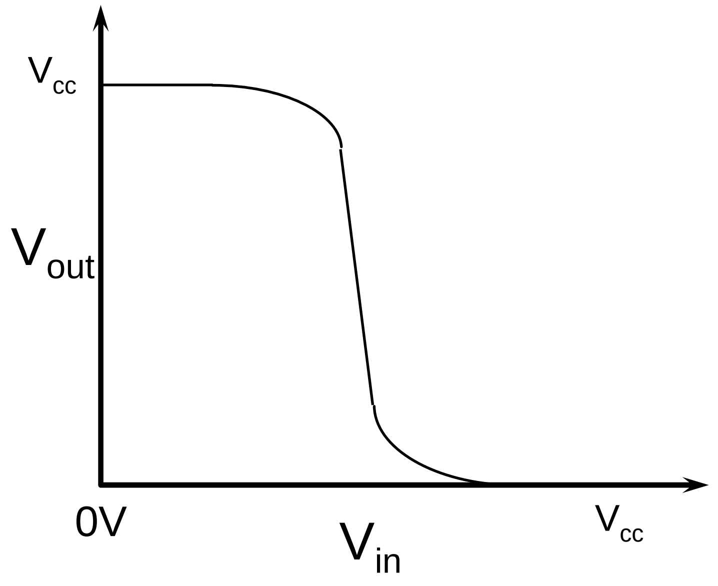
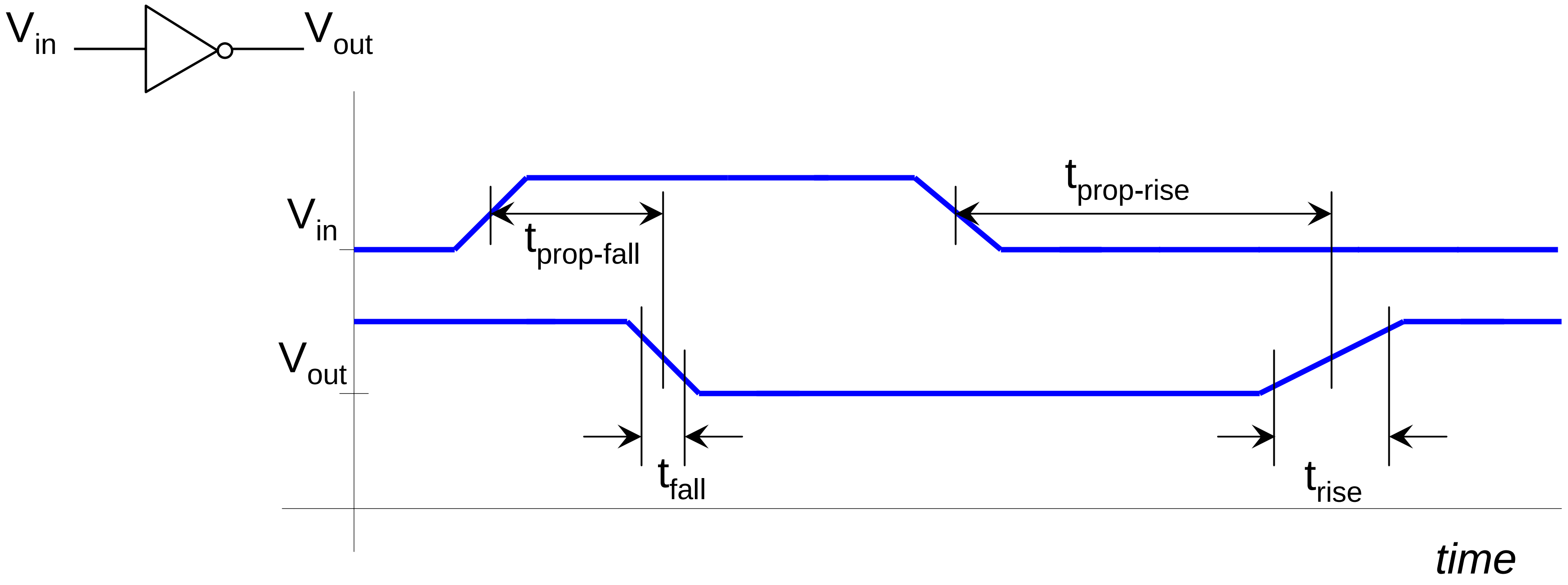
Integrated Circuits -- Timing Behavior of Gates

Gates Have Non-Linear Input/Output Behavior



Plotting Vout vs. Vin shows non-linear voltage behavior

Gates Also Don't React Immediately



$t_{prop\text{-}fall}$ and $t_{prop\text{-}rise}$ are measured from 50% of input swing to 50% of output swing

t_{rise} is measured from 10% of output swing to 90% of output swing
t_{fall} is measured from 90% of output swing to 10% of output swing

Kinds of Gate Delays

- Propagation Delays
  - Match intuition about *how fast* the gate is…
  - Rising and falling delays may be different
    - Take into acount which nodes rising, which nodes falling to compute delay through network
  - Rising and falling delays may be the same
    - Ignore rising/falling behavior
    - Will use most of this semester
- Rise and Fall Times
  - Only indirectly related to *how fast* the gate is…
  - Will not use in this course

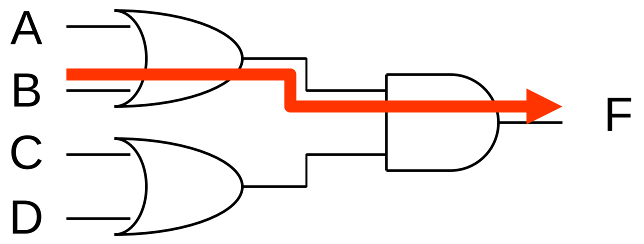
Gate Delays

- Delay values are usually provided for the gates you use
- Use those delay values when analyzing a circuit's timing
- Here are the timing values we will use this chapter
  - Wider gates are slower
  - AND = NAND+NOT, etc…

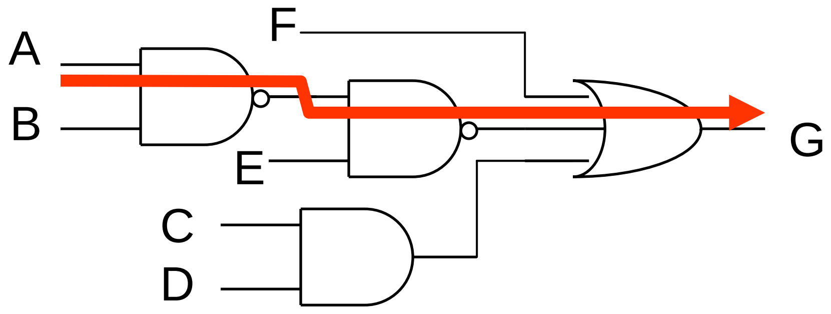
Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

Critical Path Analysis

- Given a logic circuit, how *fast* will it run?
  - How fast will its output react to changes on its inputs?
- Find the slowest path from any input to the output
  - That is the ***critical path***
  - Add up propagation delays along that path



Delay = $t_{OR2} + t_{AND2}$ = 7ns



Delay = 2 x $t_{NAND2} + t_{OR3}$ = 10ns

Kinds of Timing Analysis

- The more detail you provide ⇔ the more accurate the result
- Separate $t_{prop\text{-}rise}$ and $t_{prop\text{-}fall}$ ⇔ more accurate than a single t_{prop}
- Take into account other effects ⇔ more accurate
  - Temperature
  - Power supply voltage
  - Manufacturing variation
  - Rise/fall times

Environmental Delay Effects

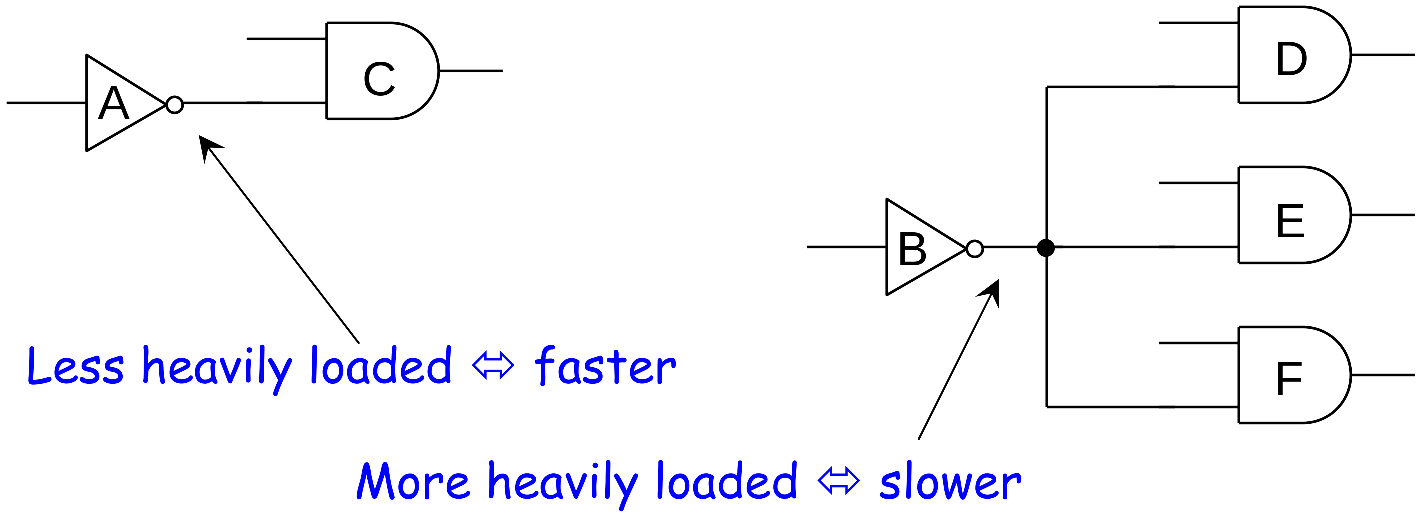
- Temperature Variation
  - Chips run faster when cool, slower when hot
- Voltage Variation
  - Value of Vcc can affect delay
- Manufacturing Variation
  - Two gates on the same integrated circuit may run at slightly different speeds

- Summary: gate delays are always given as a range of values that operational delay is guaranteed to be within.

- Example: 2ns +/- 0.2ns

Loading Effects

- A gate that drives more circuitry is slower than a gate that drives less circuitry



Typical Loading-Dependent Delay

- Typical delay = $t_{prop} + k \times C_{load}$

Loading-independent delay

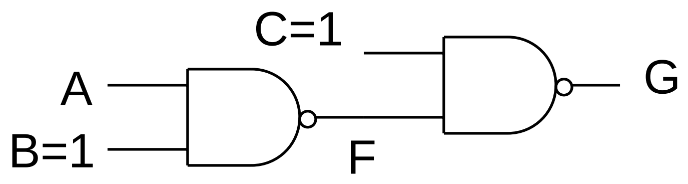
Load

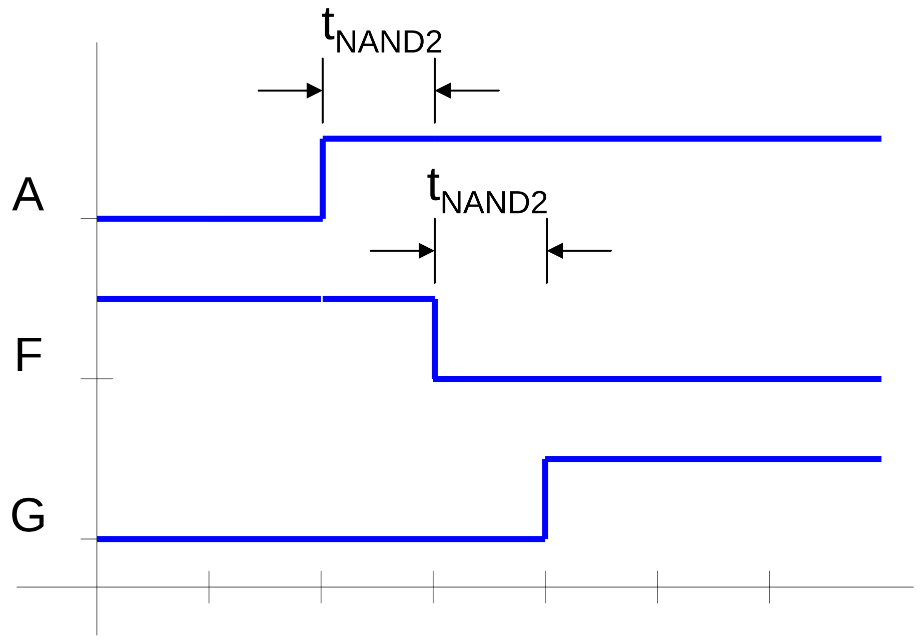
Loading-dependent factor

For MOS technology, load is mostly capacitance

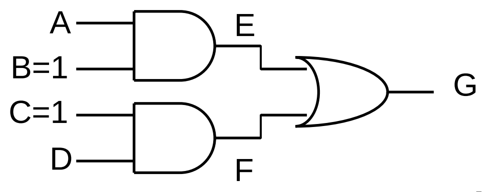
For bipolar technology, load is both capacitance and current

Analyzing A Logic Network

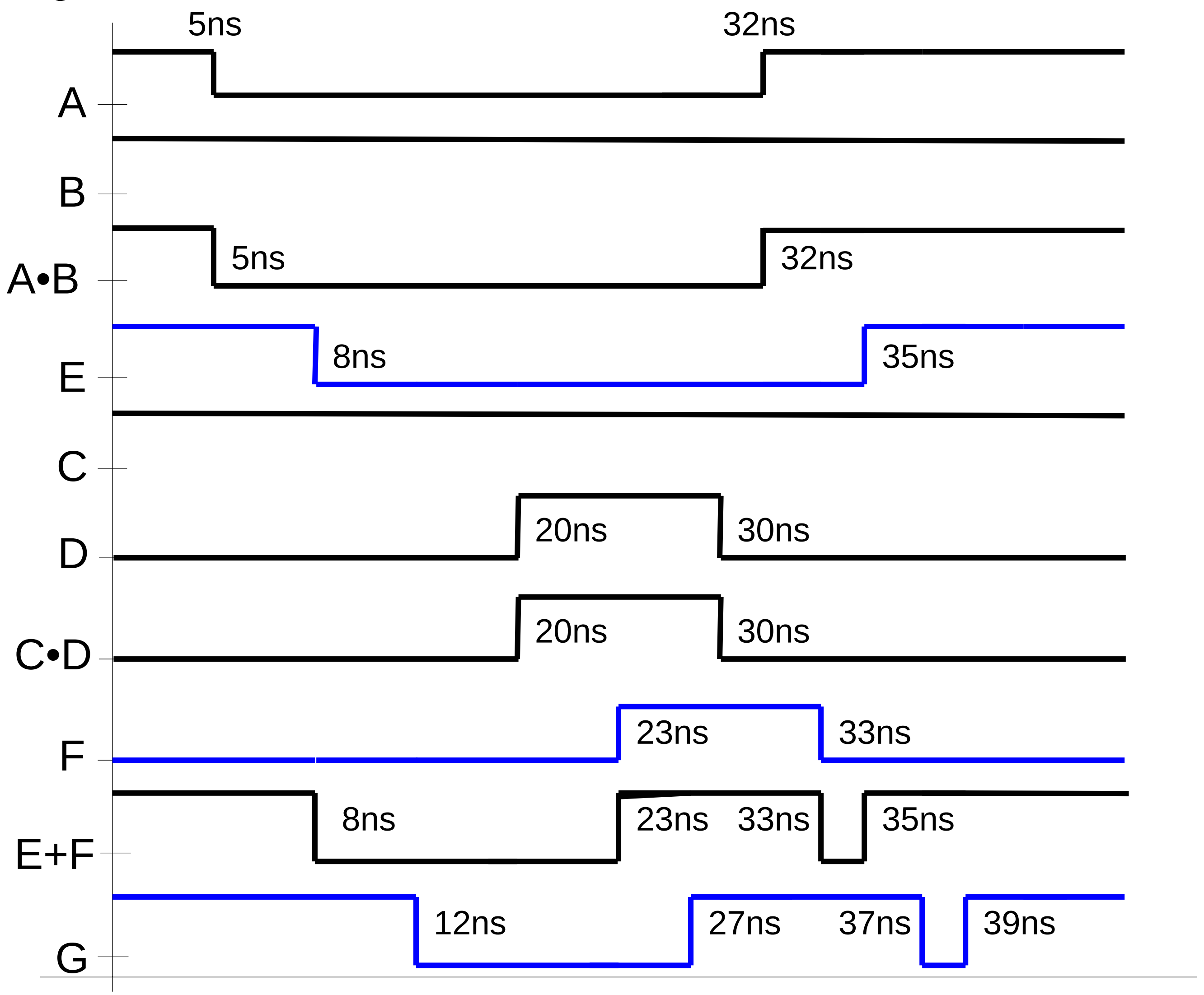


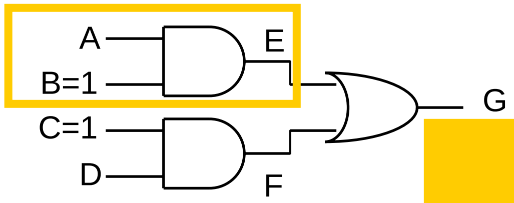


More Analysis

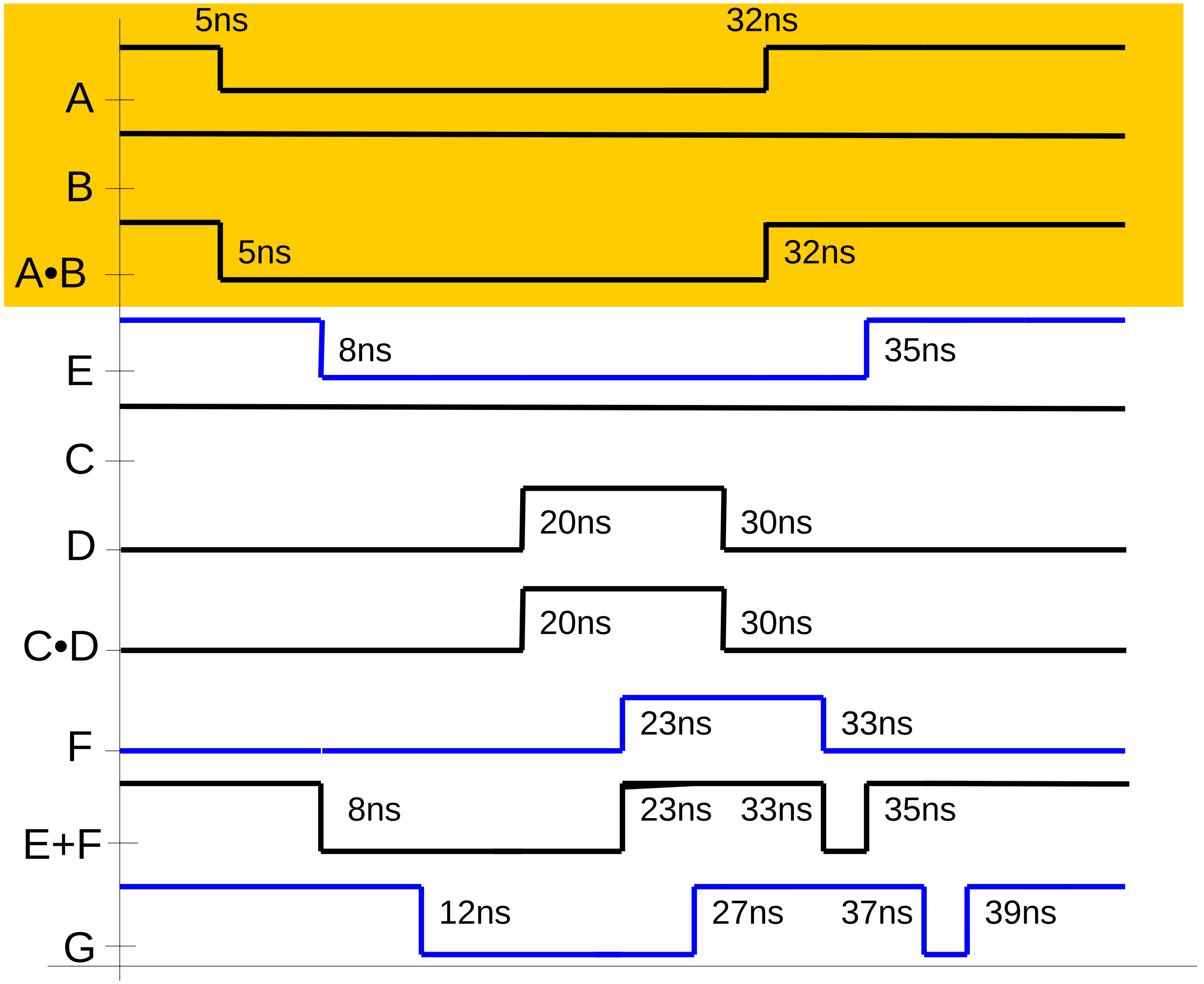


Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

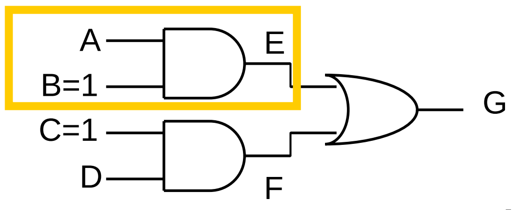




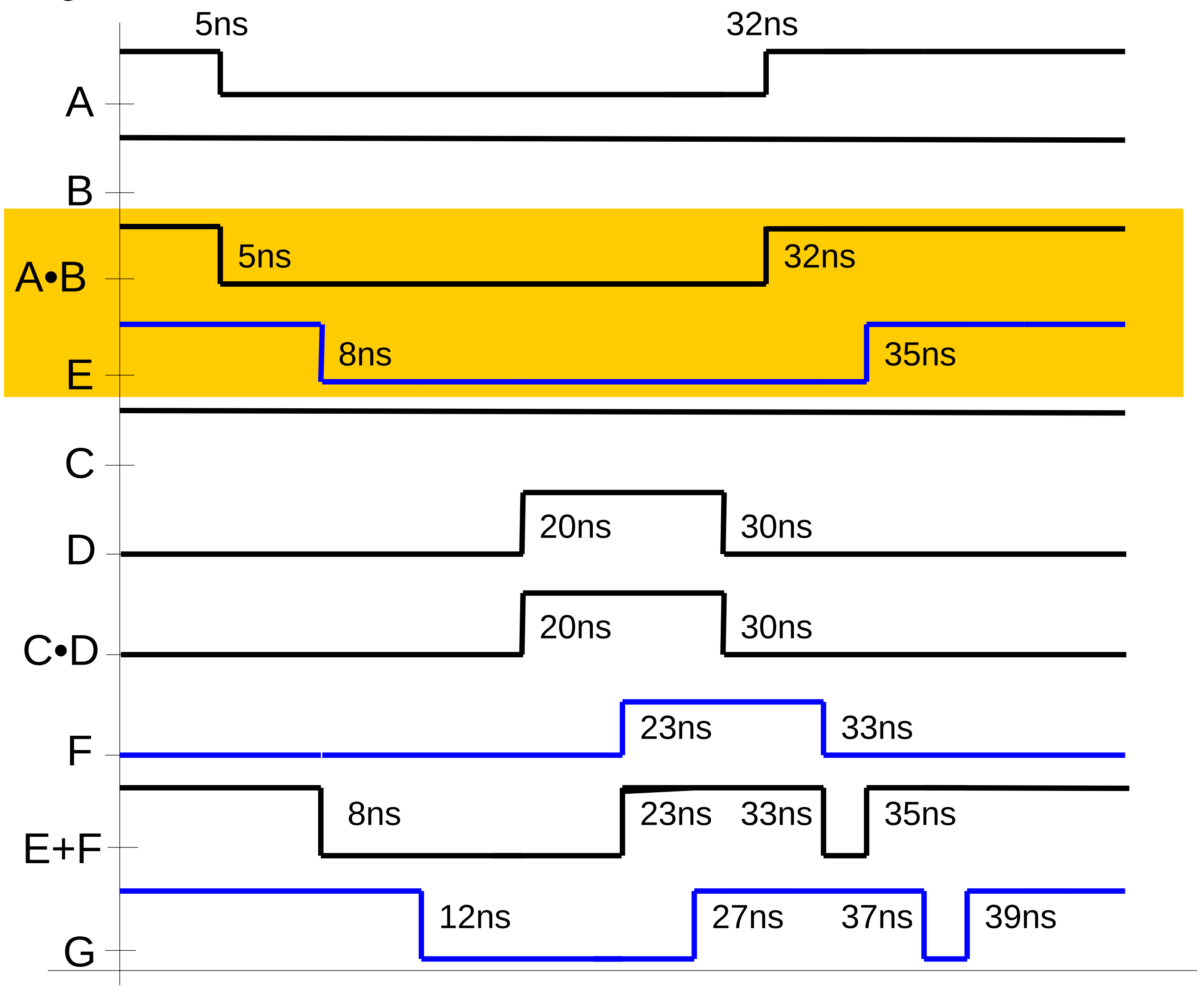
More Analysis



Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

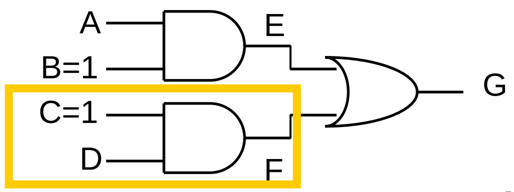


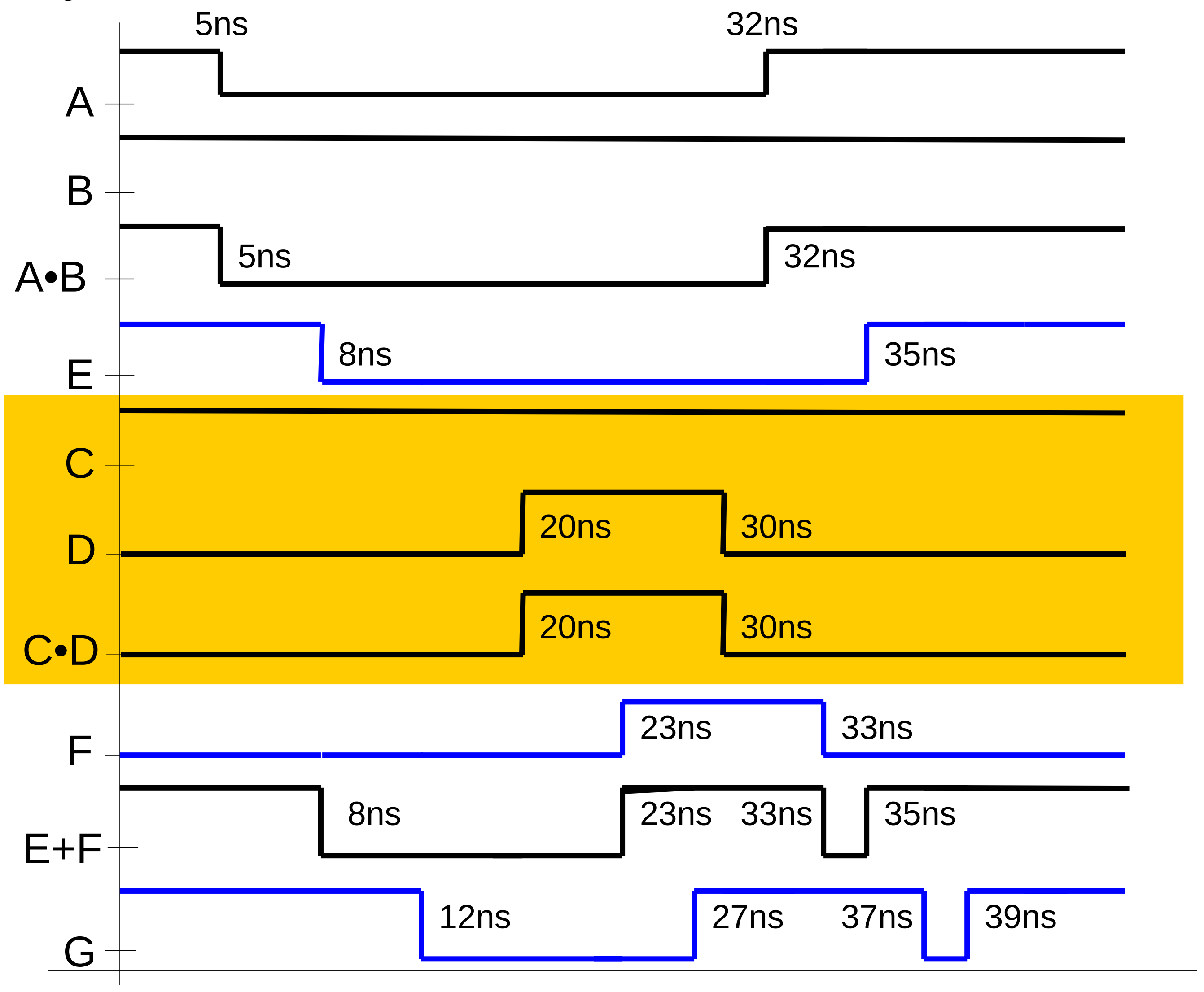
More Analysis



Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

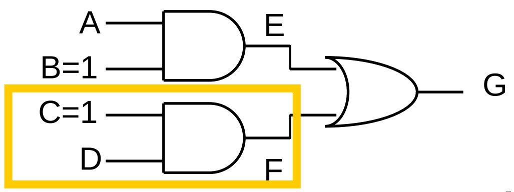
More Analysis



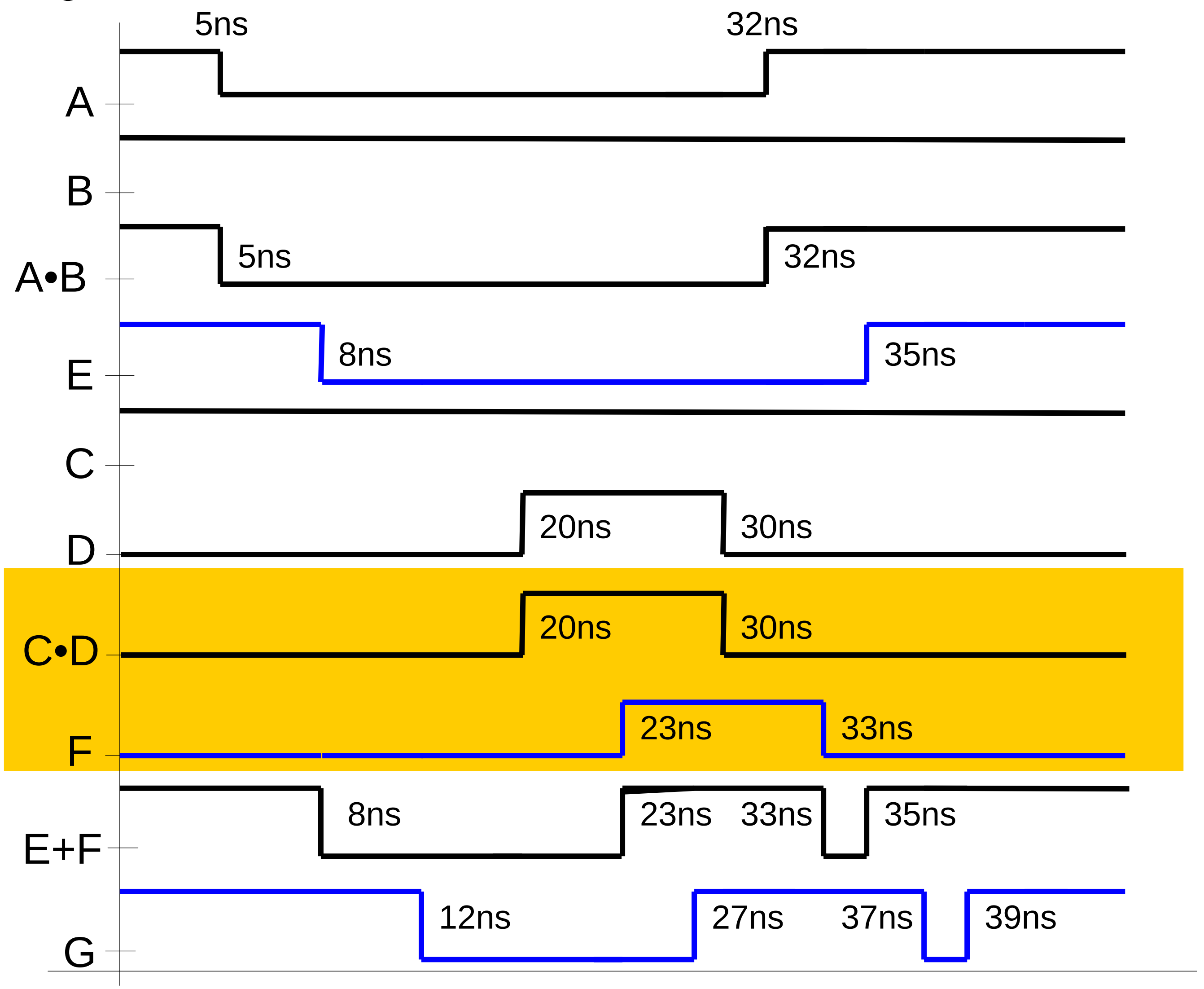


Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

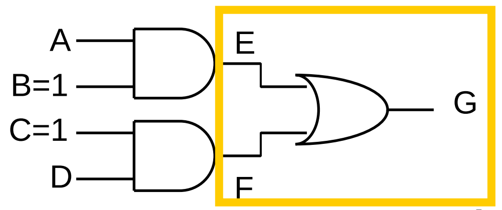
More Analysis

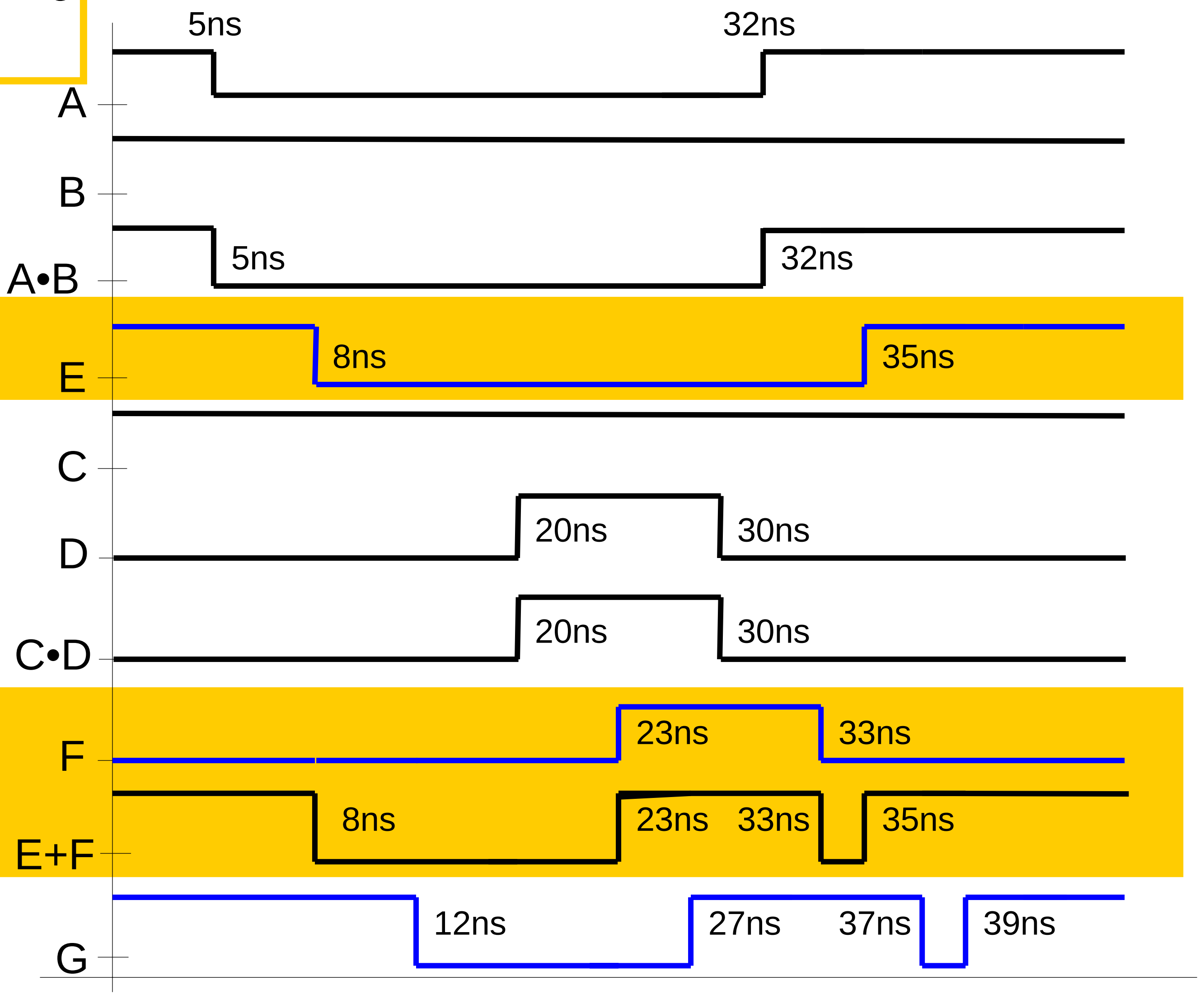


Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

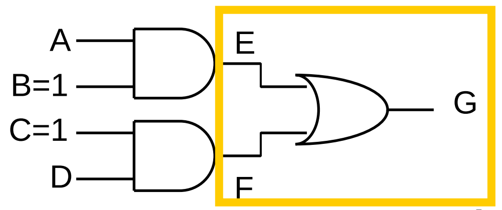


More Analysis

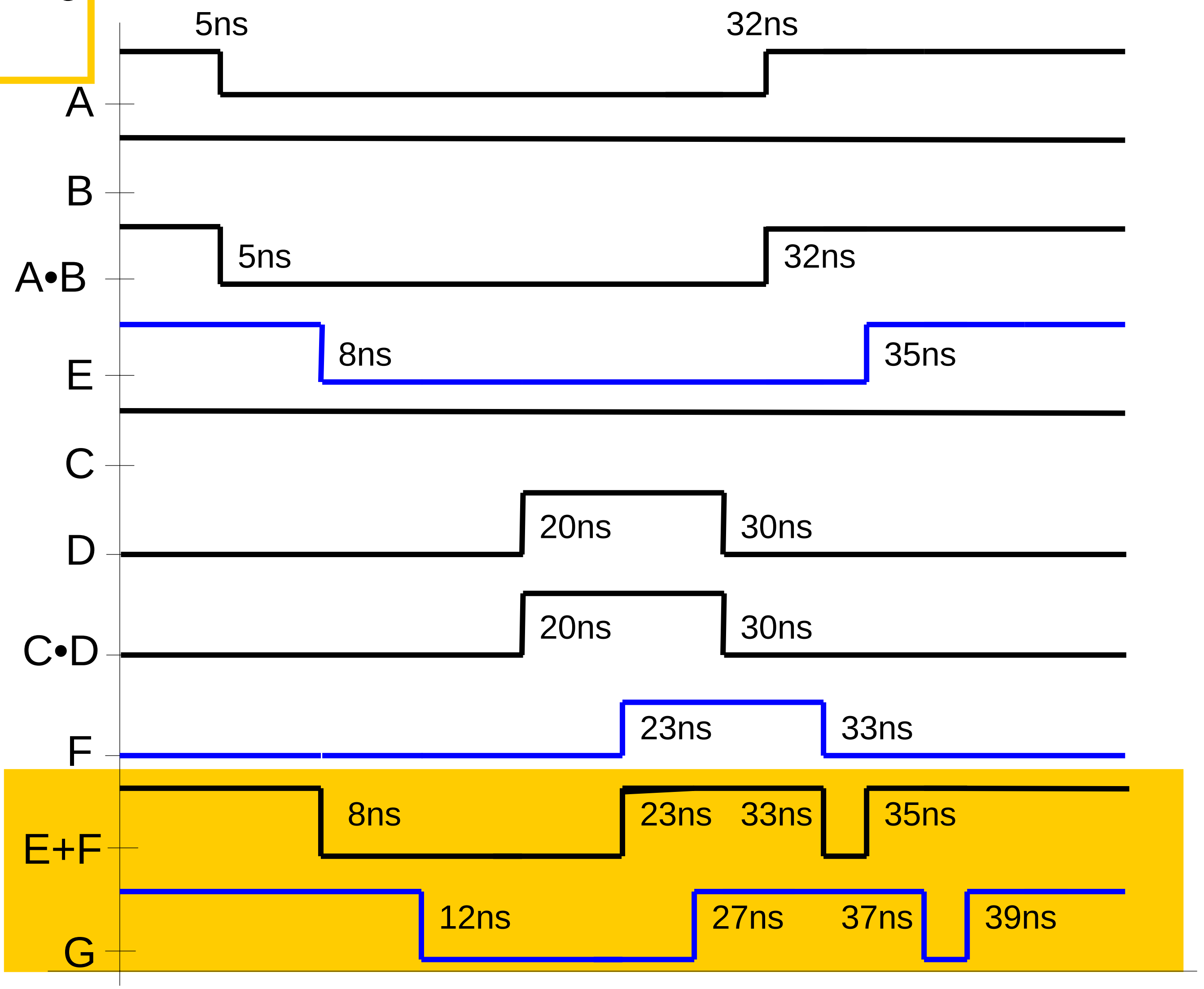




Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns



More Analysis

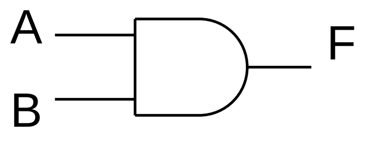


Type	t_{prop}
NOT	1ns
AND2	3ns
AND3	5ns
AND4	7ns
NAND2	2ns
NAND3	4ns
NAND4	6ns
OR2	4ns
OR3	6ns
OR4	8ns
NOR2	3ns
NOR3	5ns
NOR4	6ns
XOR2	5ns
XOR3	7ns

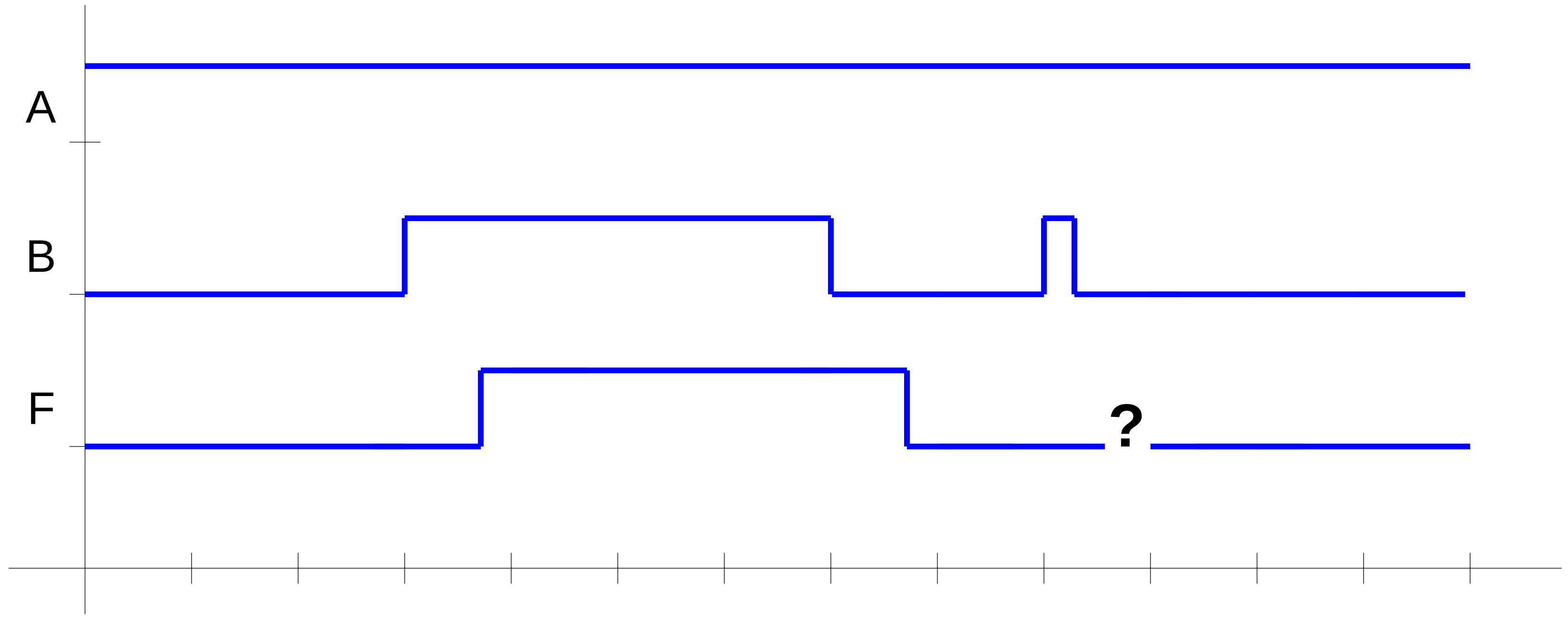
Completing a Timing Diagram – A Method

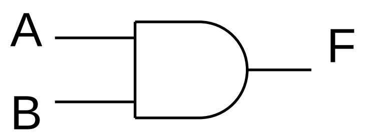
- Do *most upstream* signals first
  - *E* and *F* in previous picture
- For every instant in timing diagram
  - Compute each gate output as function of input(s)
  - Place new gate output after appropriate delay on timing diagram.
- Move to downstream signals

Glitch Behavior



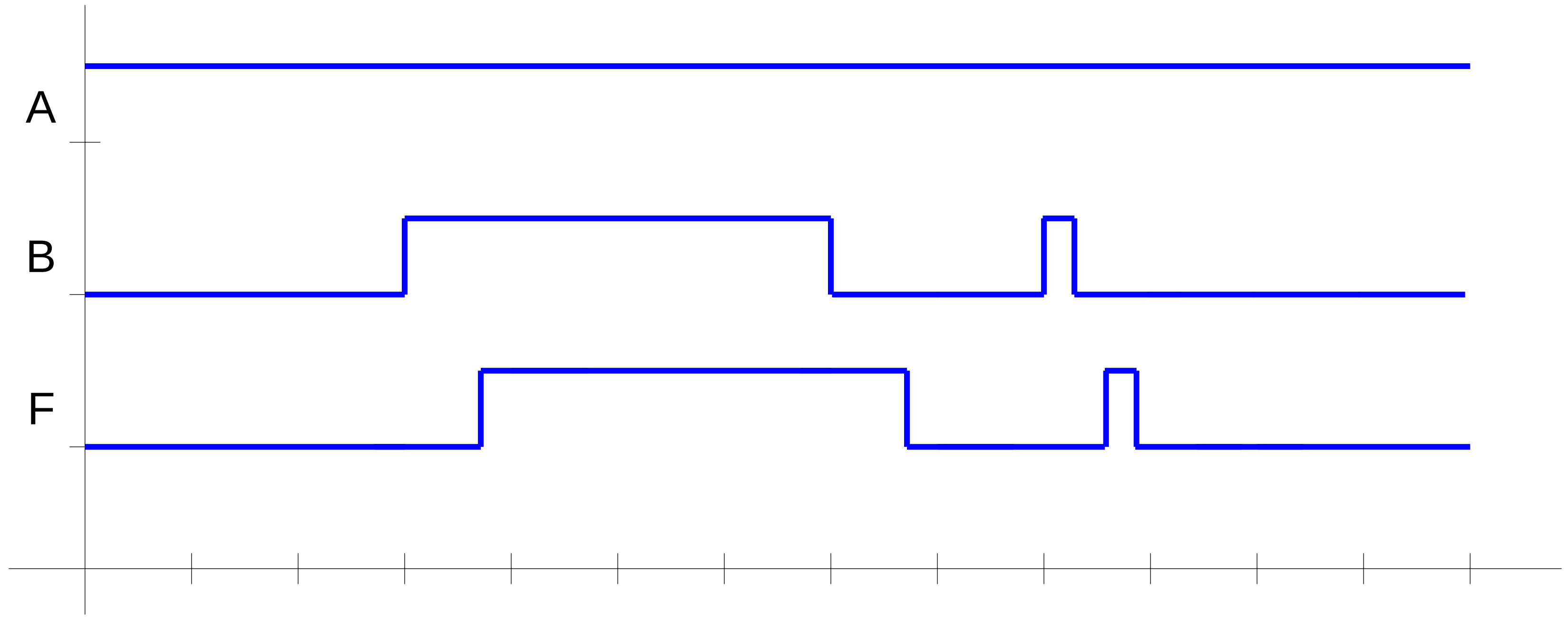
- Does the glitch propagate to the AND output?

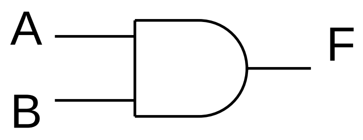




Behavior #1

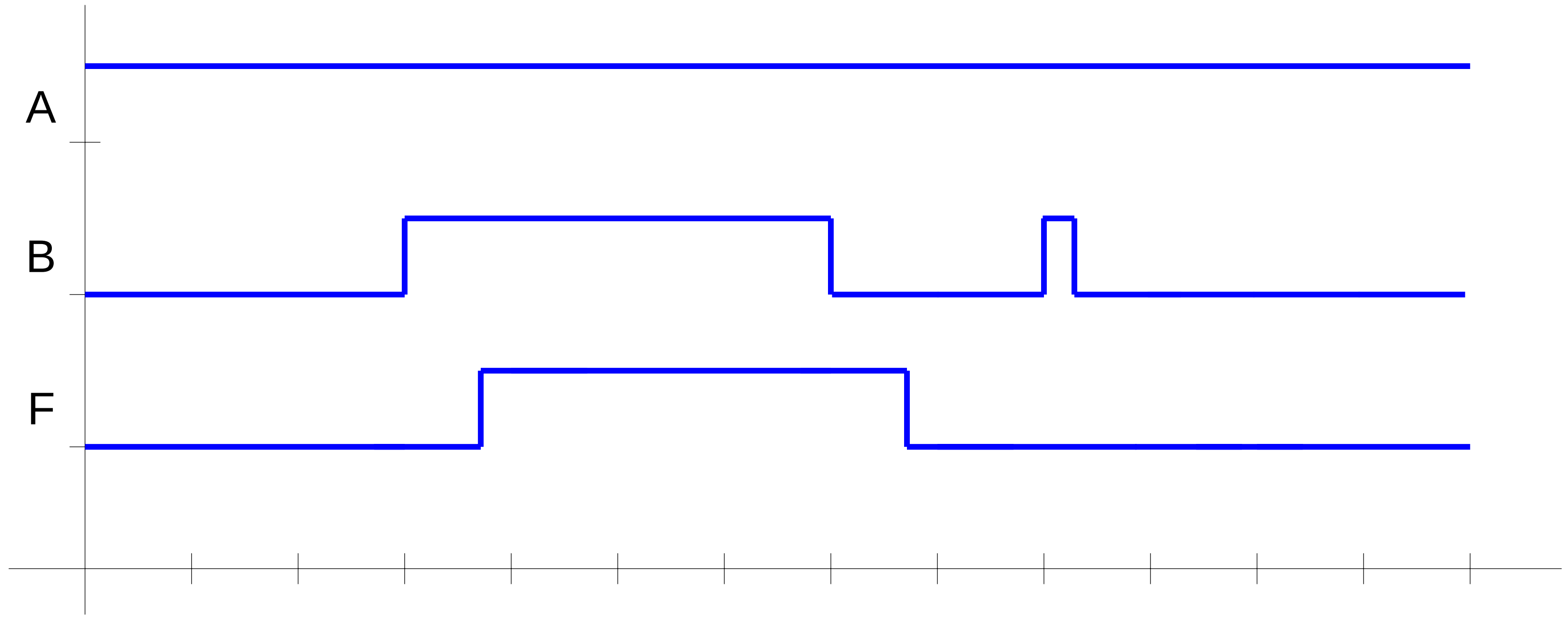
- Called *transport* delay
  - All glitches, no matter how narrow are propagated to output





Behavior #2

- Called *inertial* delay
  - Glitches narrower than some threshold are filtered out



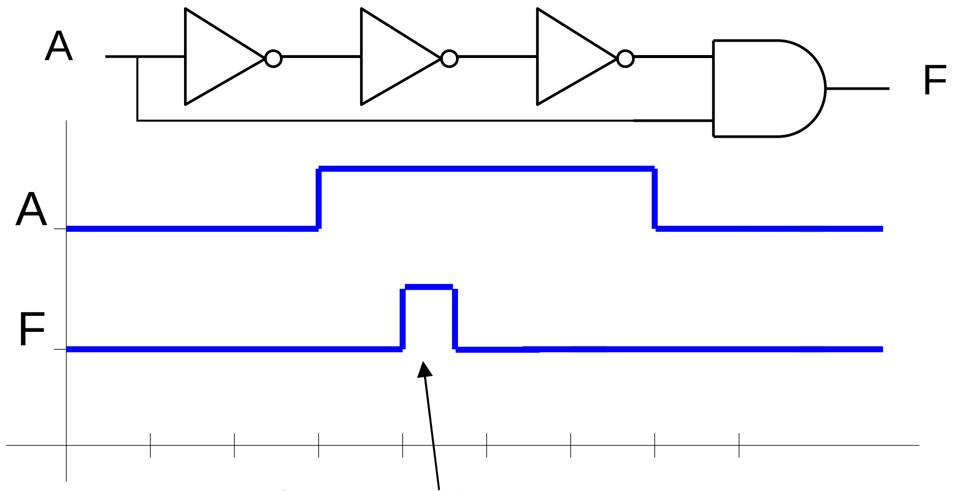
Which Behavior Is Correct?

- Wires usually exhibit <u>transport</u> delay
- Gates usually exhibit some form of <u>inertial</u> delay
- Regardless, both wires and gates will filter out (or change the waveform shape of) <u>*extremely*</u> narrow pulses
- We will use <u>transport</u> for both gates and wires since we don't know the pulse width thresholds

Completing a Timing Diagram – A Shortcut Method

- Focus on input changes (edges)
- Everything constant in between

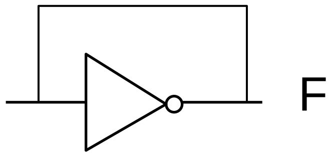
A Pulse Generator



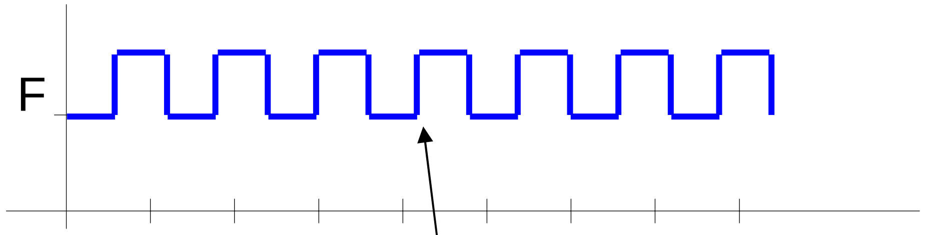
Pulse width = 3 x t_{NOT}

- F = A•A' = 0 so why the pulse?
  - Due to gate delays

More Dynamic Gate Behavior



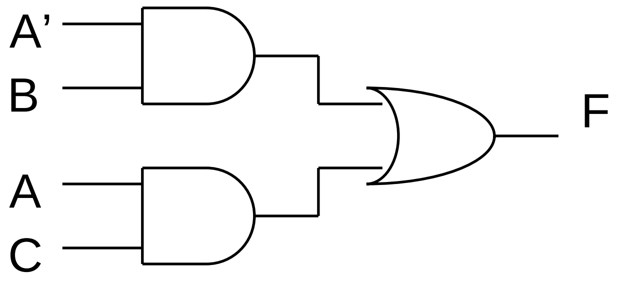
- What will this circuit do?
  - It will oscillate (0 – 1 – 0 – 1 - …)



Pulse width = t_{NOT}

Period of oscillation is 2 x t_{NOT}

Logic Hazards



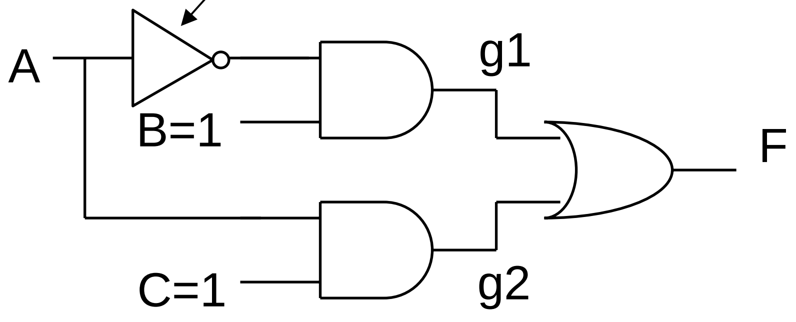
BC \ A	0	1
00	0	0
01	0	1
11	1	1
10	1	0

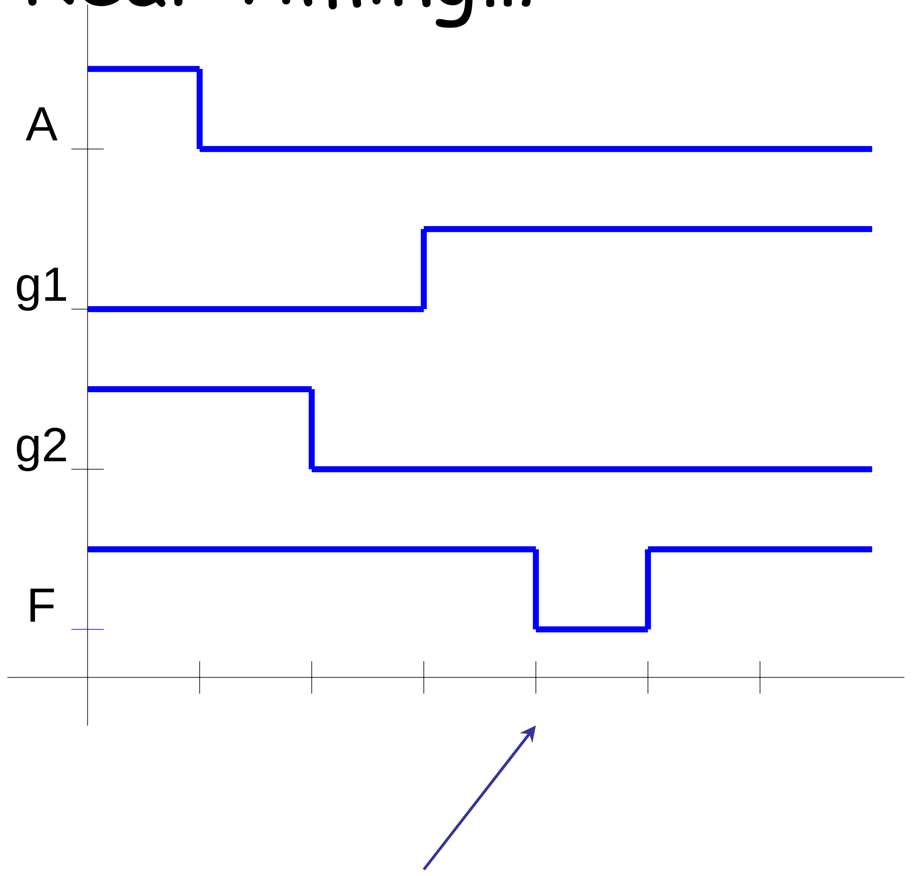
F = A’B + AC

This is the conventional KMap solution

Gates Have Real Timing…

Need to take into account timing of this inverter …

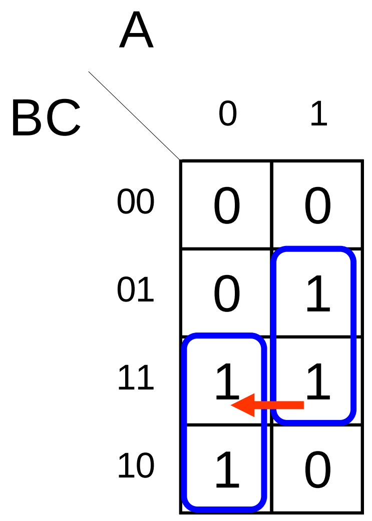




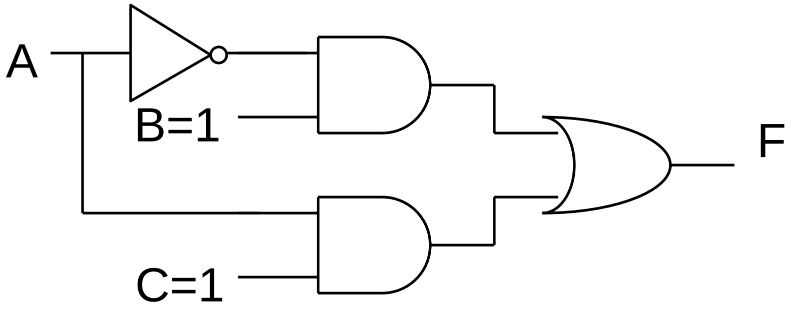
Called a hazard or false output – static equations indicate F=1 but dynamic behavior gives a “glitch”

Hazards

- Each prime implicant from KMap is implemented using a single AND gate
- When moving between implicants, gates turn off and on at different times
- Momentarily get false outputs

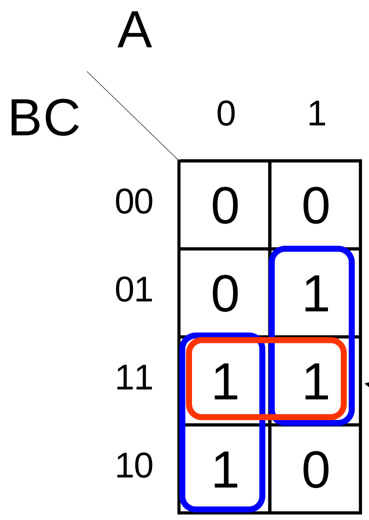


BC \ A	0	1
00	0	0
01	0	1
11	1	1
10	1	0



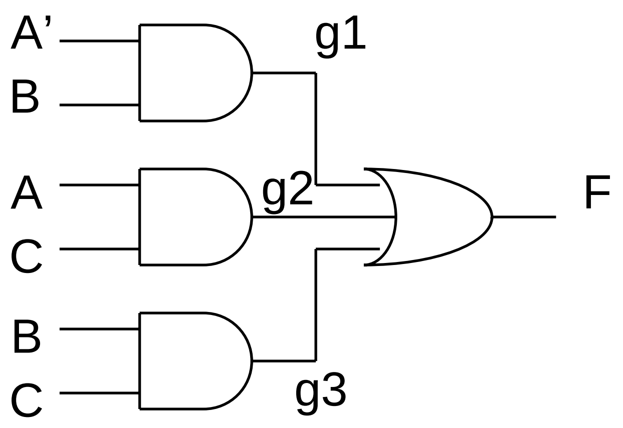
Hazard-Free Logic Design

- Make sure all prime implicants *overlap*



BC \ A	0	1
00	0	0
01	0	1
11	1	1
10	1	0

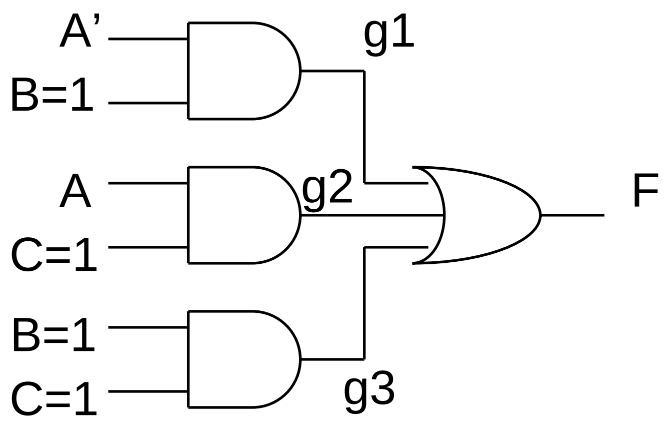
Redundant but will eliminate false output

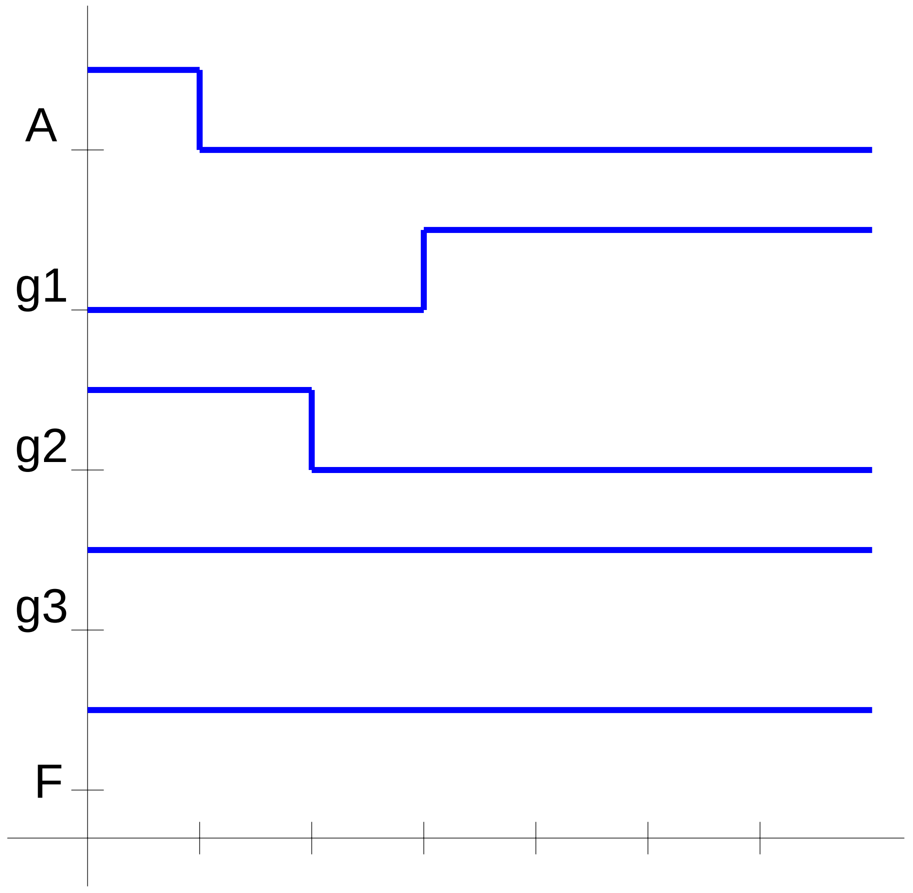


F = A’B + AC + BC

On ABC = ‘111’ to ABC = ‘011’, g3 will hold F high entire time.

No False Output…

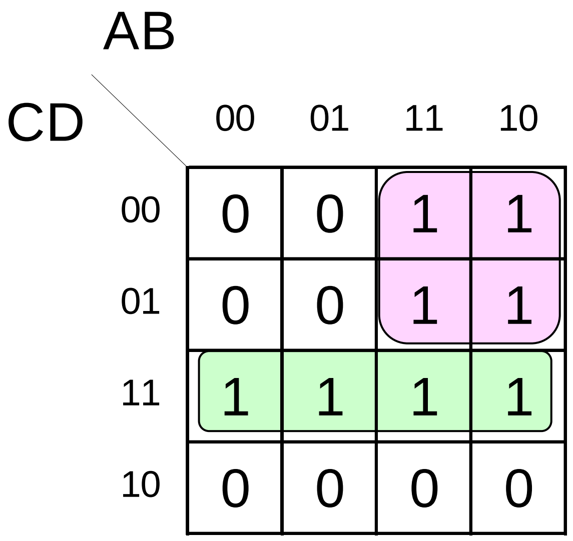




g3 holds F high whole time…

Another Example

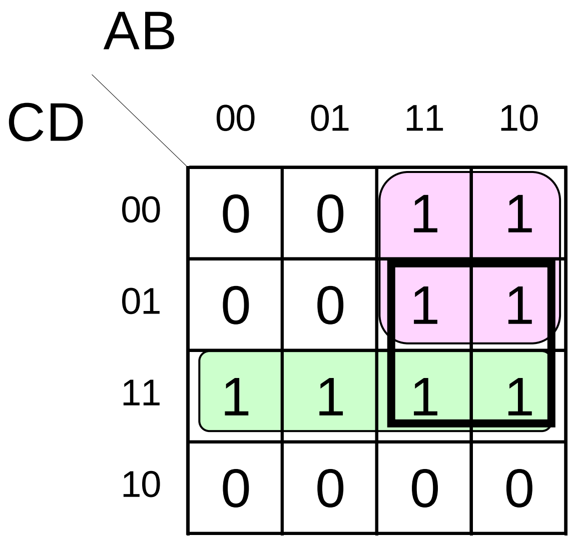
Rule: No 2 adjacent 1's should be in different implicants



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	1
10	0	0	0	0

F = CD + AC'

This is ***<u>not</u>*** hazard-free



CD \ AB	00	01	11	10
00	0	0	1	1
01	0	0	1	1
11	1	1	1	1
10	0	0	0	0

F = CD + AC' + AD

This ***is*** hazard-free

When To Do Hazard-Free Design?

- Do you <u>always</u> need to do HFD?
  - No…

- Do it when you need to generate a signal which will not glitch
  - Interfacing with other circuitry which is sensitive to edges
    - Asynchronous memories

A Caveat

- Will only work for single-input changes

- Other techniques for multiple-input changes