Overview of Static Timing Analysis in OpenSTA

Latch-Up 2025

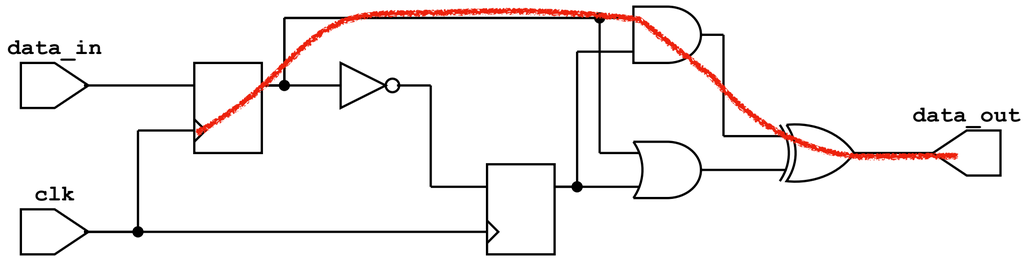
05/03/25

Akash Levy, CTO @ Silimate Inc.

Talk Overview

Primer on static timing analysis

- What is static timing analysis (STA)?
- STA concepts
- Synopsys Design Constraints (SDC)
- Open-Source Liberty Standard
  - Non-Linear Delay Model (NLDM)
  - Composite Current Source (CCS)
- OpenSTA: open-source implementation of all the above, status update



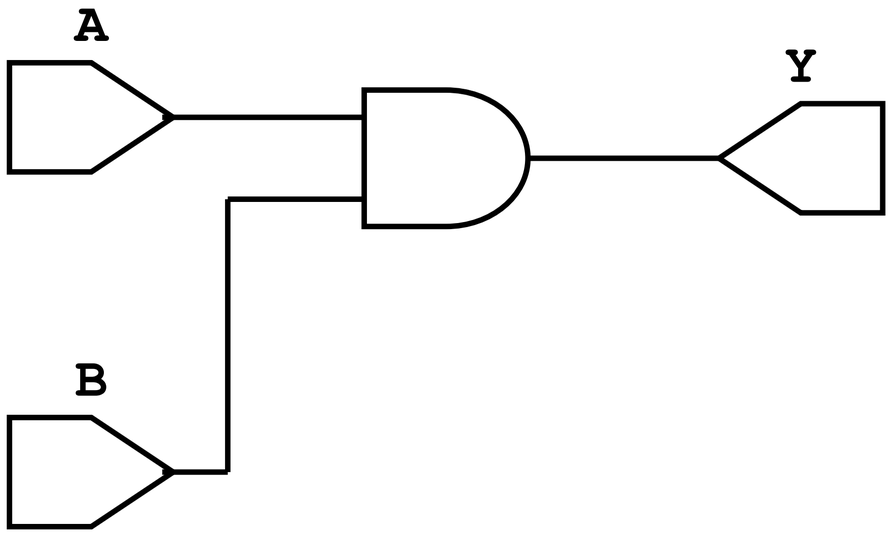
What is static timing analysis (STA)?

And why do we need it?

- Statically analyze a digital circuit to:
  - Calculate maximum clock frequency
  - Prevent metastability and incorrect data capture
  - Find setup/hold critical paths
  - Define timing interface to other blocks/chips
- No simulation required to validate timing behavior!

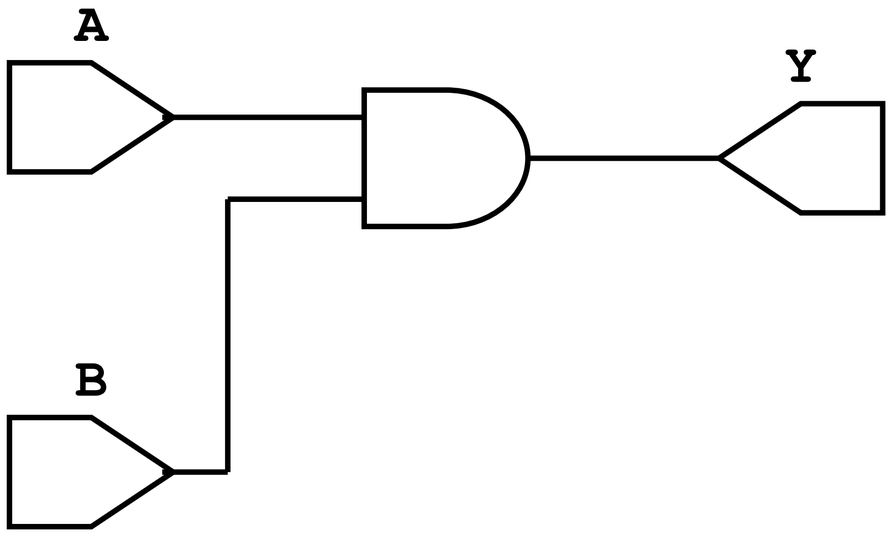
Combinational Timing Arcs

AND gate



Combinational Timing Arcs

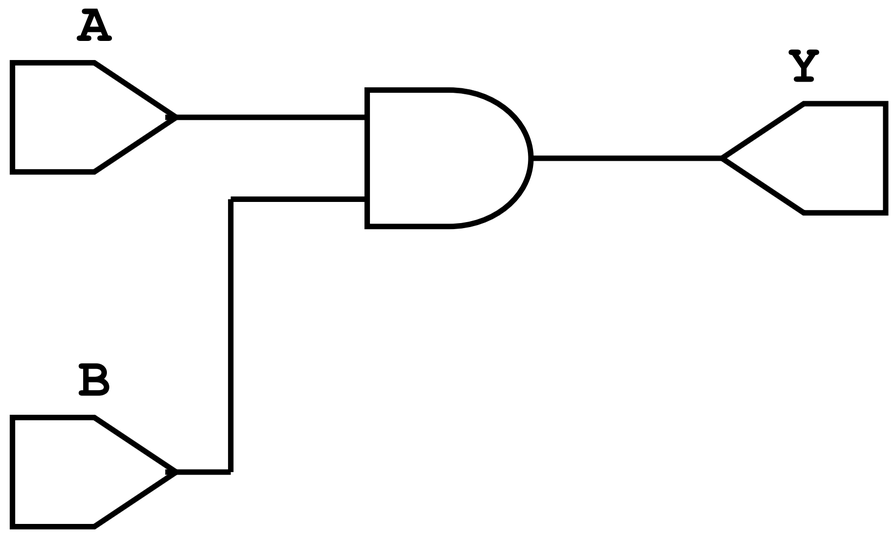
AND gate



Question: How many possible timing arcs to consider?

Combinational Timing Arcs

AND gate

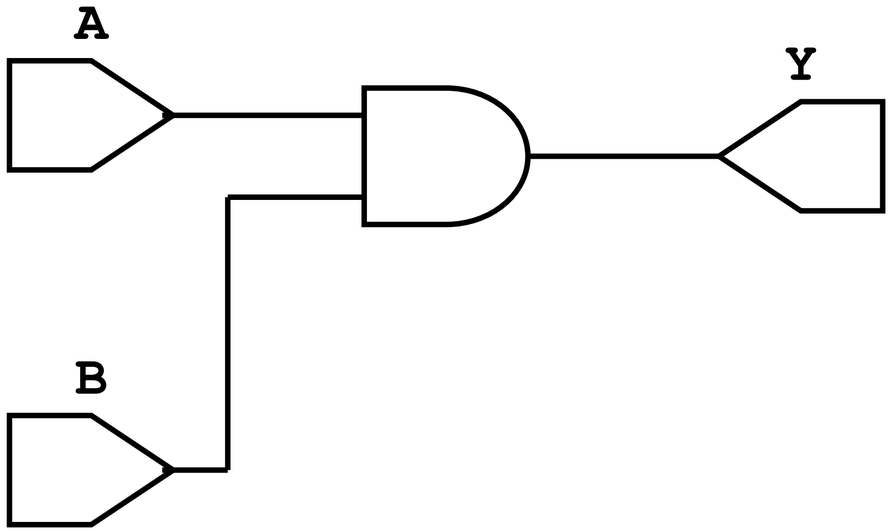


Question: How many possible timing arcs to consider?

Answer: 4, both A and B are positive unate w.r.t. Y

Combinational Timing Arcs

AND gate

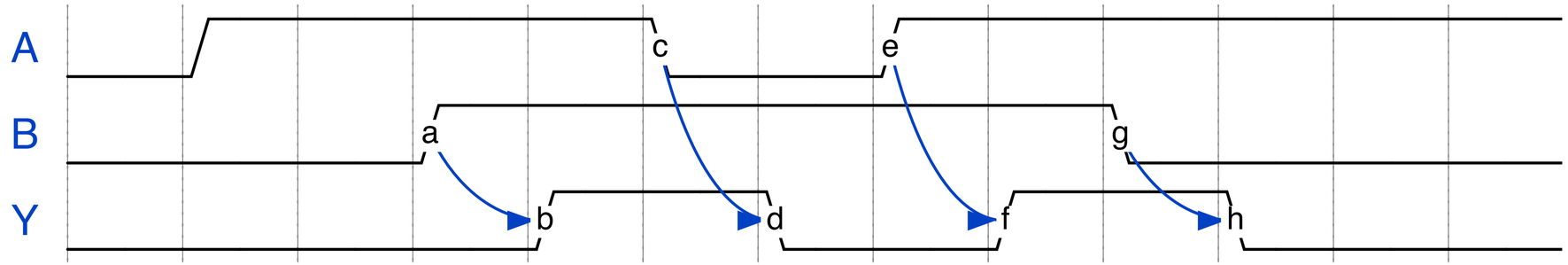


a→b: B↑, Y↑

c→d: A↓, Y↓

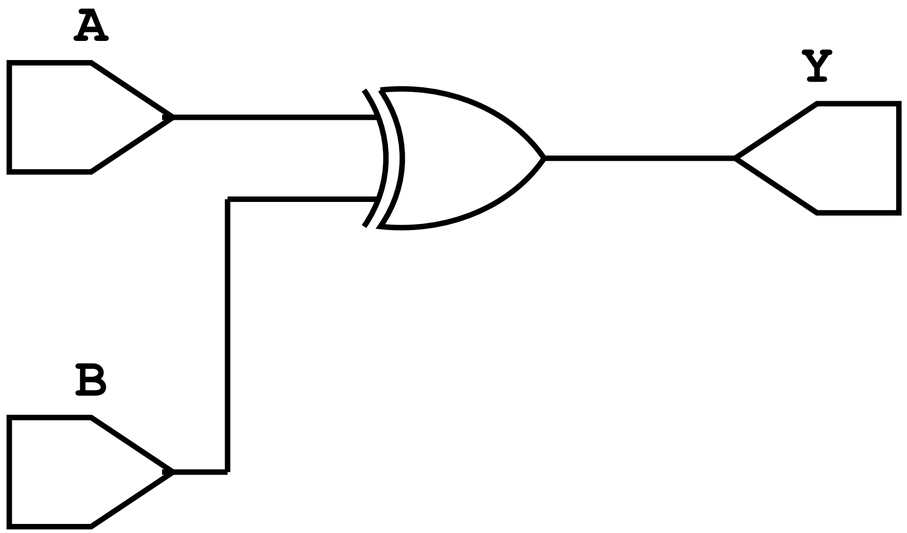
e→f: A↑, Y↑

g→h: B↓, Y↓



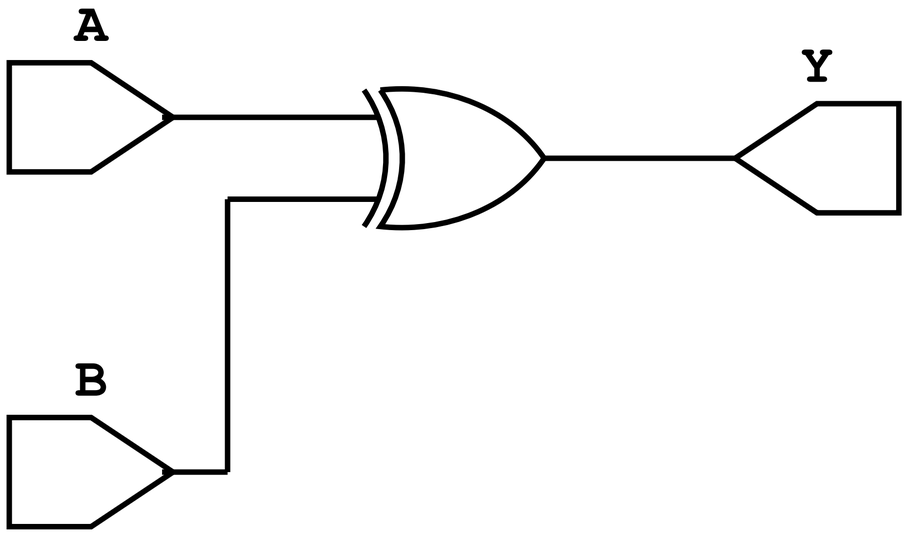
Combinational Timing Arcs

XOR gate



Combinational Timing Arcs

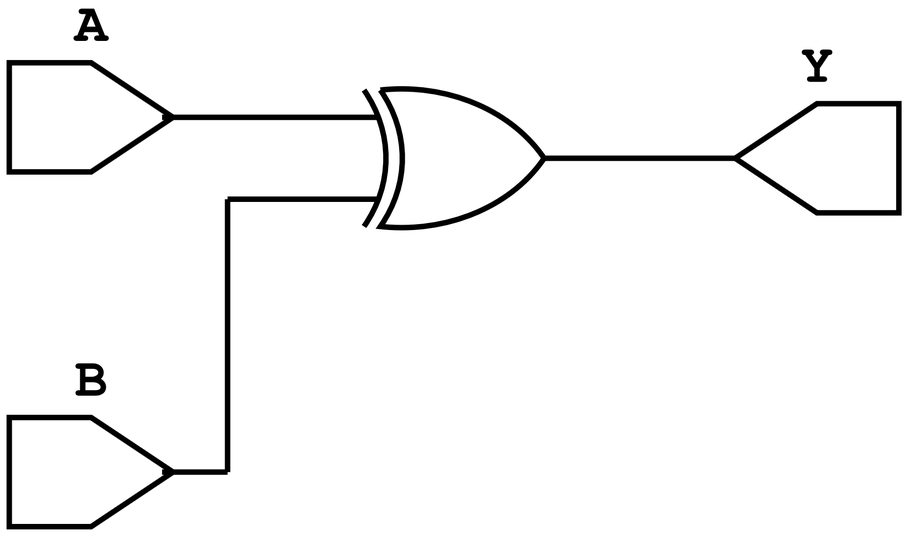
XOR gate



Question: How many possible timing arcs to consider?

Combinational Timing Arcs

XOR gate

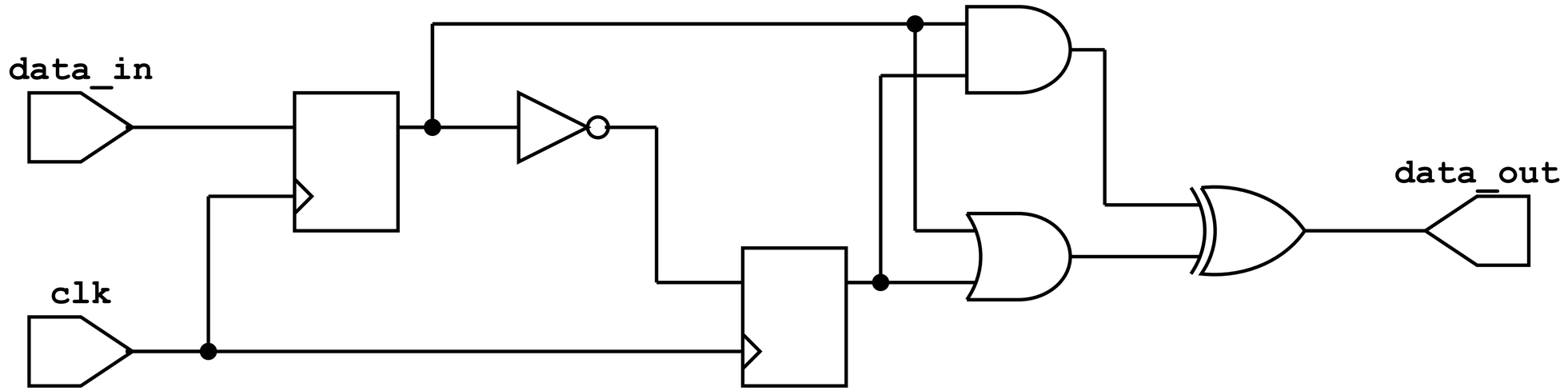


Question: How many possible timing arcs to consider?

Answer: 8, both A and B are non-unate w.r.t. Y

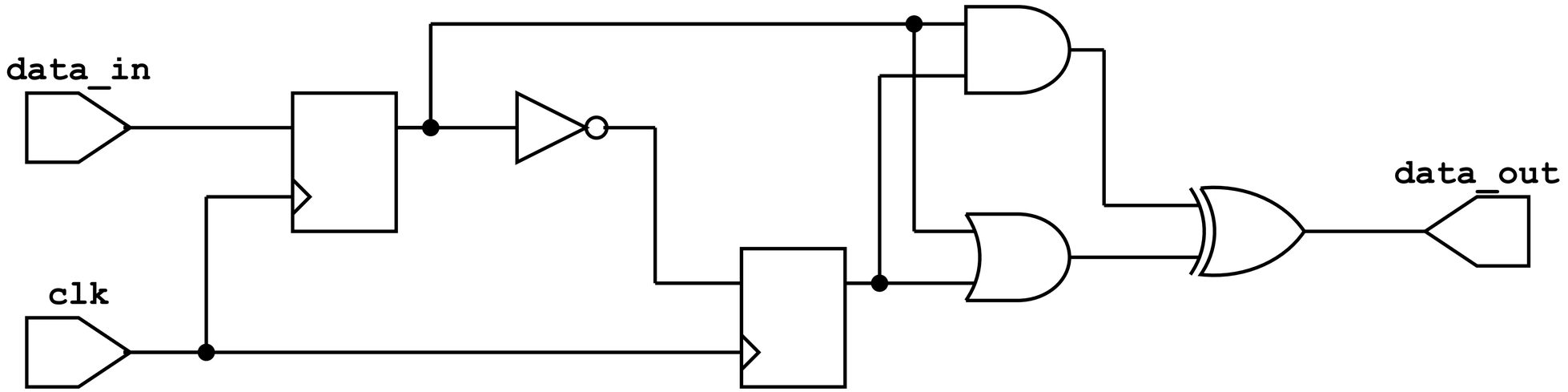
Launch and Capture

Startpoints and Endpoints



Launch and Capture

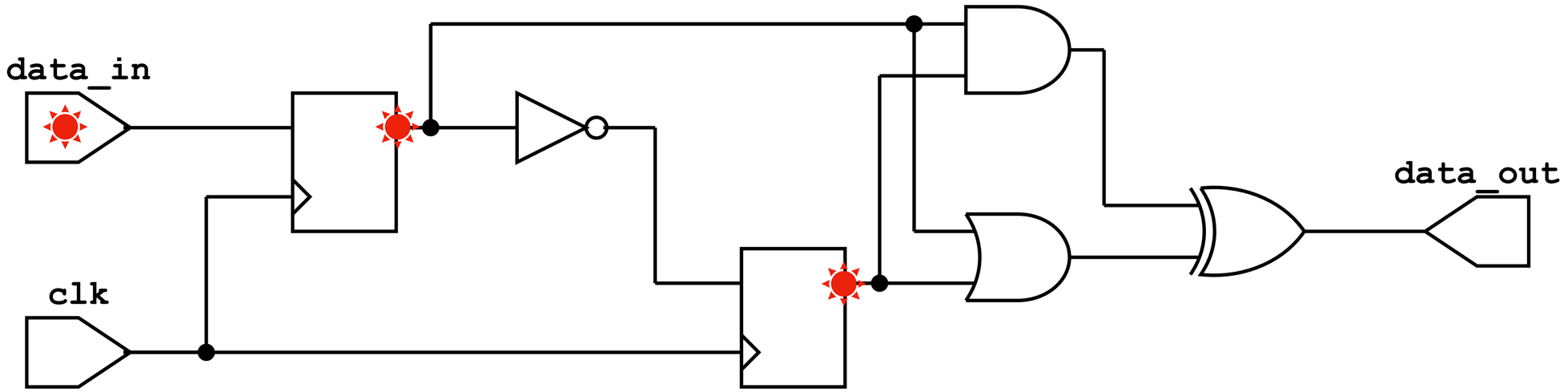
Startpoints and Endpoints



How many startpoints?

Launch and Capture

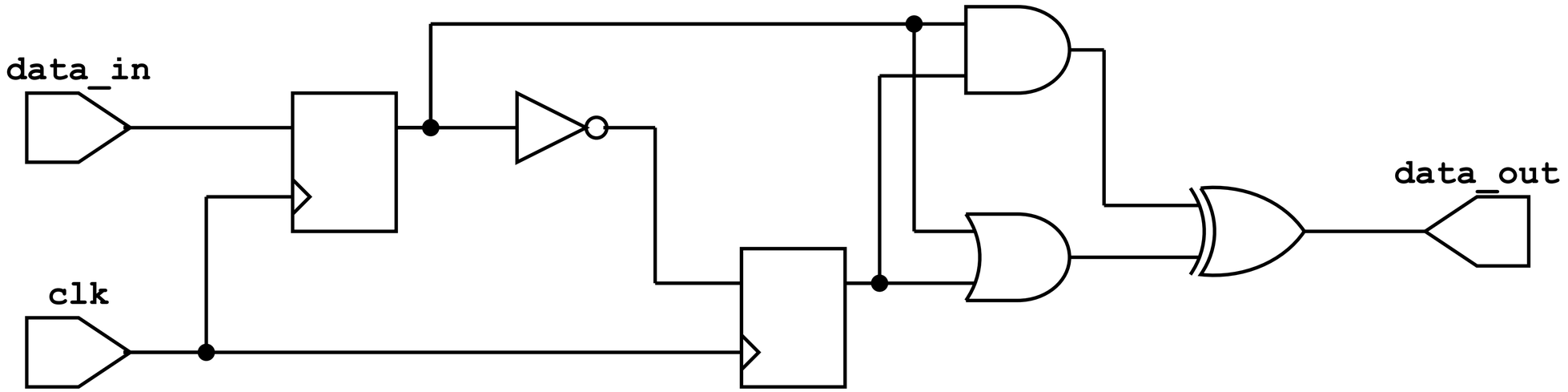
Startpoints and Endpoints



How many startpoints?

Launch and Capture

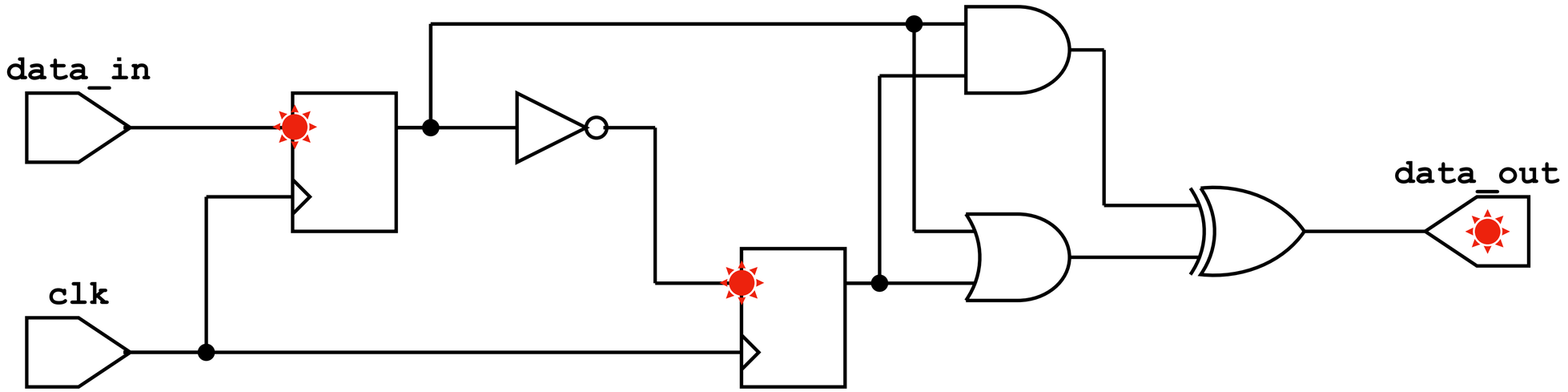
Startpoints and Endpoints



How many endpoints?

Launch and Capture

Startpoints and Endpoints

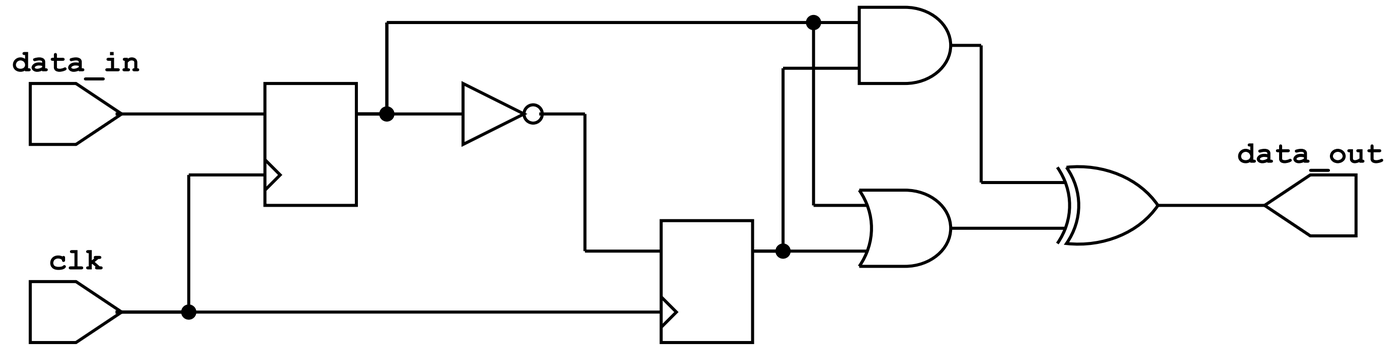


How many endpoints?

Endpoint Constraints

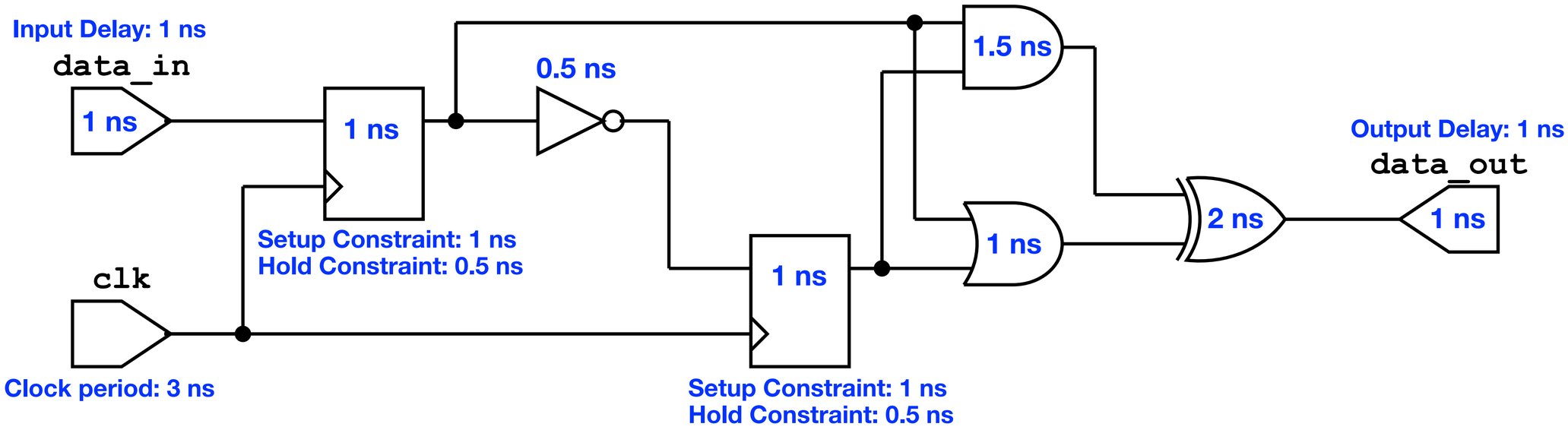
Setup and hold time concepts

- Setup time: stable time required before capture clock edge
- Hold time: stable time required after capture clock edge
- Violations lead to:
  - Incorrect data capture
  - Metastability
- Required times
  - t_min > t_hold, t_max < t_setup
- Slack: worst negative slack (WNS), total negative slack (TNS)



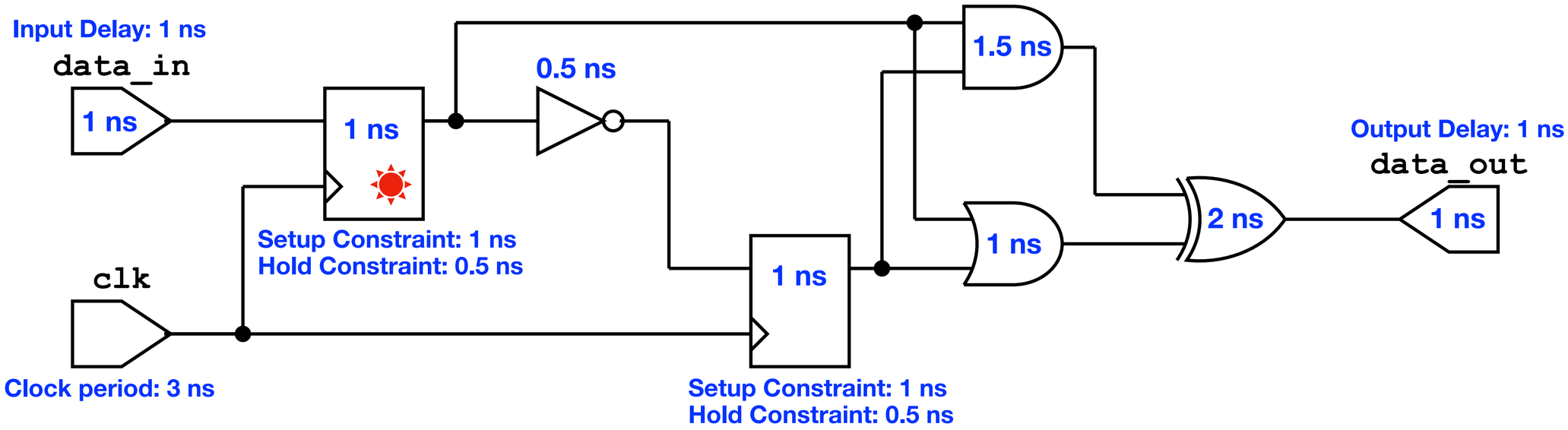
Static Timing Analysis

An simple example run-through



Static Timing Analysis

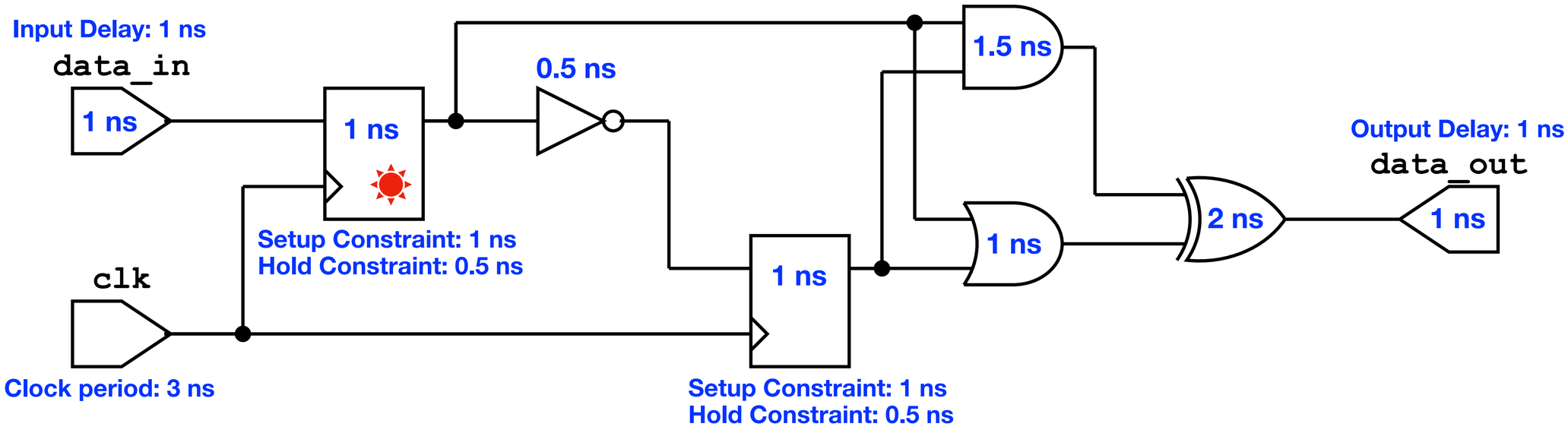
An example



Question: What is the setup required time?

Static Timing Analysis

An example

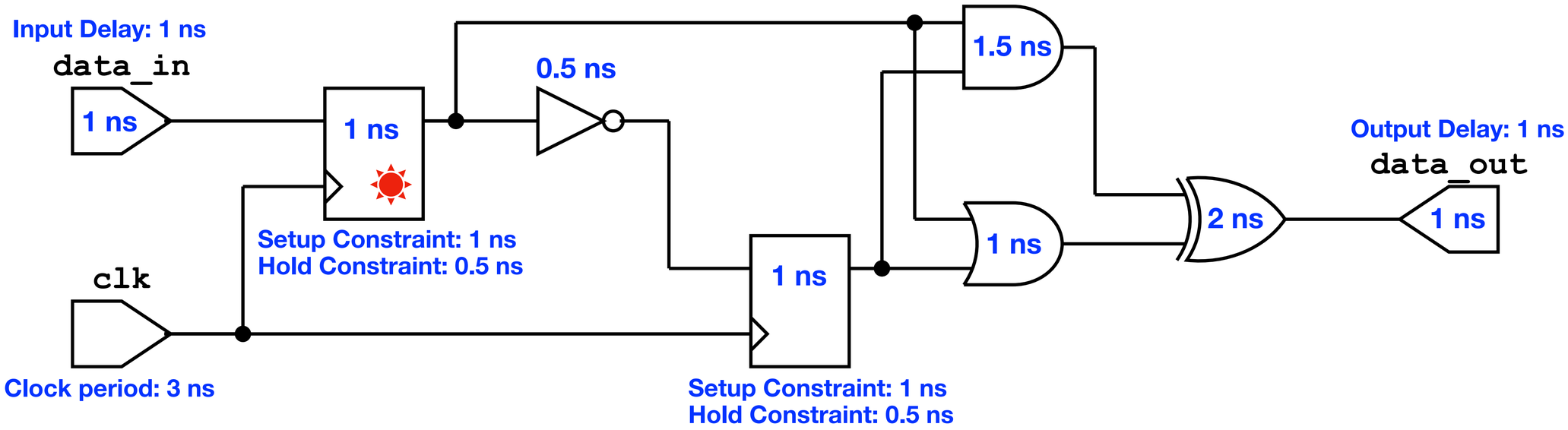


Question: What is the setup required time?

Answer: t_clk - t_setup = 2 ns

Static Timing Analysis

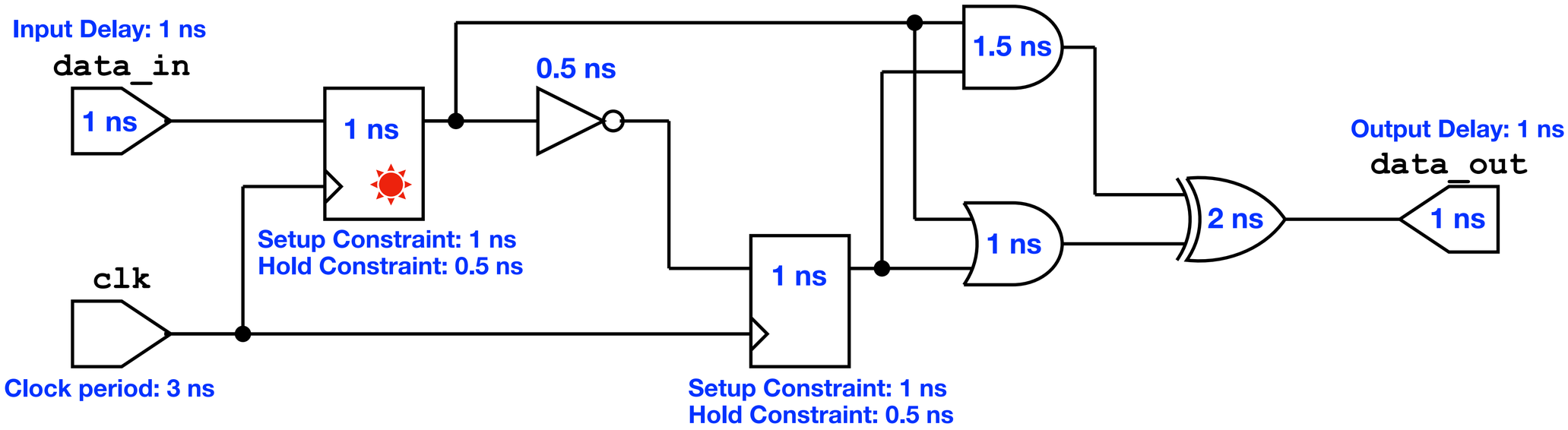
An example



Question: What is the setup arrival time?

Static Timing Analysis

An example

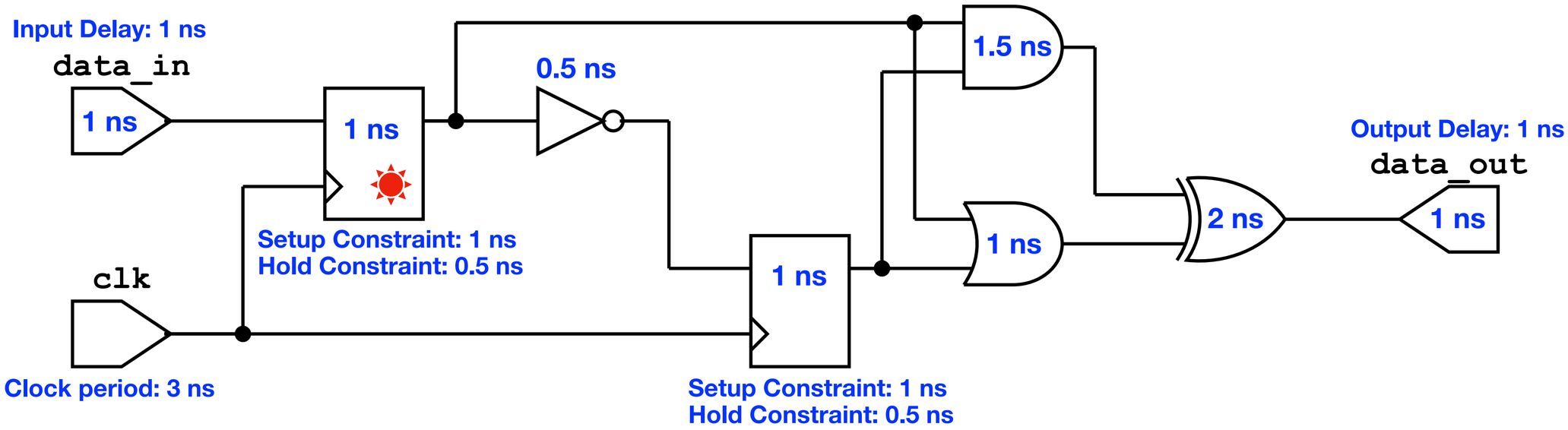


Question: What is the setup arrival time (slowest path)?

Answer: 1 ns. Requirement was < 2 ns (MET)

Static Timing Analysis

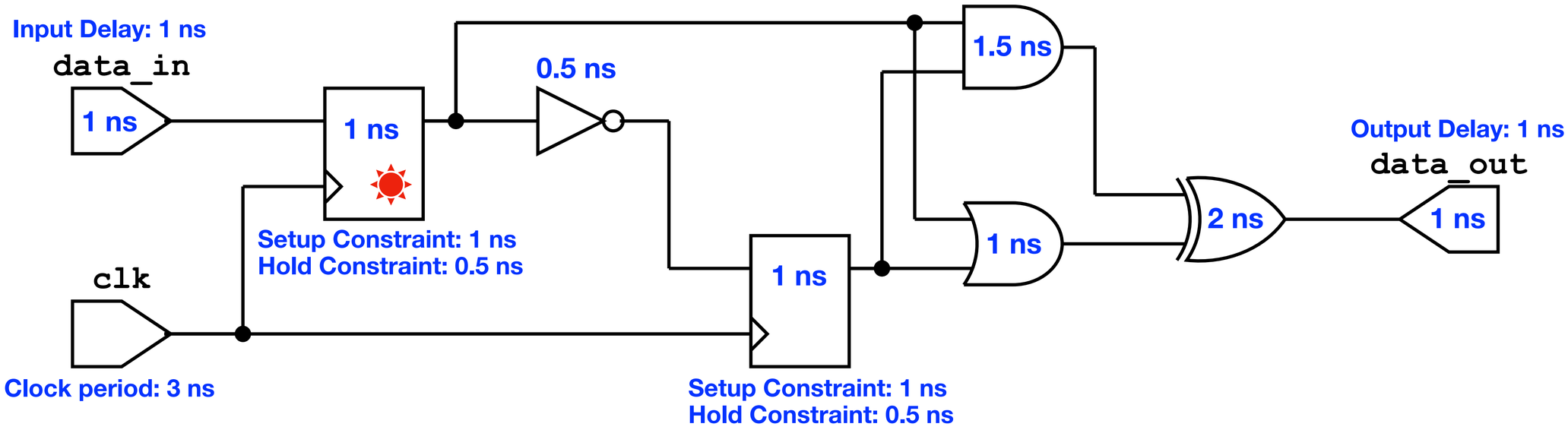
An example



Question: What is the hold required time?

Static Timing Analysis

An example

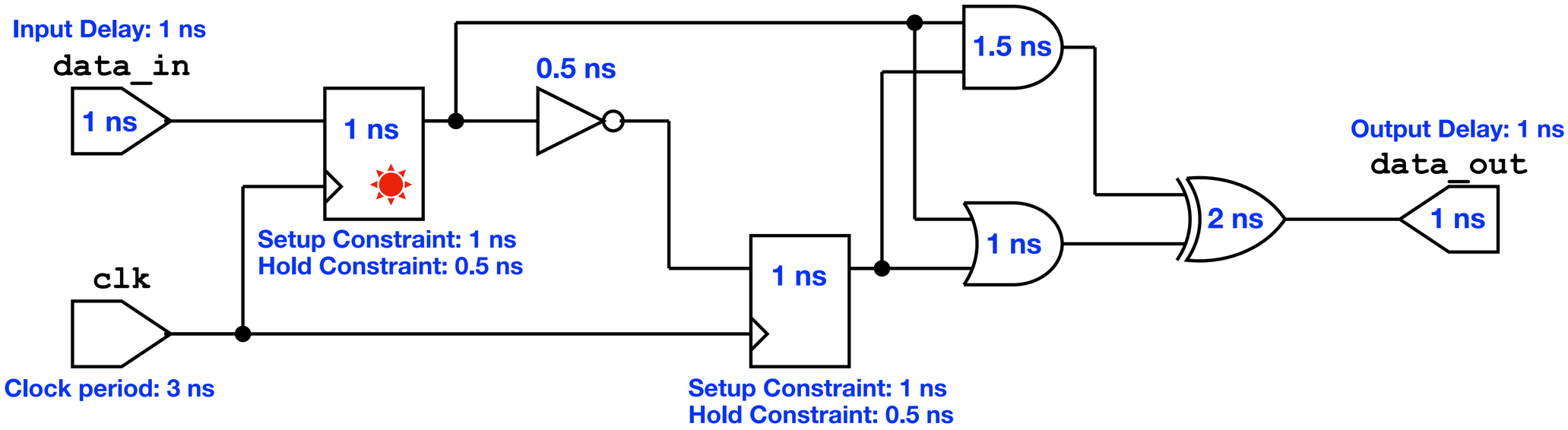


Question: What is the hold required time?

Answer: t_hold = 0.5 ns

Static Timing Analysis

An example

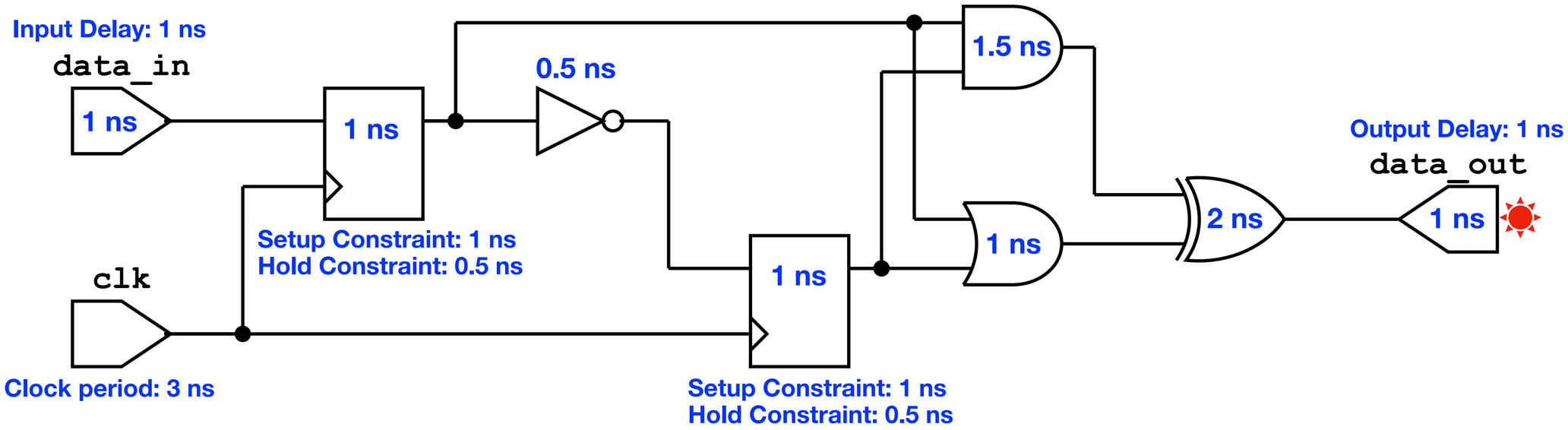


Question: What is the hold arrival time (fastest path)?

Answer: 1 ns. Requirement was > 0.5 ns (MET)

Static Timing Analysis

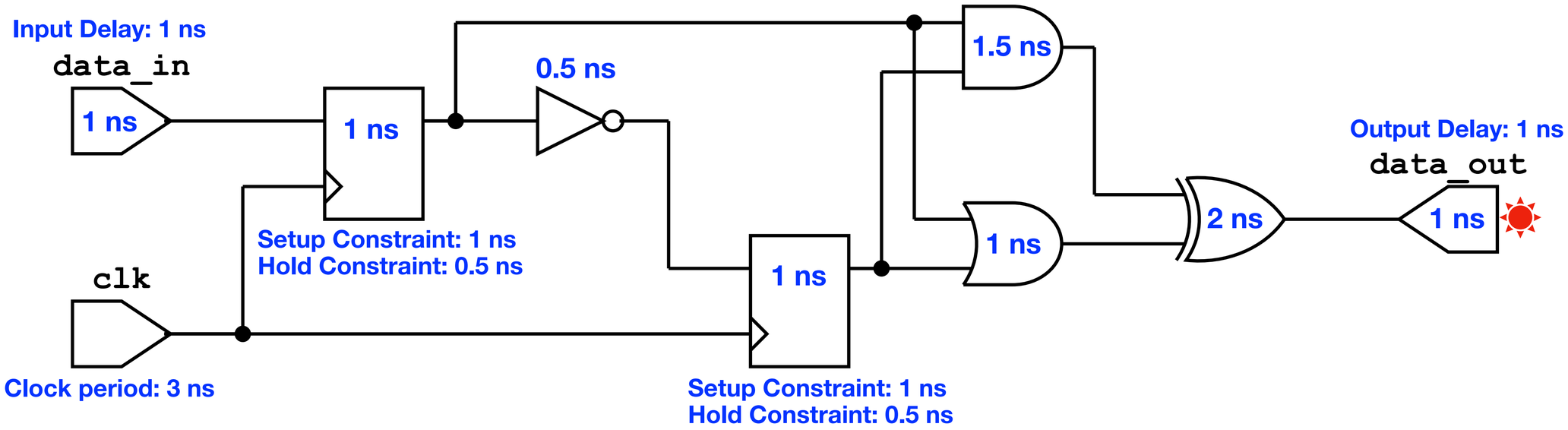
An example



Question: What is the setup required time?

Static Timing Analysis

An example

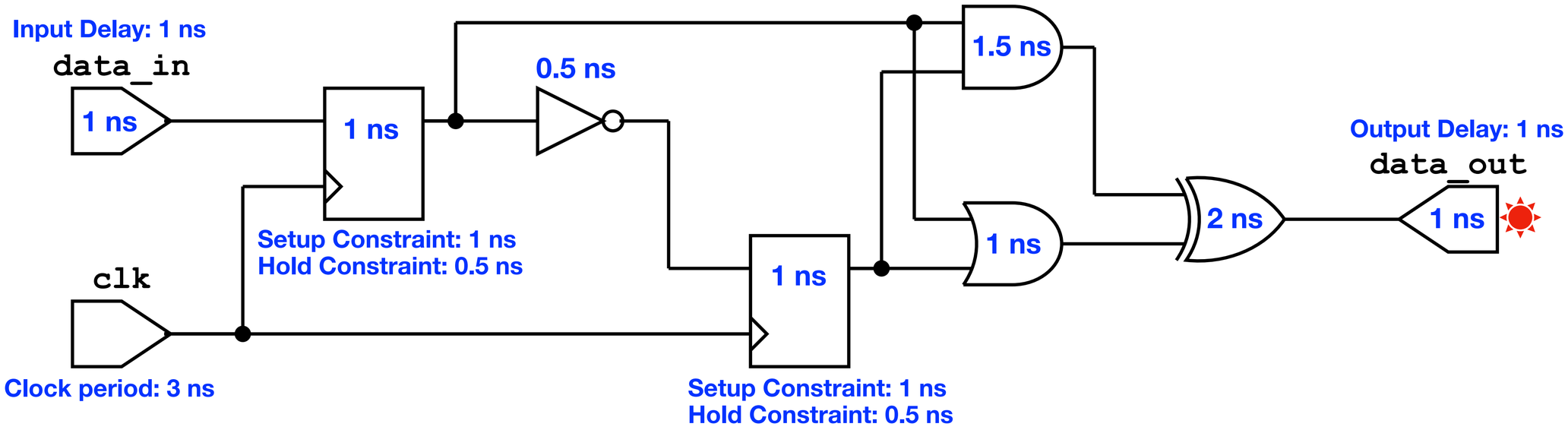


Question: What is the setup required time?

Answer: t_clk - t_output = 2 ns

Static Timing Analysis

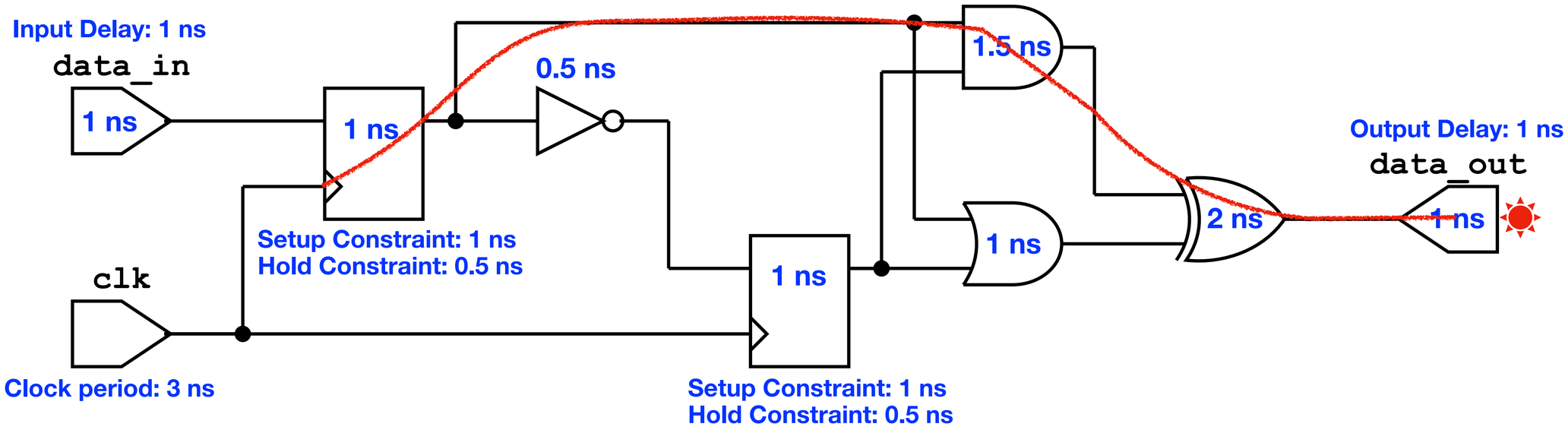
An example



Question: What is the setup arrival time?

Static Timing Analysis

An example



Question: What is the setup arrival time?

Answer: 4.5 ns. Requirement was < 2 ns (VIOLATED)

Synopsys Design Constraints (SDC)

Most important TCL commands

- **Clock creation**
  - create_clock
  - create_generated_clock
- **Input/output constraints**
  - set_input_delay
  - set_output_delay
  - set_input_transition
  - set_output_transition
  - set_driving_cell
  - set_load
- **Case analysis and timing exceptions**
  - set_case_analysis
  - set_false_path/set_disable_timing
  - set_multicycle_path
  - set_min_delay

So how are timing arcs actually calculated?

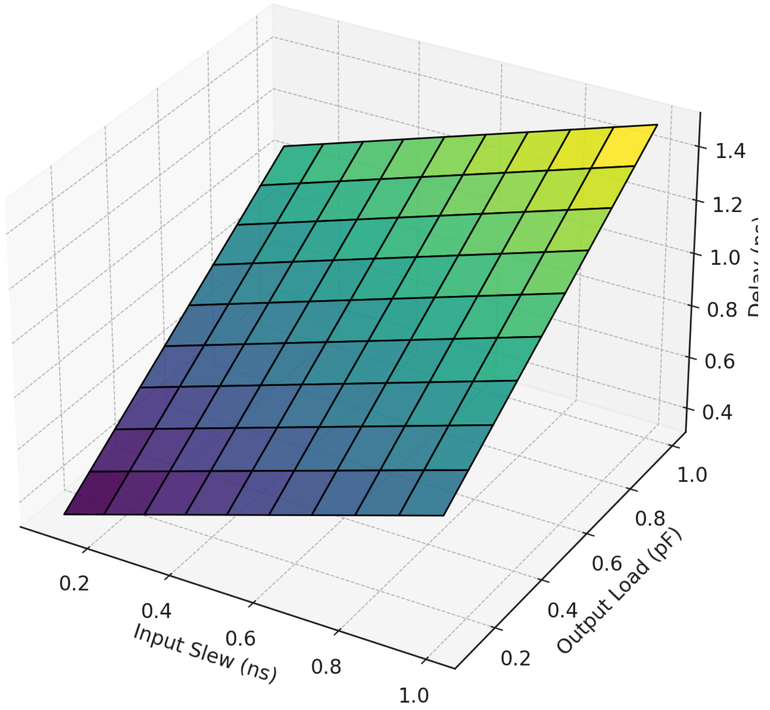
Three models: Constant, NLDM, CCS

- Constant: no dependence on inputs/outputs
  - Inaccurate, has to be very pessimistic to be useful
- Non-linear delay model (NLDM): 2D load-slew LUT interpolation
  - Reasonably accurate for early timing estimates
  - Should not be used for sign-off with < 45 nm technology
- Composite current source (CCS): driver/receiver I-t waveform LUT
  - Quite complex; huge files with lots of data, but near SPICE-level accuracy

Non-linear delay model (NLDM)

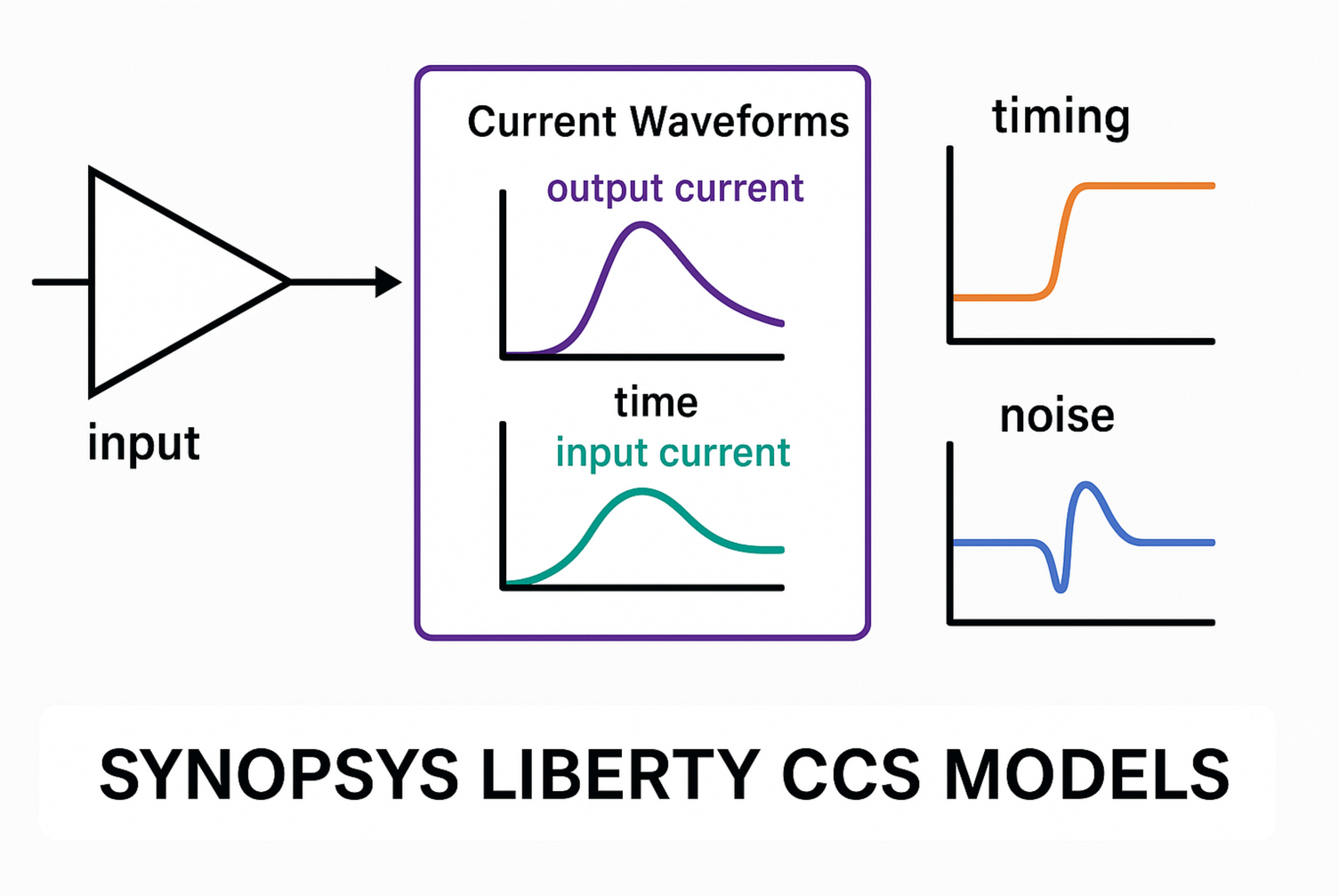
Used for fast but coarse timing estimates

- Delay = F(Input Slew, Output Load Cap)
  - F is interpolation of 2D LUT
- Output Slew = G(Input Slew, Output Load Cap)
  - G is interpolation of 2D LUT
- Does not account for Miller capacitance
  - Effective input cap increase
- Slew rates are not the full story, full shape is



Composite current source (CCS)

Used for accurate timing analysis



An analogy
Difference between NLDM/CCS

- You're trying to predict how fast a bucket fills
- NLDM: You’re told “it takes 5 seconds to fill with faucet set to medium”
- CCS: You’re given a graph showing exact water flow rate over time (starts slow, increases, then levels off).
- Second one (CCS) is much more accurate
    - Especially when the pipe pressure (voltage), bucket size (load), or faucet knob (input slew) varies

OpenSTA

And other (runner-up) FOSSi STA tools

- Three main open-source STA tools:
  - OpenSTA (C++, GPLv3 license, used by OpenROAD)
  - iSTA (C++, Mulan-PSLv2 license, used by iEDA)
  - OpenTimer (C++, MIT license, not maintained/used much anymore...)
- All are designed to be bolted onto other tools

OpenSTA Commands

Most important ingestion and reporting TCL commands

- **Ingestion**
  - read_liberty
  - read_verilog
  - read_sdc
  - read_spef
- **Setup/Hold Reporting**
  - group_path
  - report_checks
  - report_units
  - report_tns
  - report_wns

OpenSTA Demo

```
Startpoint: reg2_reg (rising edge-triggered flip-flop clocked by clk)
Endpoint: data_out (output port clocked by clk)
Path Group: clk
Path Type: max

        Delay          Time   Description                          Src Attr
---------------------------------------------------------------------------------------------------------------------
   0.00000000    0.00000000   clock clk (rise edge)
   0.00000000    0.00000000   clock network delay (ideal)
   0.00000000    0.00000000 ^ reg2_reg/CLK (dff_cell)               /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:31.12-34.8
   1.00000000    1.00000000 v reg2_reg/Q (dff_cell)                /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:31.12-34.8
                            reg2 (net)                           /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:31.12-34.8
   0.00000000    1.00000000 v i4/B (and_cell)                      /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:23.20-23.31
   1.50000000    2.50000000 v i4/Y (and_cell)                      /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:23.20-23.31
                            and_out (net)                        /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:23.20-23.31
   0.00000000    2.50000000 v i7/A (xor_cell)                      /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:28.23-28.39
   2.00000024    4.50000000 ^ i7/Y (xor_cell)                      /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:28.23-28.39
                            data_out (net)                       /Users/akashlevy/Documents/OpenSTA/latchup/latchup.sv:28.23-28.39
   0.00000000    4.50000000 ^ data_out (out)
                 4.50000000   data arrival time

   3.00000000    3.00000000   clock clk (rise edge)
   0.00000000    3.00000000   clock network delay (ideal)
   0.00000000    3.00000000   clock reconvergence pessimism
  -1.00000000    2.00000000   output external delay
                 2.00000000   data required time
---------------------------------------------------------------------------------------------------------------------
                 2.00000000   data required time
                -4.50000000   data arrival time
---------------------------------------------------------------------------------------------------------------------
                -2.50000024   slack (VIOLATED)

```

Recent OpenSTA contributions

Latest upstreaming activity from Silimate team

- RTL source attribution
- Advanced SDC filter expressions for get_*
- CentOS/RHEL7 Dockerfile support
- Brewfile support
- JSON reporting for checks/units
- Several minor bug fixes and performance improvements
- **File issues to: https://github.com/parallaxsw/OpenSTA/**

Related topics not covered here

Advanced, but worth learning about!

- Parasitics: SPEF files and how these affect delay calculations
- Library characterization: how Liberty files get stuffed using SPICE simulations
- Clock non-idealities: clock trees, clock skew (+ useful skew), clock uncertainty, jitter, etc.
- Max load and max transition constraints
- Multi-clock: clock domain crossings (CDC)
- Resets: recovery/removal checks, reset domain crossings (RDC)
- Noise modeling: CCSN
- Timing corners: PVT, MMMC
- Lots more...

Thank you!

Silimate is hiring: https://www.silimate.com

- Building the co-pilot for chip designers
- We love, use, and support open-source EDA (OpenSTA, Yosys, etc.)
- Well-funded and backed by Y Combinator
- Used by companies like SiFive, Tenstorrent
  - And more companies we can't disclose yet...
- Come chat with me!
  - Also, my email is akash@silimate.com

SILIMATE