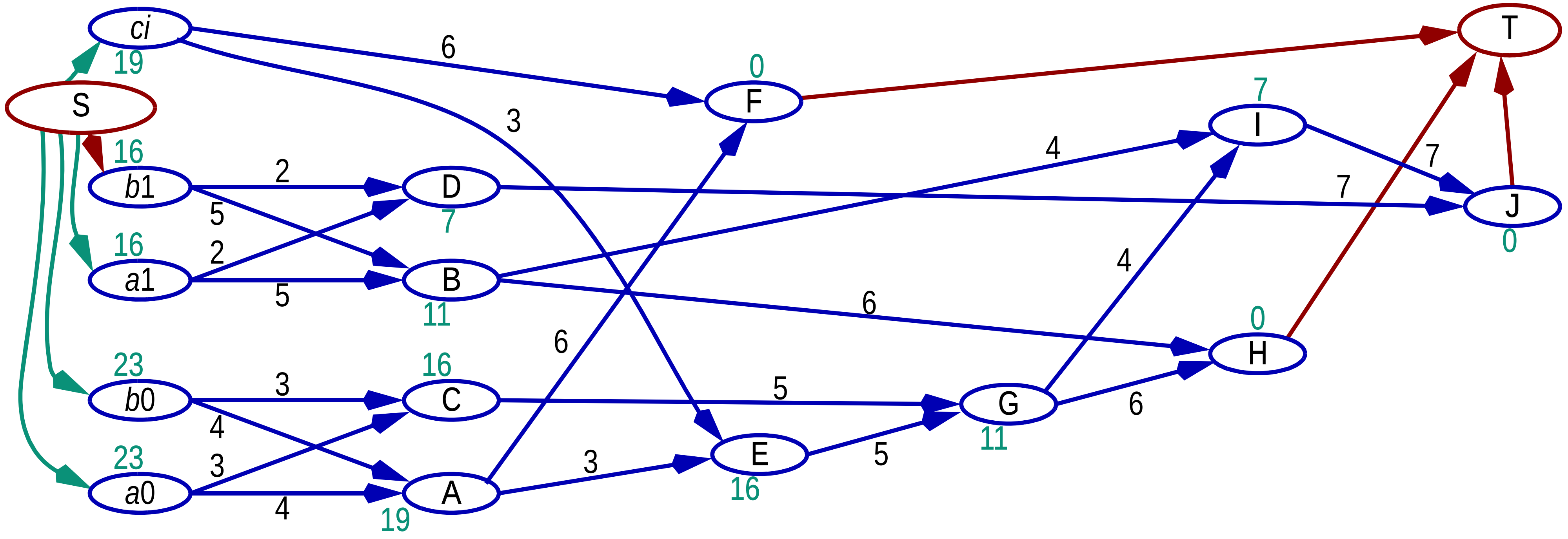


Start from source and keep the *N* longest paths (= paths with largest max_path_delay). Define:

max_path_delay = delay<S, *vertex*> + max_delay<*vertex*, T>

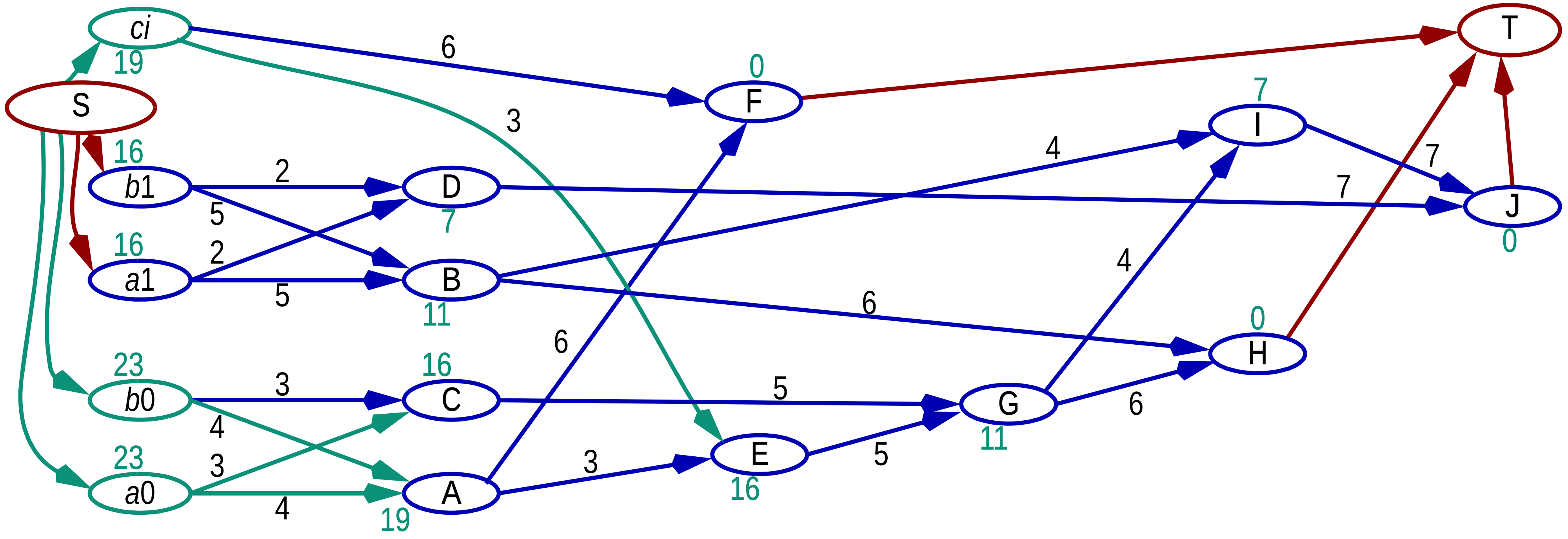
Example: Find the 4 longest paths!

PATH ENUMERATION 3(9)



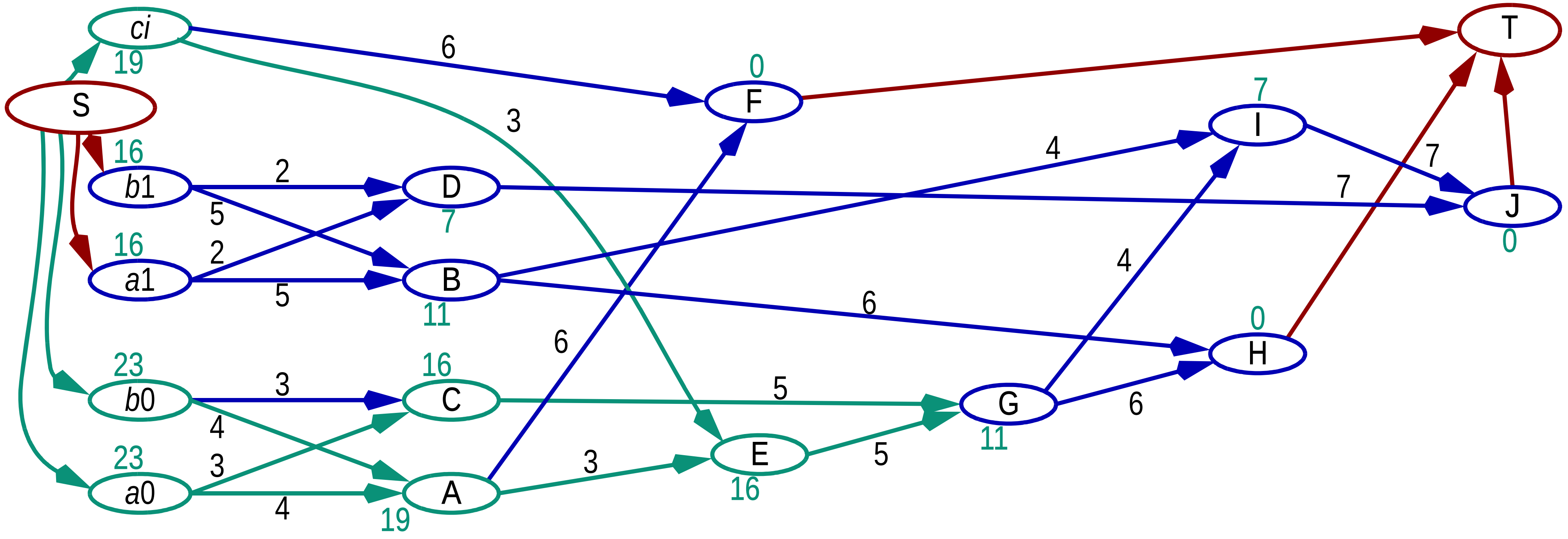
path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0	0	23	23
*b*0	0	23	23
ci	0	19	19
*a*1	0	16	16

Path Enumeration 4(9)



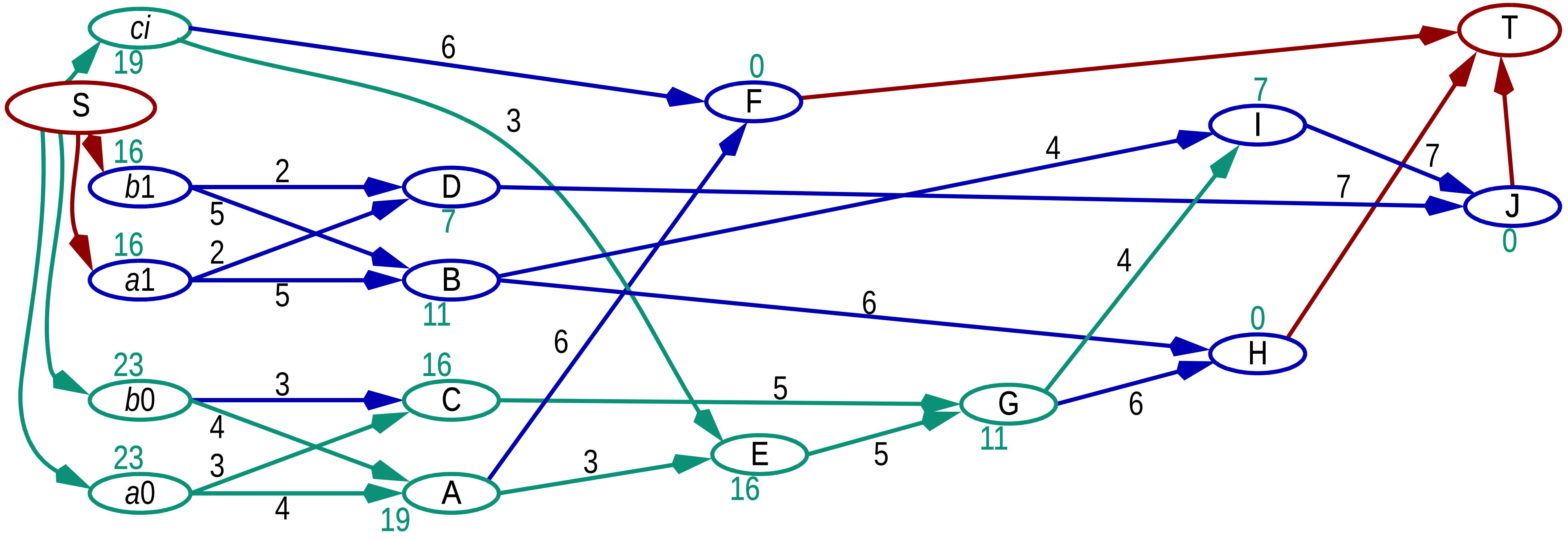
path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A	4	19	23
*b*0, A	4	19	23
ci, E	3	16	19
*a*0, C	3	16	19

PATH ENUMERATION 5(9)



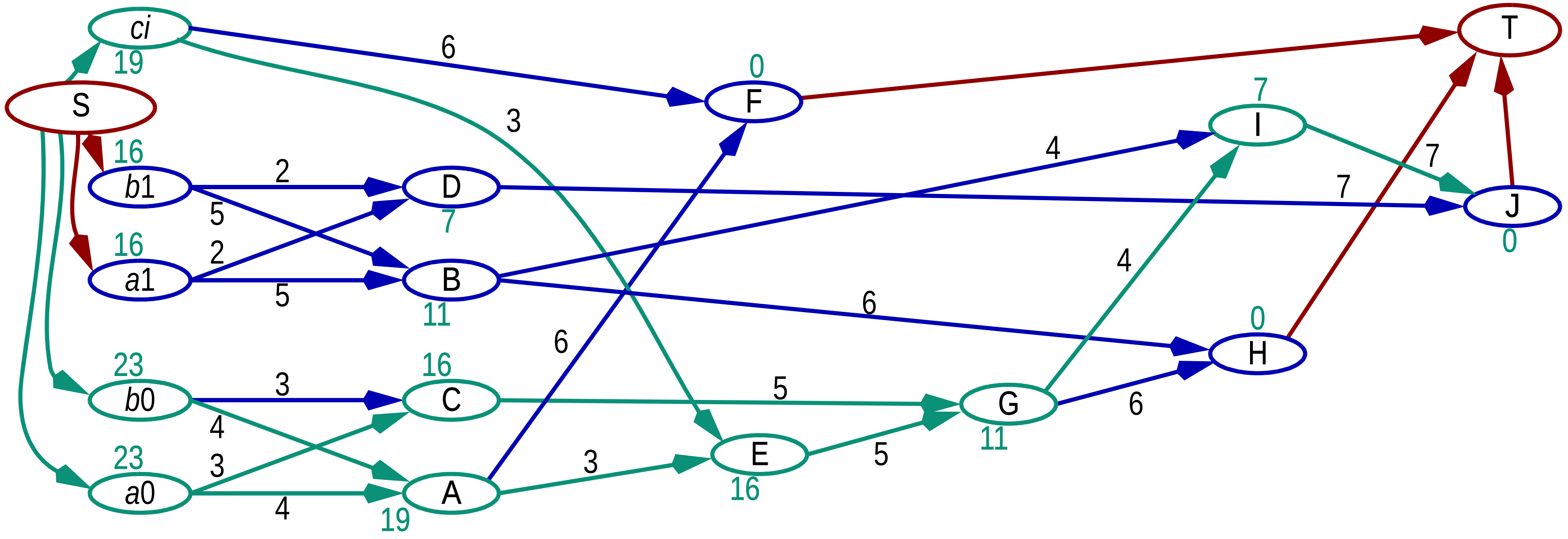
path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A, E	7	16	23
*b*0, A, E	7	16	23
ci, E, G	8	11	19
*a*0, C, G	8	11	19

Path Enumeration 6(9)



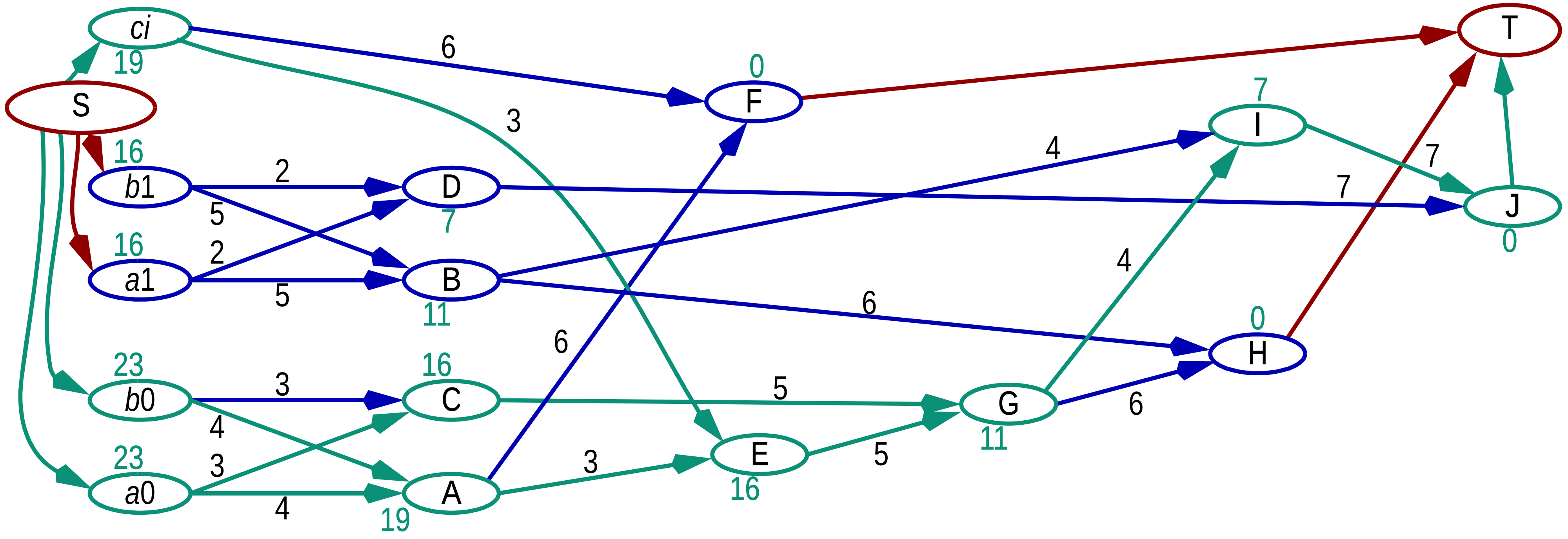
path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A, E, G	12	11	23
*b*0, A, E, G	12	11	23
ci, E, G, I	12	7	19
*a*0, C, G, I	12	7	19

Path Enumeration 7(9)



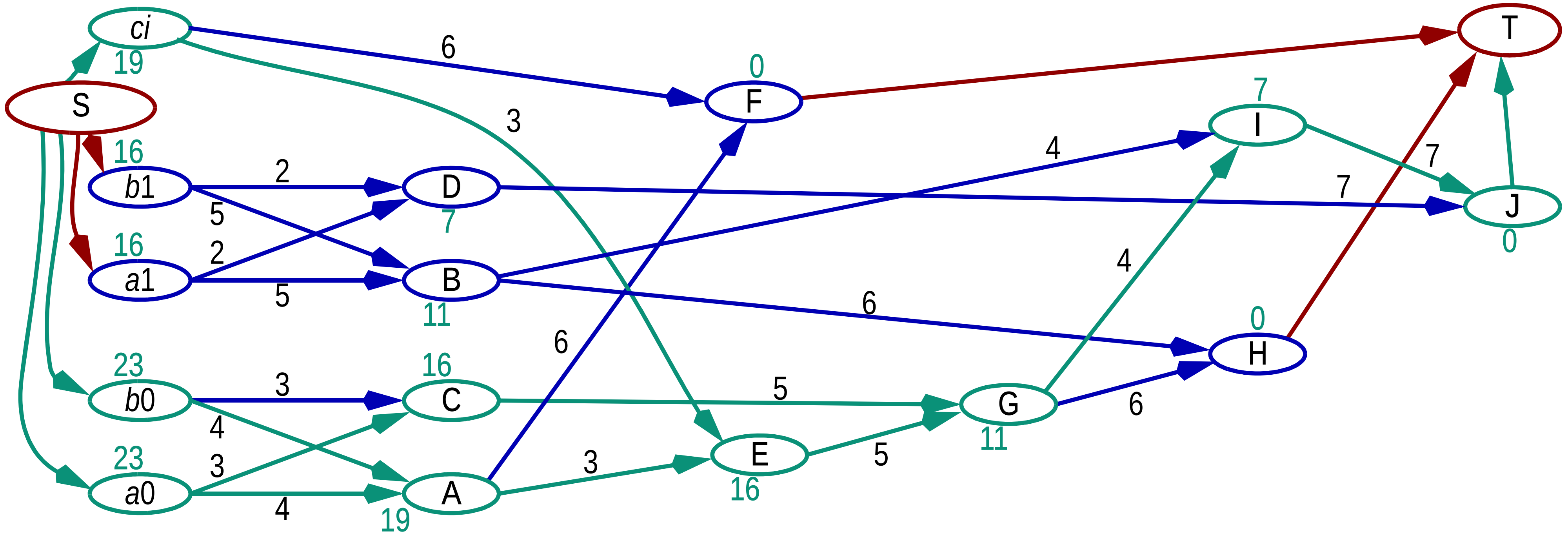
path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A, E, G, I	16	7	23
*b*0, A, E, G, I	16	7	23
ci, E, G, I, J	19	0	19
*a*0, C, G, I, J	19	0	19

Path Enumeration 8(9)



path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A, E, G, I, J	23	0	23
*b*0, A, E, G, I, J	23	0	23
ci, E, G, I, J, T	19	0	19
*a*0, C, G, I, J, T	19	0	19

Path Enumeration 9(9)

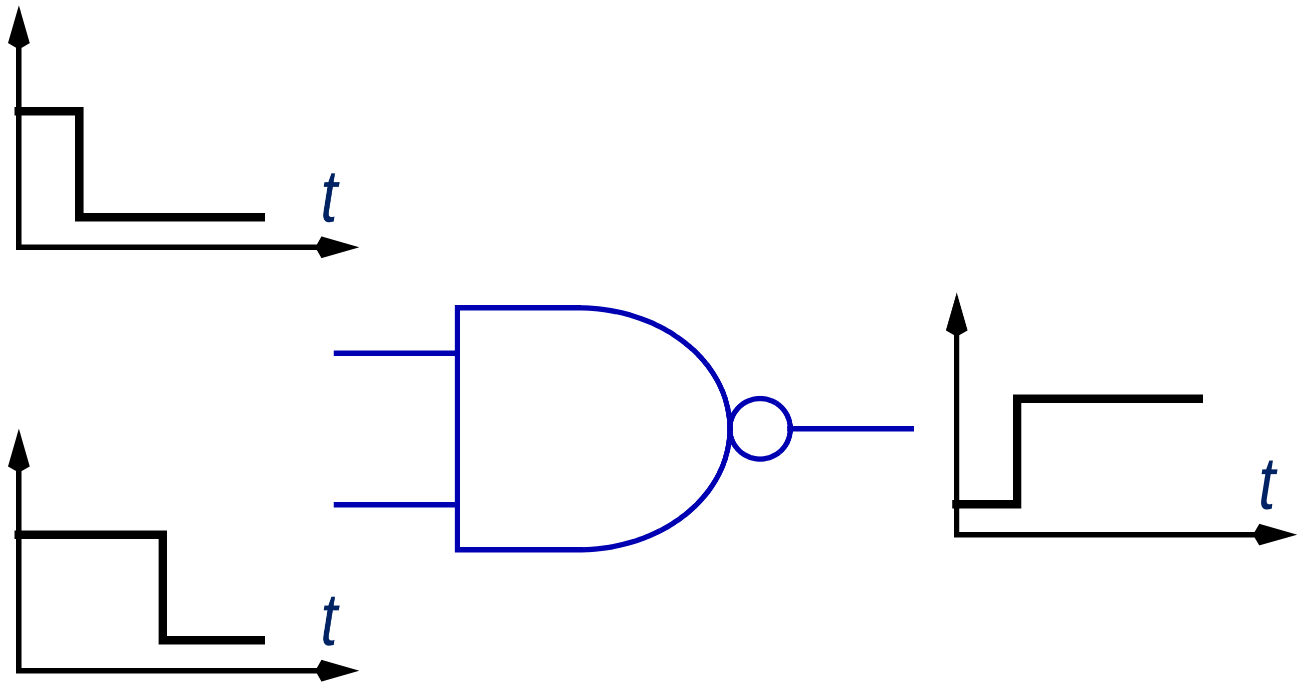


path = <S, ..., *v*>	delay <S, ..., *v*>	max_delay <*v*, T>	max_path_delay
*a*0, A, E, G, I, J, T	23	0	23
*b*0, A, E, G, I, J, T	23	0	23
ci, E, G, I, J, T	19	0	19
*a*0, C, G, I, J, T	19	0	19

Weaknesses of Timing Analysis

- Some input changes won’t make output values change.
  - False paths can give rise to too pessimistic delay values.
- If the circuit is not working properly, the reported paths might be erroneous.
  - A circuit must be extensively (functionally) verified before subjected to timing analysis.
- Hard to capture dynamic events.
  - For example, crosstalk effects between wires, in deep submicron process technologies.

A Simple Example of False Path



The bottom path is false,
since it does not affect the output.

Timing Analysis Output

```
Pin                 Type        Fanout Load   Slew Delay   Arrival
                                       (fF)   (ps)  (ps)    (ps)
---------------------------------------------------------------
ALUOp_reg[2]/CP                                 0             0 R
ALUOp_reg[2]/Q      FD2QHSP          6  30.8   83  +106     106 F
g1410/Z             NR2HSP          33 185.9  859  +300     407 R
g1409/Z             ND2HSP          33 206.2  683  +448     855 F
g1407/Z             F_ENHSX05        1   8.2  265  +207    1062 F
U1/OpB[0]
  FA0/B
    g27/CO          F_FA1HS          1  10.7   76  +175    1237 F
...
```

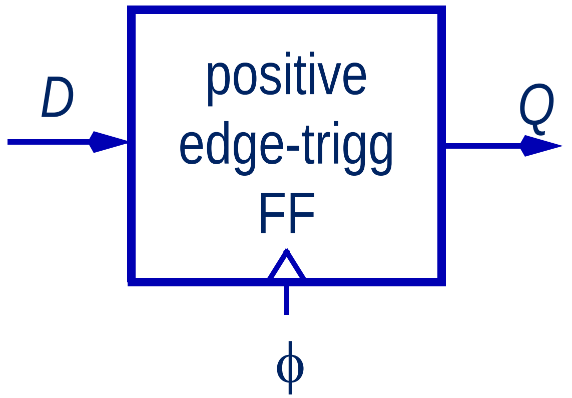
Timing Terminology 1(2)

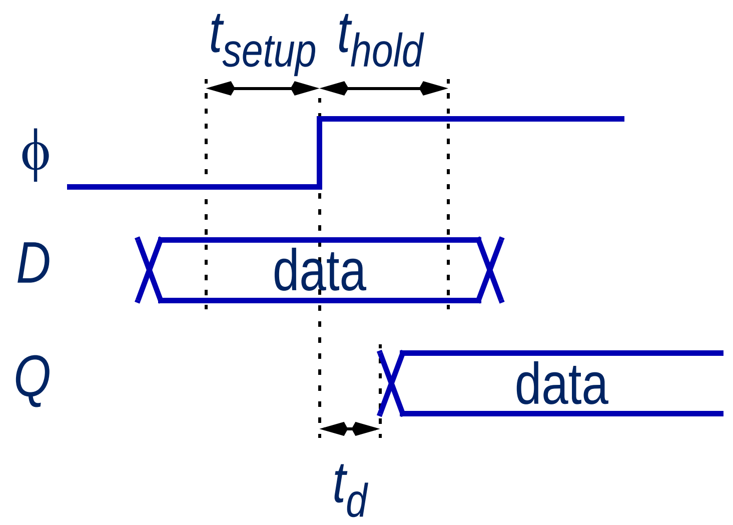
- Slack:

```
...
g4310/Z             ND2AHSX4       1  6.5   33    +24     2272 F
Outs_reg[31]/D      FDD2QHSP                       +0     2272
Outs_reg[31]/CP     setup                    0    +17     2289 R
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
(clock main_clk)    capture                               1250 R
---------------------------------------------------------------
Timing slack :   -1039ps (TIMING VIOLATION)
```

- When experiencing negative slack, like above, you need to redesign your RTL or resynthesize using higher effort.

Timing Terminology 2(2)





- Setup time = the time that *D* must be stable before the clock edge.
- Hold time = the time that *D* must be stable after the clock edge.
- Recovery = the time *FF reset* must be stable before a clock edge.
- Removal = the time *FF reset* must be stable after a clock edge.

Example of Timing Constraints

Cell: FD1QHS (130-nm CORE9GPHS_1.2V_databook.pdf)

Conditions: Nominal 1.2V/25C

Constraint:	Expression:
D_CP_HOLD (fall)	0.005+0.080*Tr(CP)
D_CP_HOLD (rise)	0.005+0.027*Tr(CP)
D_CP_SETUP (fall)	0.116-0.092*Tr(CP)+0.136*Tr(D)
D_CP_SETUP (rise)	0.089-0.064*Tr(CP)+0.197*Tr(D)
Pulse Width High CP	0.040
Pulse Width Low CP	0.140

TIMING: CONCLUSION

- Timing analysis is essential to guaranteeing performance.
- Two options exist:
  - Simulation.
  - Static timing analysis.
- When applied in a design flow, timing analysis is closely connected to timing closure.