Static Timing Fundamentals

Jason Rziha
Microchip Technology

March 22-23, 2017
Silicon Valley

Agenda

Introduction

The Fundamentals of Static Timing

Real World Examples

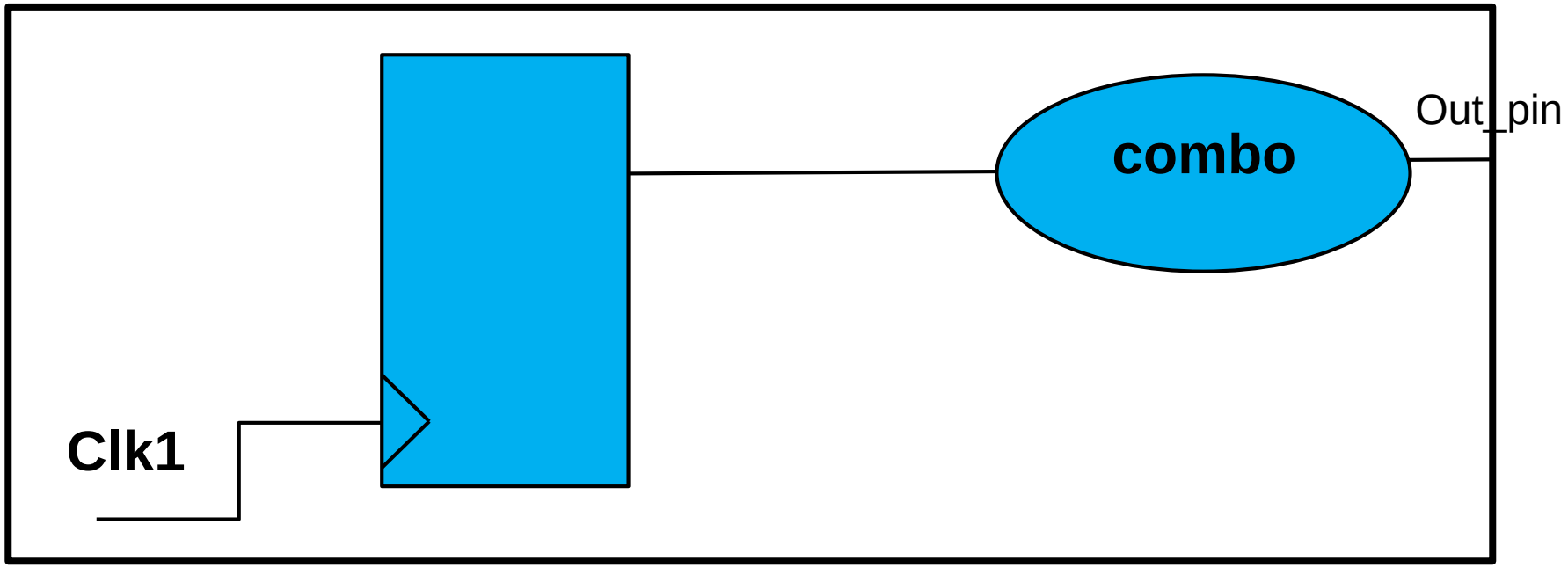
Conclusion

Introduction

- Found many people don’t really understand the fundamentals of STA
  - This included engineers with many years of experience
  - Often had to explain things, including why the tool behaved in a certain way. The fundamentals make this possible, even without intimate knowledge of the tool’s internals.

- Reduced it to the bare minimum
  - Understanding STA takes 2 fundamentals
  - Using it adds a 3rd fundamental

Do you understand the Fundamentals of STA?

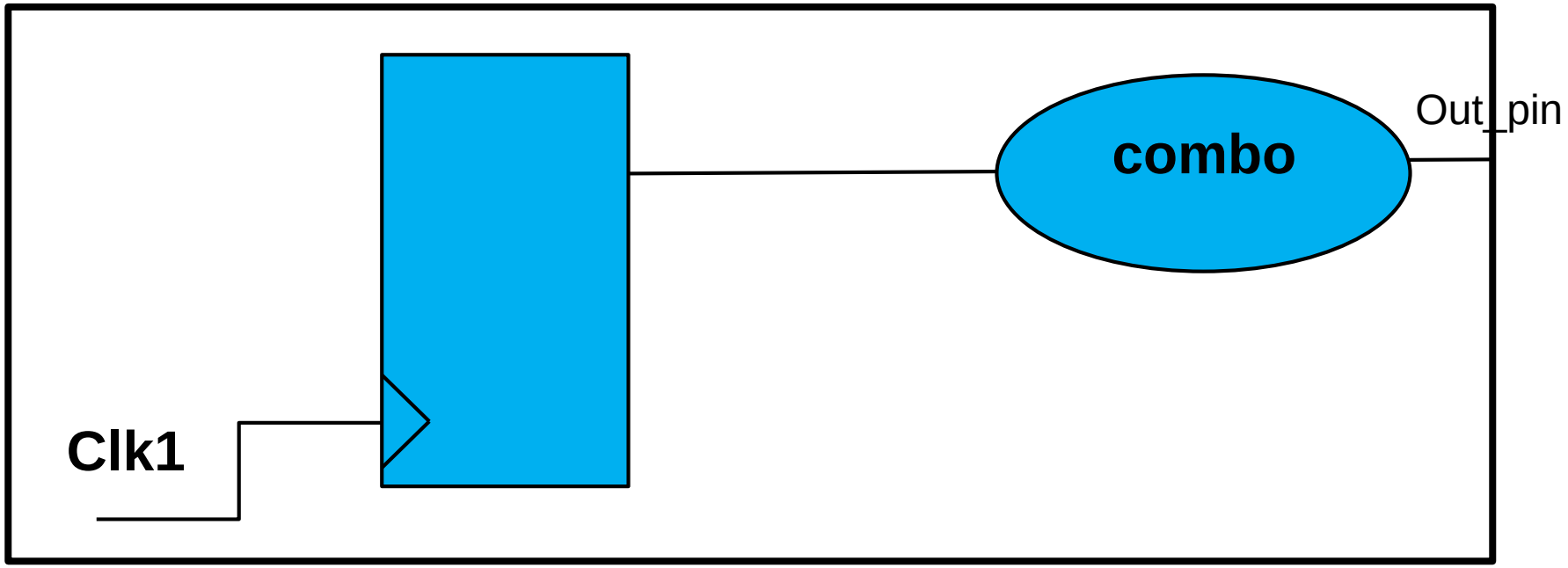


set_disable_timing <timing arc to out_pin>

report_timing –to <out_pin>

No Paths

Do you understand the Fundamentals of STA?

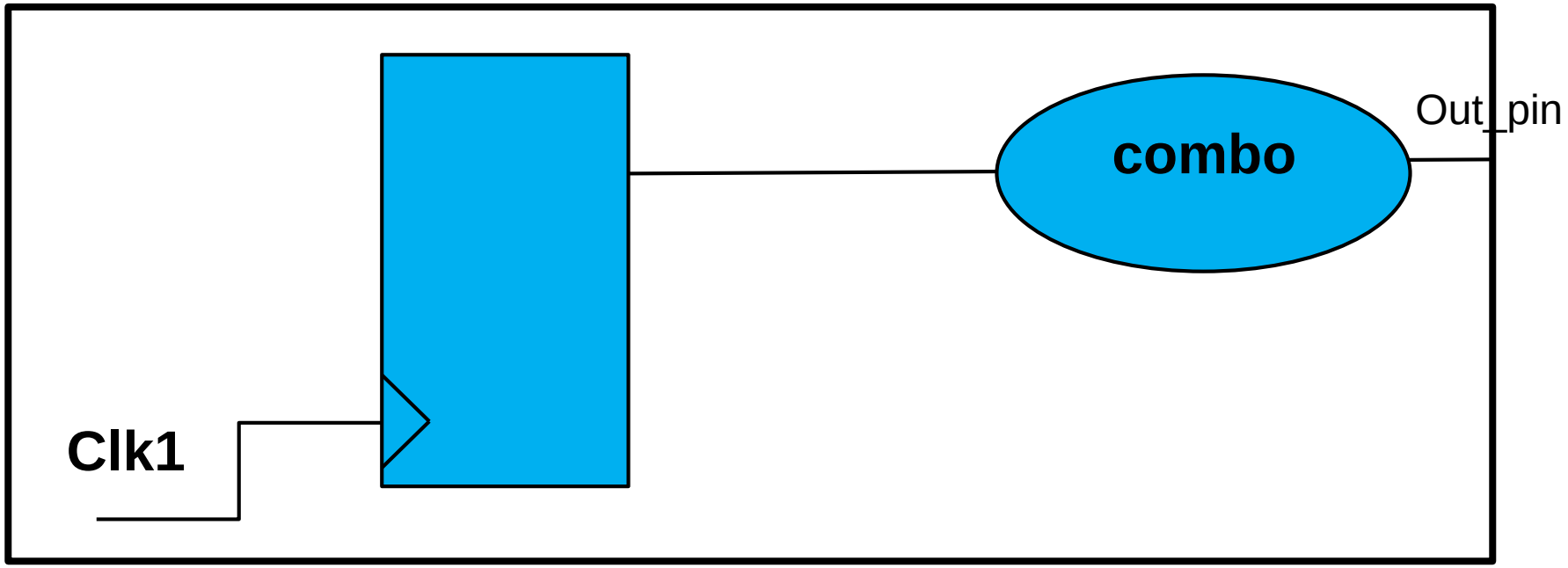


set_false_path –to <out_pin>

report_timing –to <outpin>

No Constrained Paths

Do you understand the Fundamentals of STA?



WHY?

STA Is Easy

- So easy a caveman can do it.

STA Is Easy

- So easy your boss can do it.

This Isn’t a PrimeTime Paper

- Many people think STA=PrimeTime
- Other tools depend on STA
  - Design Compiler
  - ICC

- Developers who don’t use the tools directly still need to be aware of STA
  - Architects
  - Library developers

The Fundamentals of Static Timing

Fundamental #1: It's Graph Based

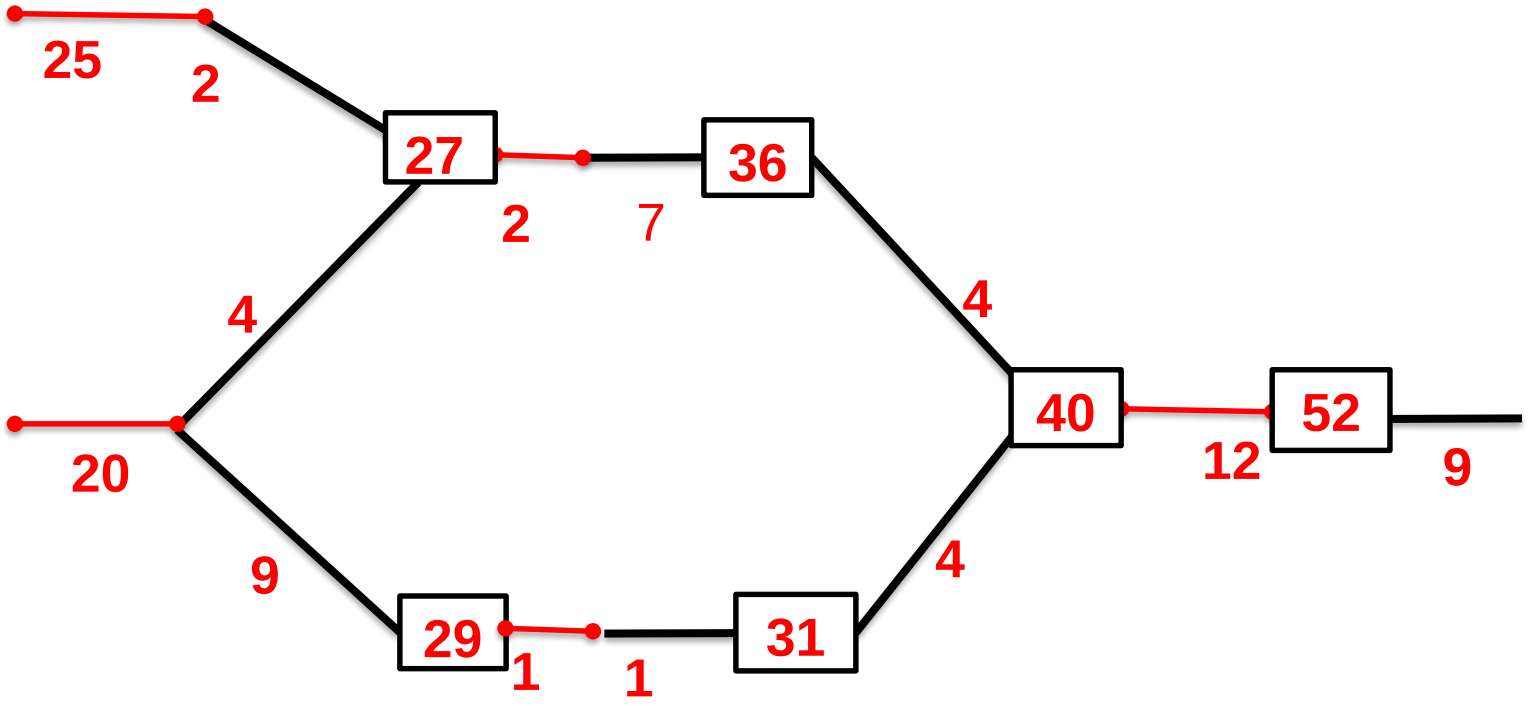
We ignore functionality, and simply treat the circuit as a graph

2 Input OR Gate:

2 Input AND Gate:

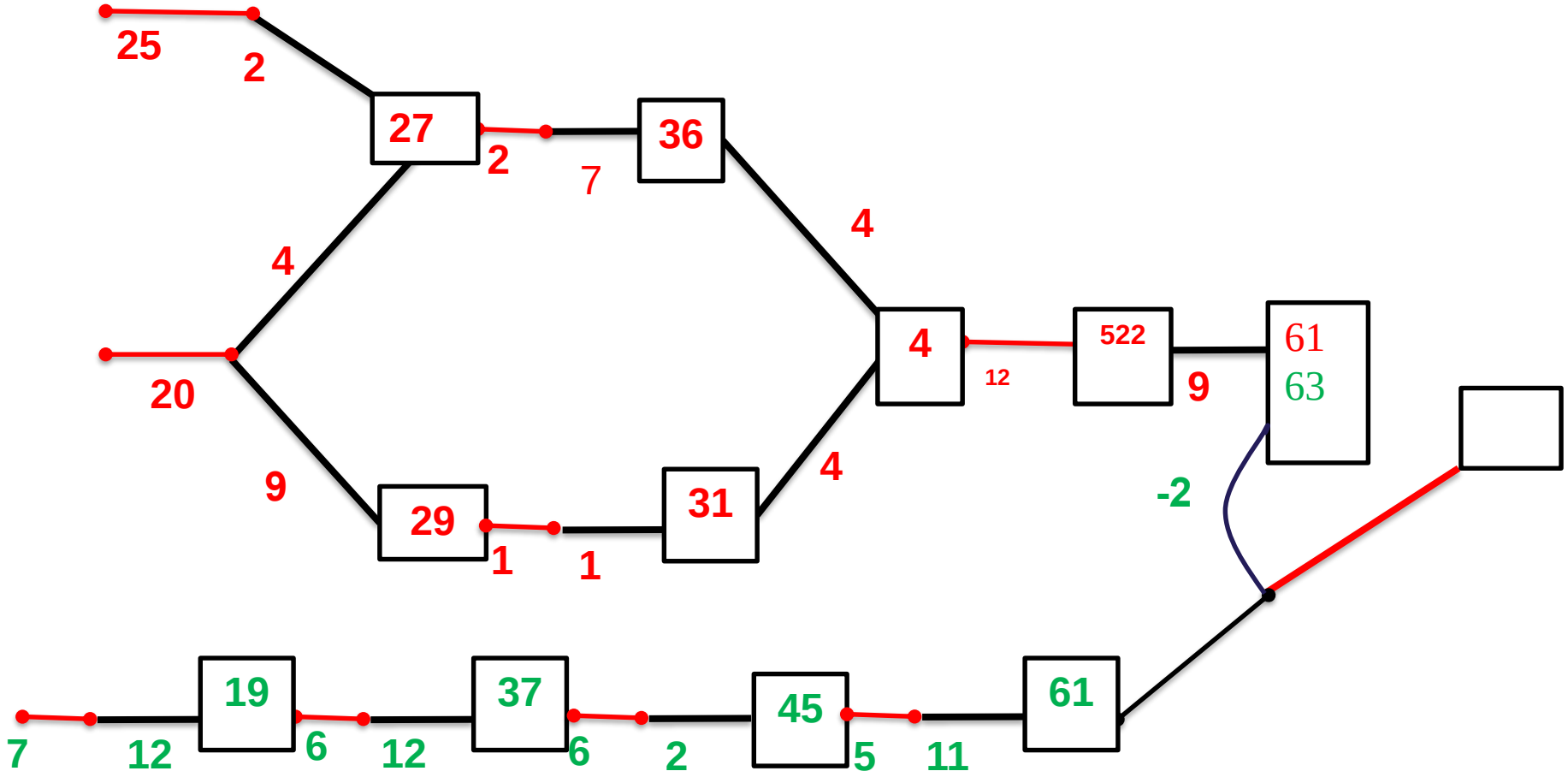
Apply simple arithmetic

We add up delays, picking the most pessimistic option whenever we have to choose.



Compare the delays

Finally, we compare the arrival and required times to determine if we met timing.

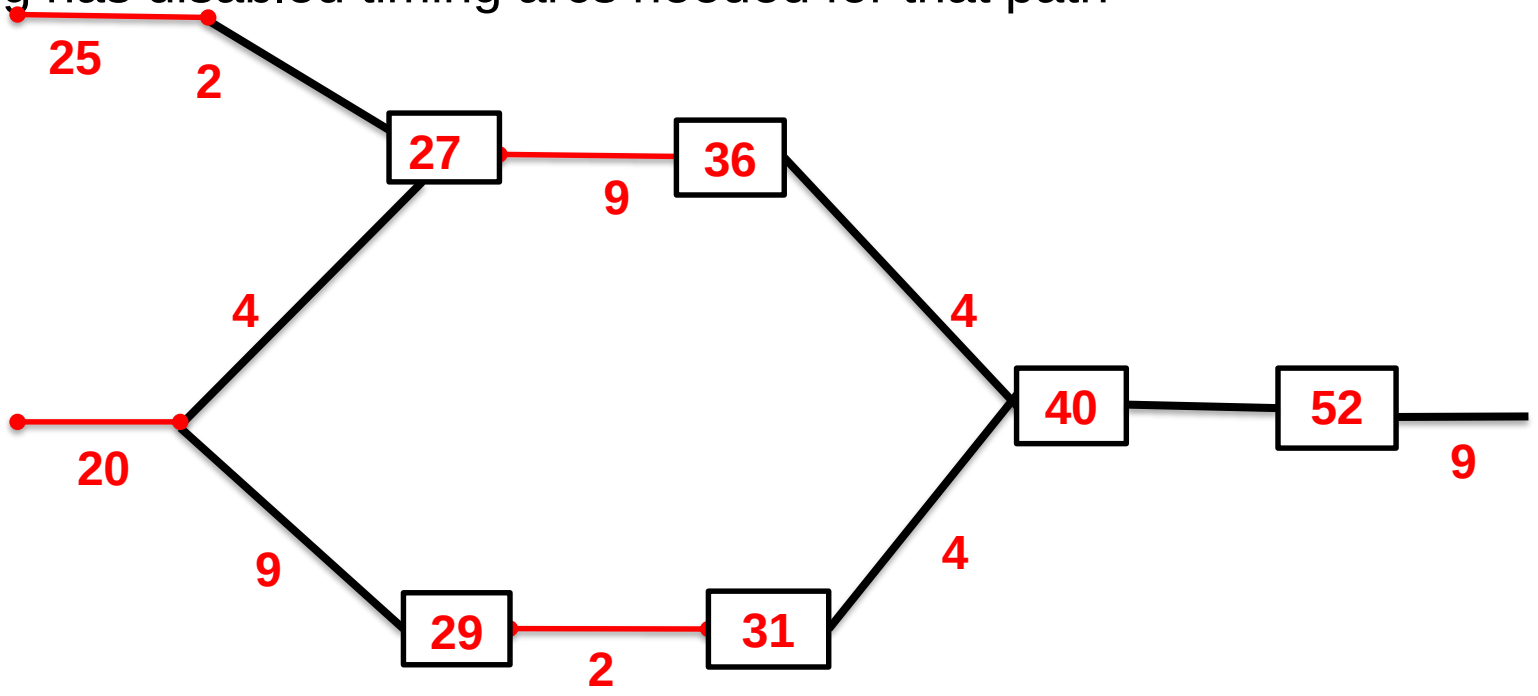


Impacts of Graph Approach (1/3)

- We don’t care about functionality
  - Many gates look the same to STA
  - This often has the biggest impact on designers who don’t directly use STA –they expect paths to not be analyzed as they wouldn’t functionally be used.
- When the tool says “No Paths”
  - This means it cannot walk from the startpoint to the endpoint of the path.
  - Prevents calculation of **arrival** time
  - Indicates the timing graph is "broken” somehow
- When the tool says “No Constrained paths”
  - This means the tool can walk from the startpoint to the endpoint of the path
  - Once there, we don’t have a required time to compare against

Impacts of Graph Approach (2/3)

- When the tool says “No Paths”
  - This means it cannot walk from the startpoint to the endpoint of the path.
  - Prevents calculation of **arrival** time
  - Indicates that either there’s no circuit that creates a path from the start to end points, or something has disabled timing arcs needed for that path

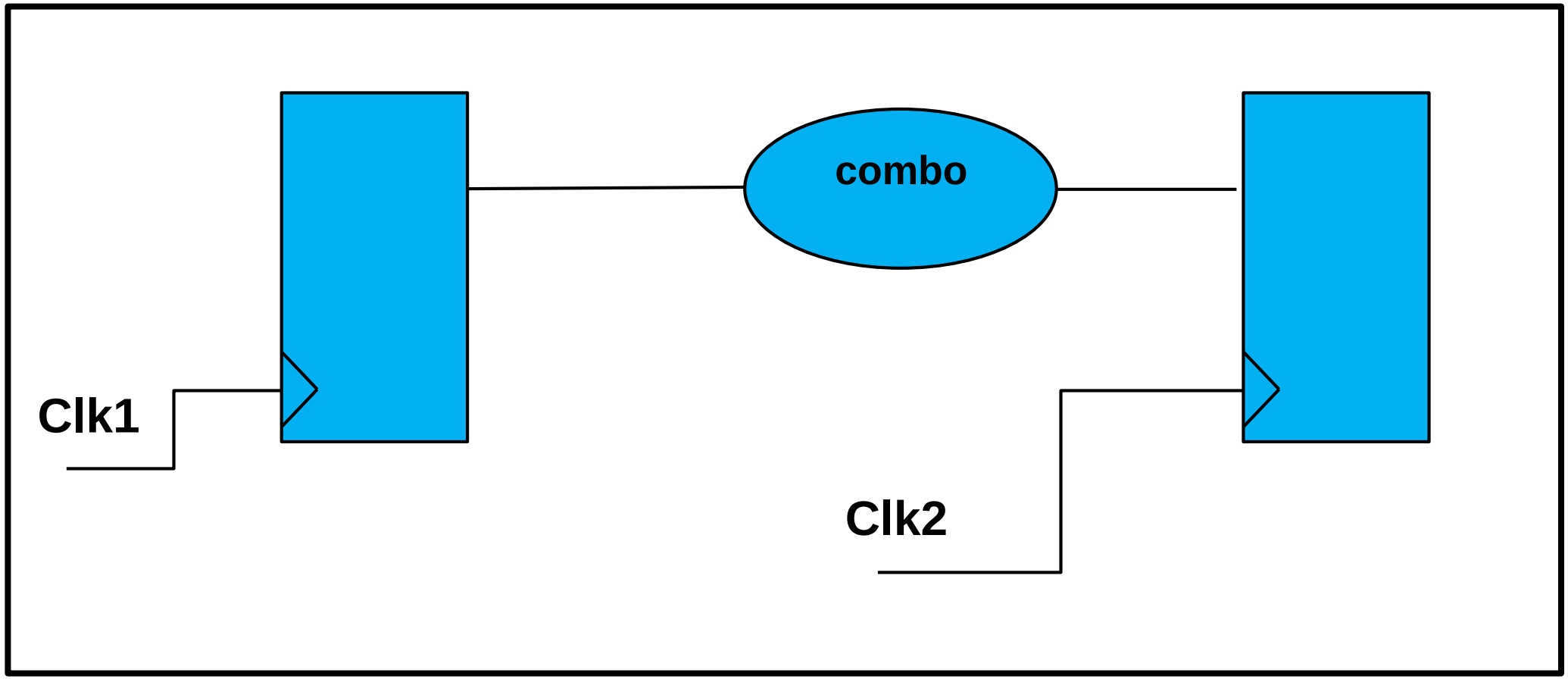


Impacts of Graph Approach (3/3)

- When the tool says “No Constrained paths”
  - This means the tool **can** walk from the startpoint to the endpoint of the path
  - Once there, we don’t have a required time to compare against
  - This has many possible causes, including
    - Broken capture path
    - Removed timing check at endpoint (set_disable_timing, set_false_path)

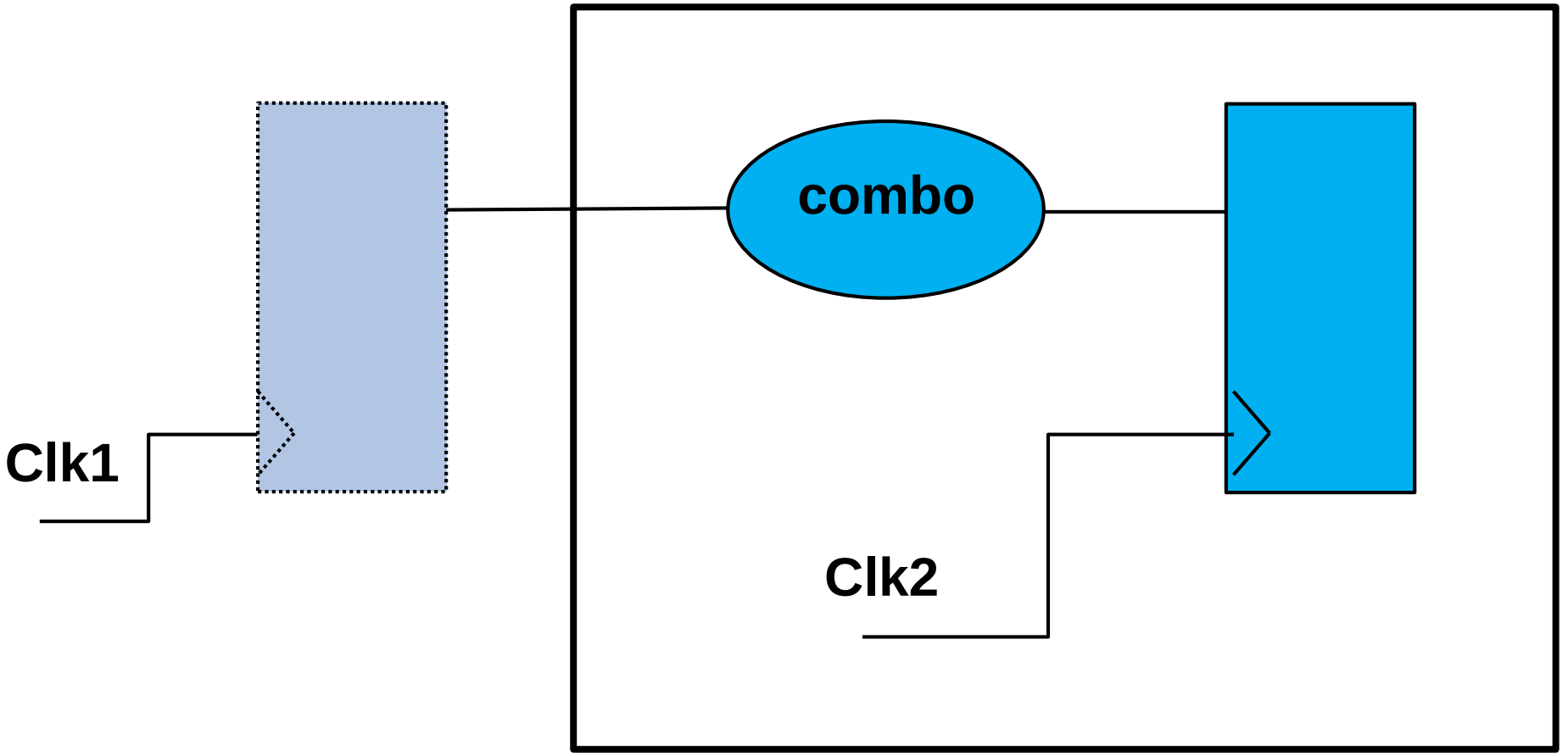
Fundamental #2: Every Path is the Same

We start with a launch flop, have a combinational cloud, and then a capture flop.



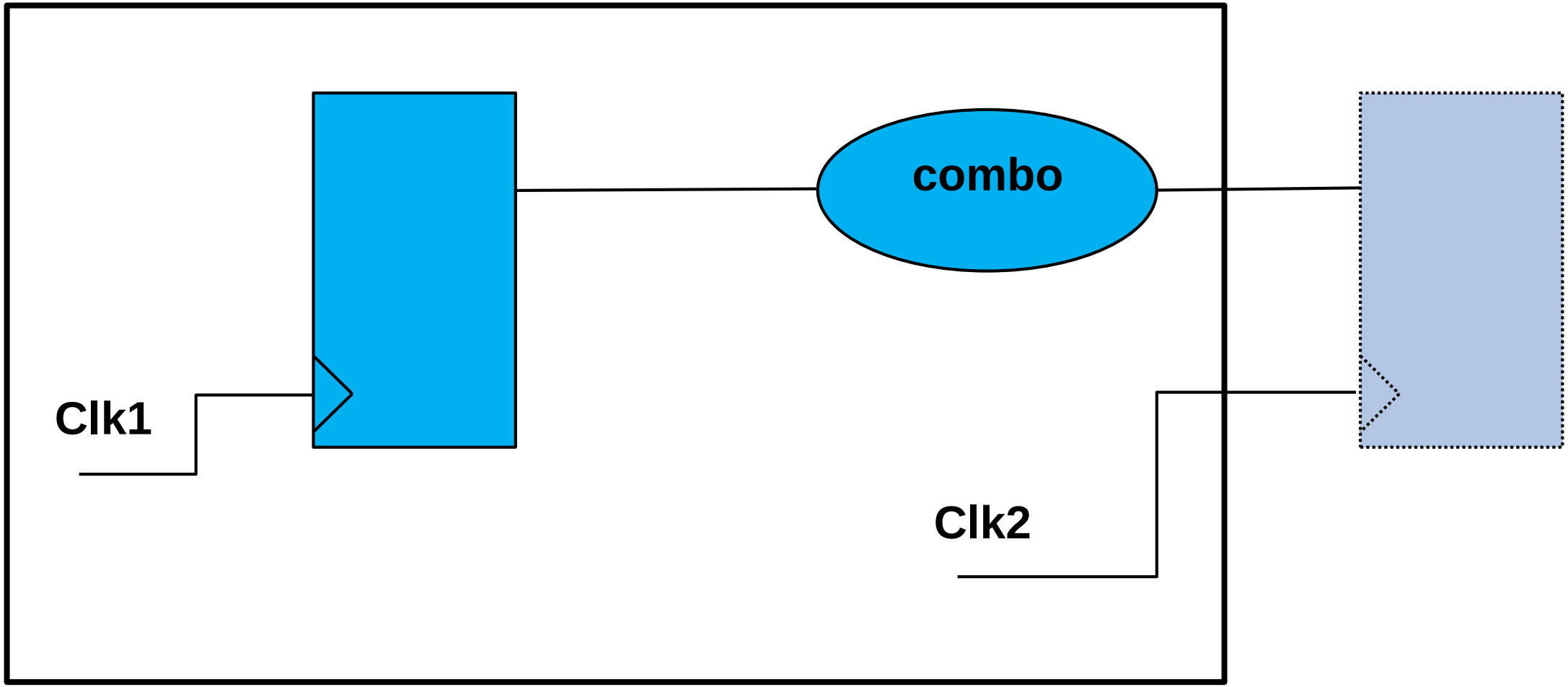
Every path is the same

For input paths, the launch flop is virtual:



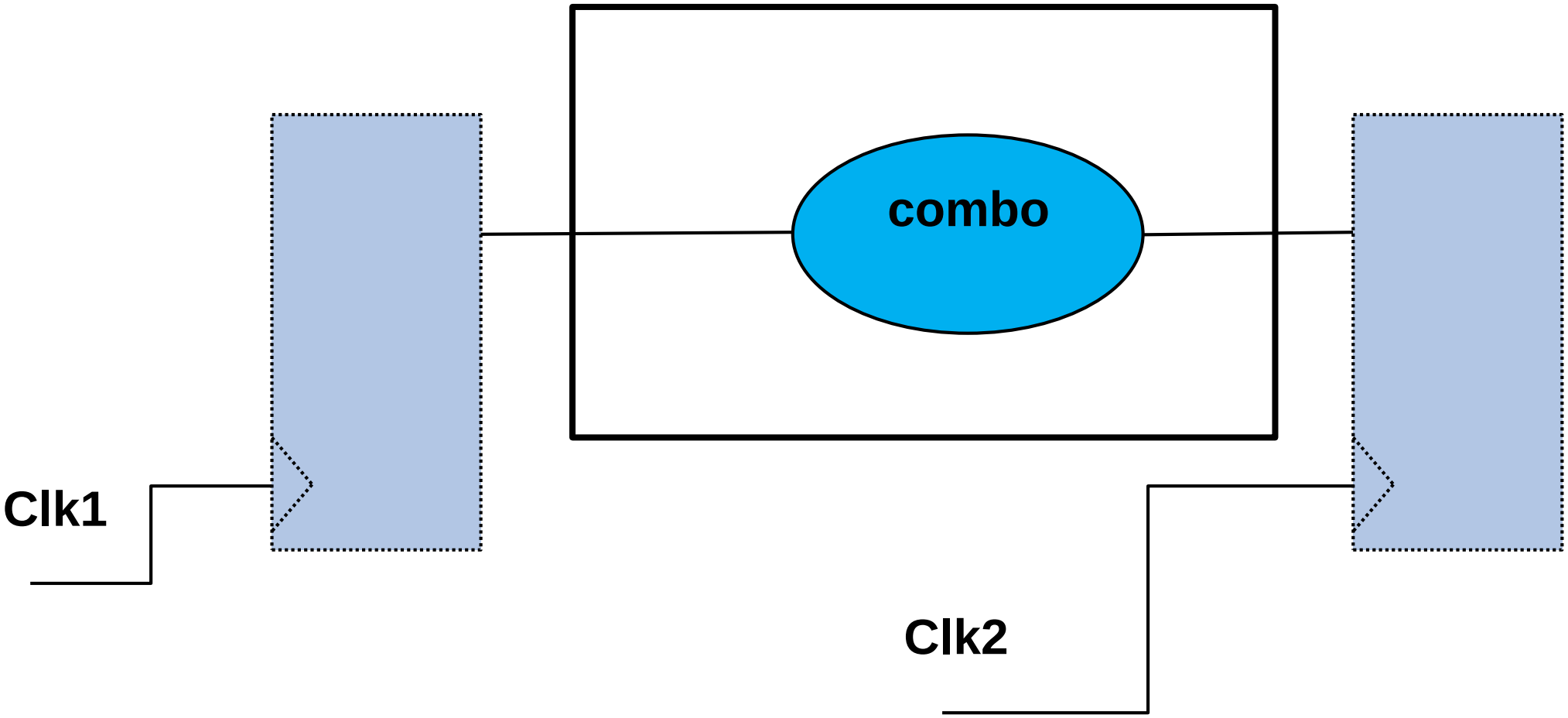
Every path is the same

For output paths, the capture flop is virtual.



Every path is the same

For feedthrough paths, both flops are virtual.



Fundamental #3: Startpoints and Endpoints

- Textbook Definition
  - Startpoint: Any input pin, and clock pins of sequential elements
  - Endpoint: Any output pin, and data pins of sequential elements

- Many engineers don’t know the textbook definition, although a few also know that clock gates become endpoints as well

- The textbook definition sometimes leaves us with cases we can’t handle- so the tools support much more.

Fundamental #3: Startpoints and Endpoints

- Supported Startpoints
  - Primary inputs
  - Clock pins of sequential elements
  - Clock gating cell clock pin
  - D inputs of a latch
  - Clock objects
  - Any pin with an input delay specified.

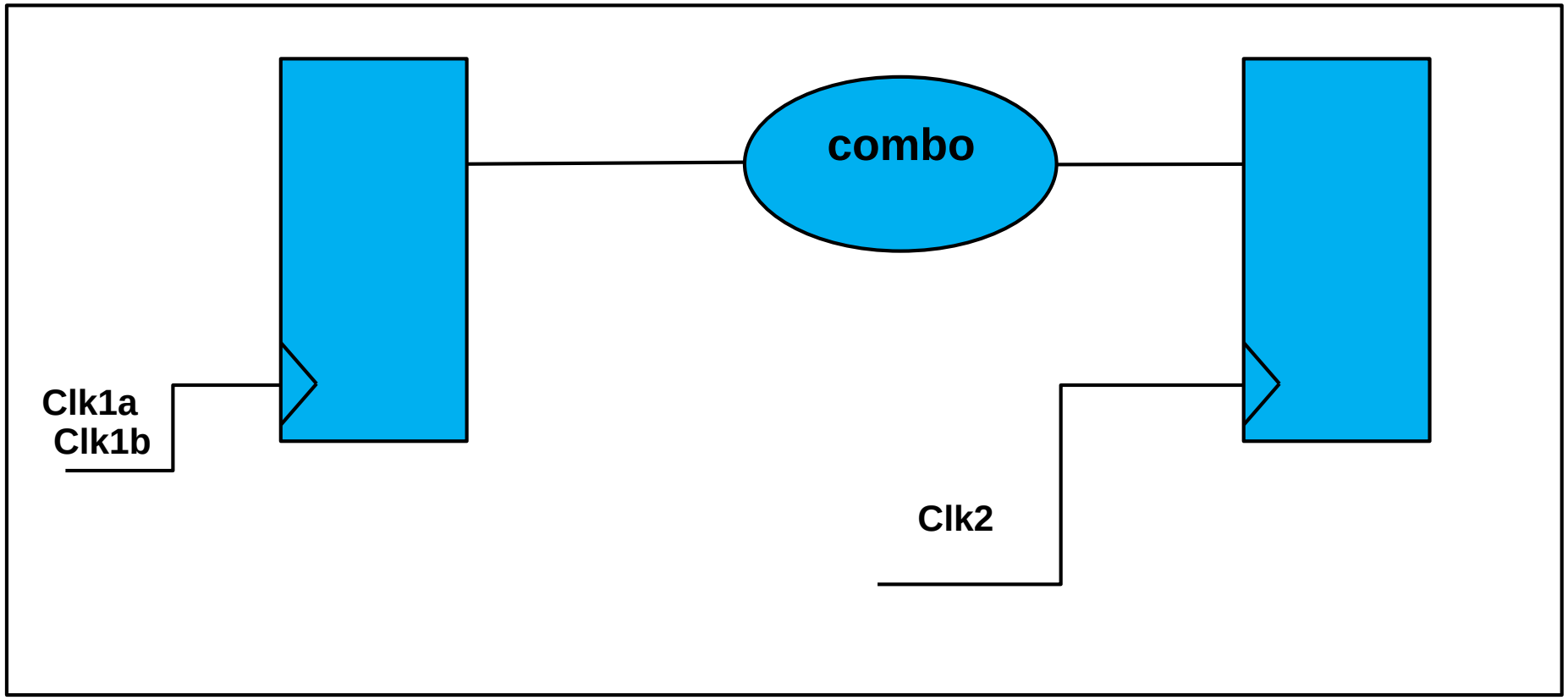
Fundamental #3: Startpoints and Endpoints

- Supported Endpoints
  - Primary outputs
  - Data pins of sequential elements
  - Clock gating cell data pin
  - Clock objects
  - Any pin with an output delay specified.

Example #1: Conditional Multi-Cycle Path

Example: Conditional Multi-Cycle Path

- User wanted to have a path from Reg1 to Reg2 be multicycle only when clocked by one clock- leaving paths clocked by other clocks single cycle



Example: Conditional Multi-Cycle Path

- With only the textbook definition of start/end points, this isn't possible.
- Using the clock pin of Reg1 as the startpoint would cause all paths from Reg1 to Reg2 to become multicycle

- The Solution is to apply the 3rd fundamental, and use the clock object as the startpoint.

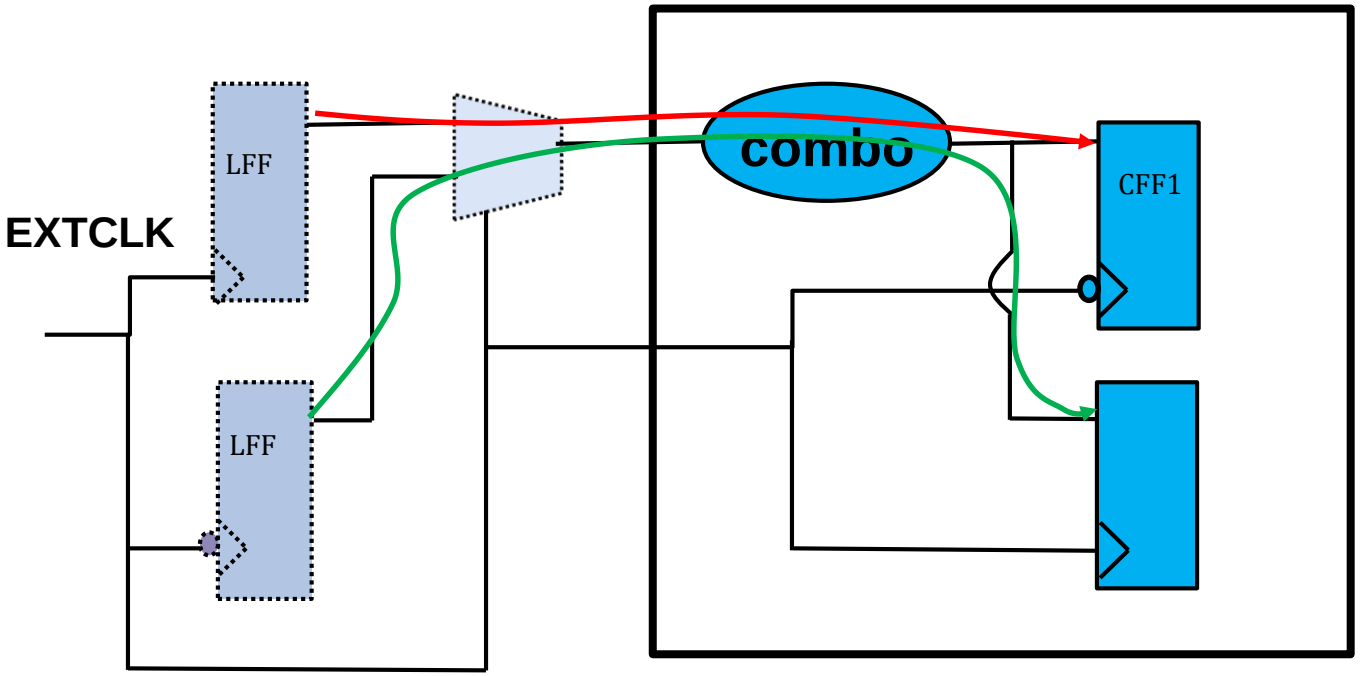
```
set_multicycle_path –from [get_clock <launch_clock>] –through \
 [get_pin launch_flop/Q] -to [get_pin <capture_register_data_pin>]
```

- Now the path is constrained as desired.

Example #2: Externally Clocked Interface

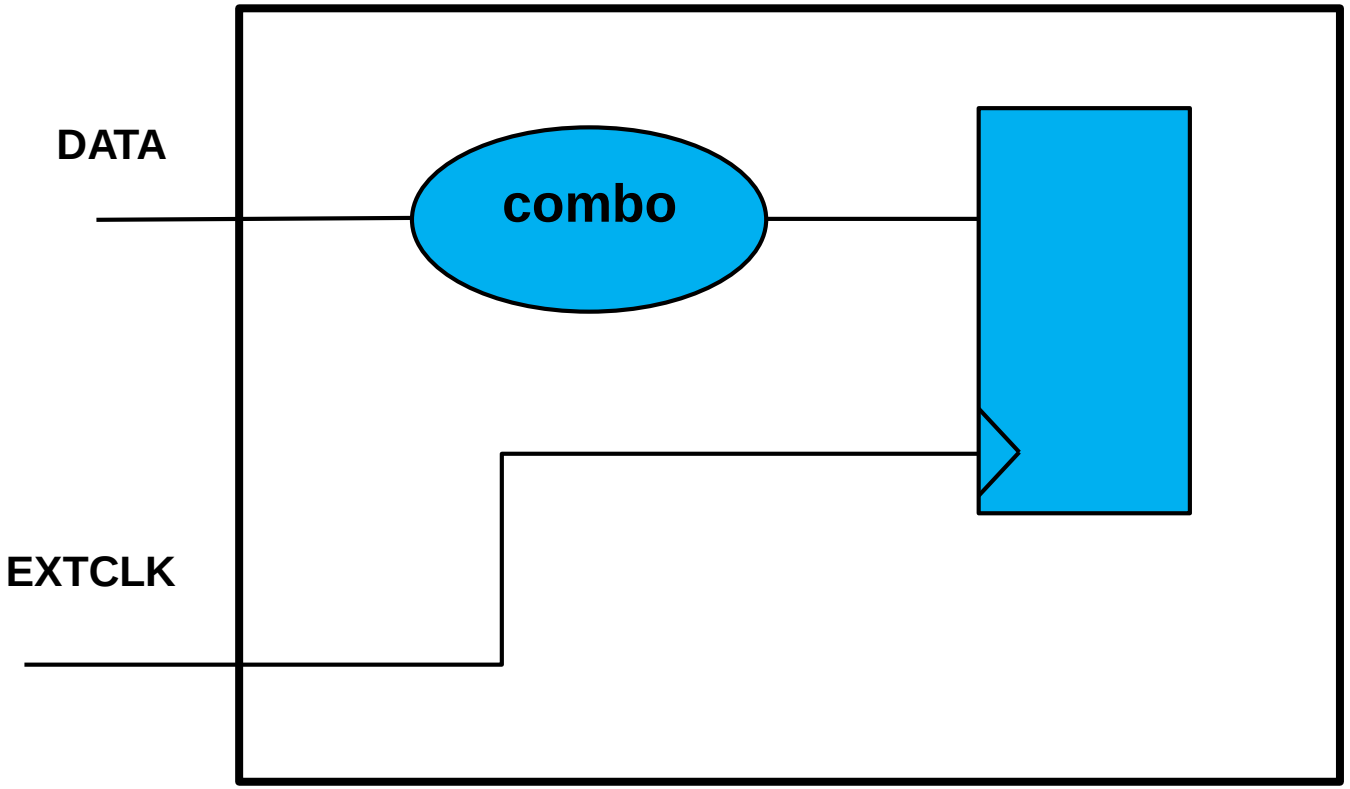
Example: Externally Clocked Interface

- An external interface was clocked on both edges of the clock
- Captured internally on the opposite edges.



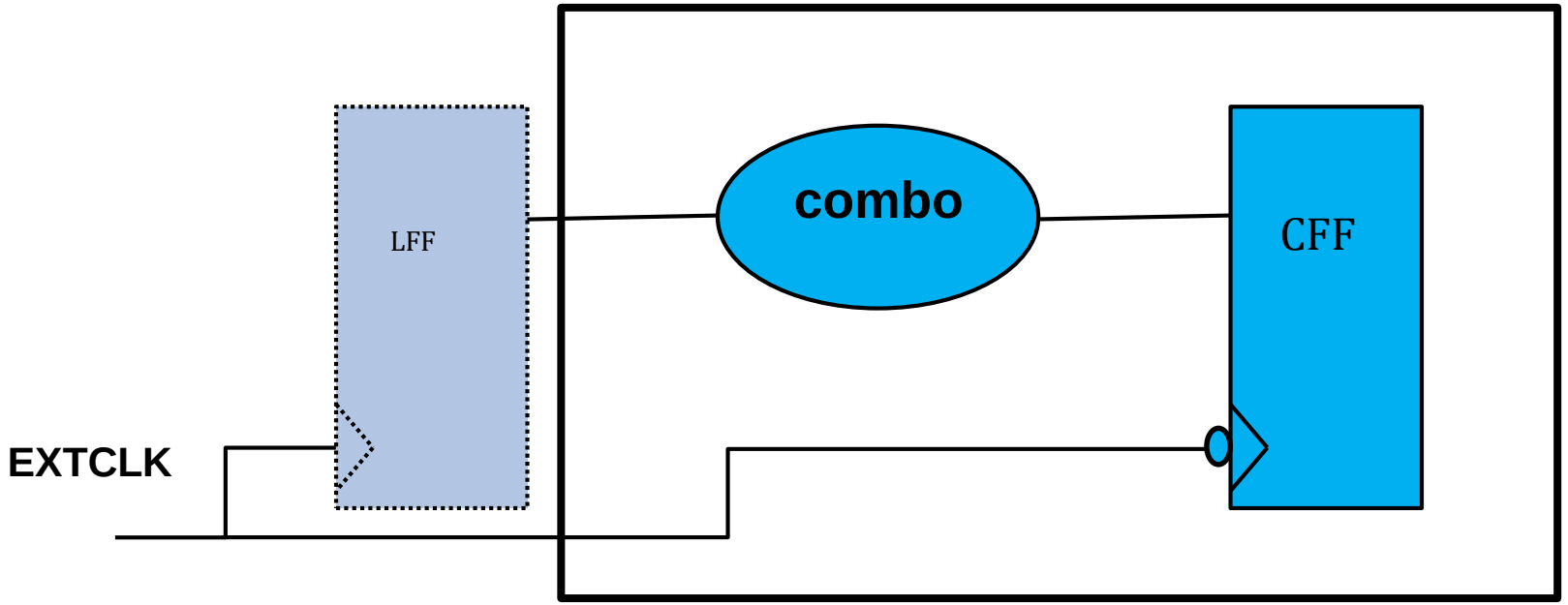
Example: Externally Clocked Interface

- The problem was that the engineer only considered the “real” circuit:



Example: Externally Clocked Interface

- Applying Fundamental #2 and #3 solve the problem.

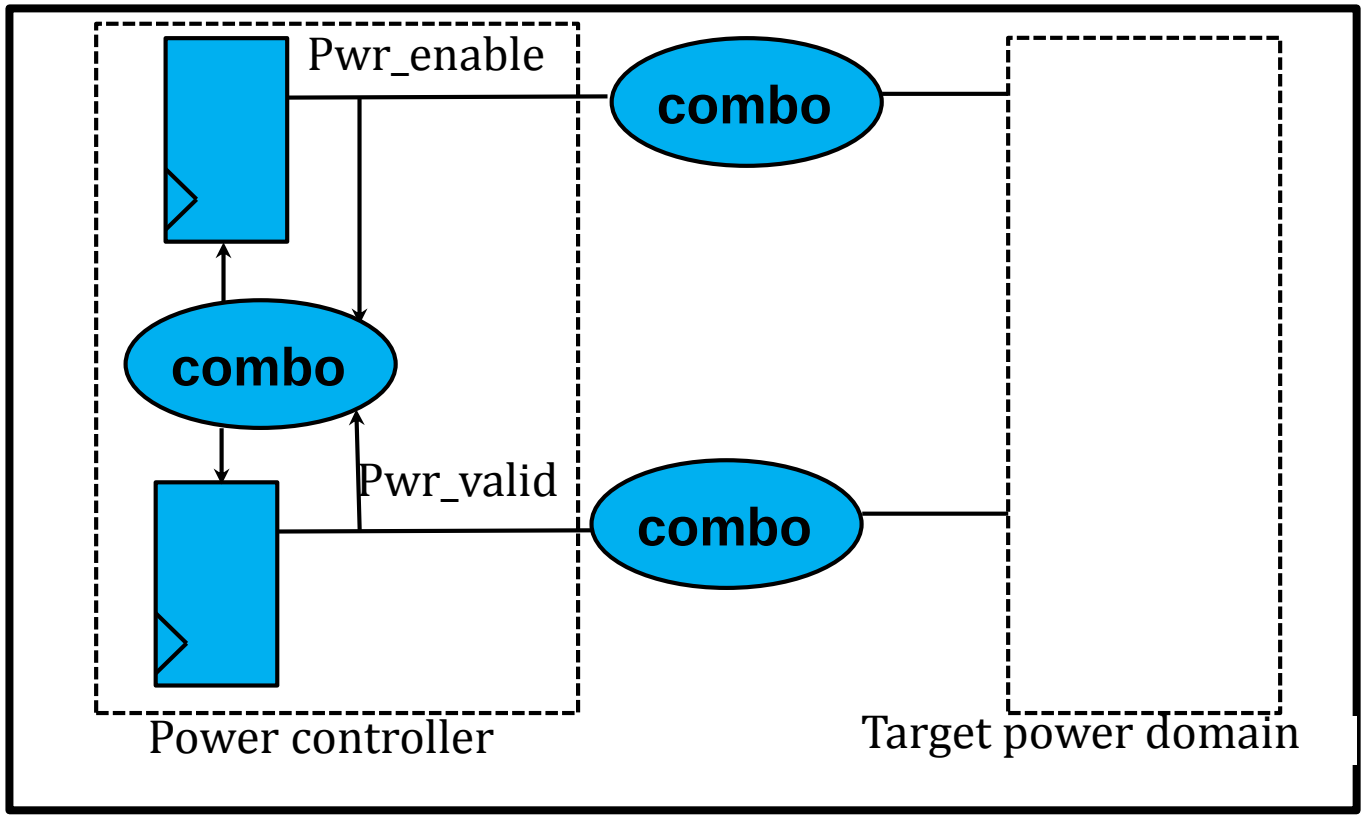



```
Report_timing -rise_from [get_clock <launch_clock>] -through \
 [get_pin input_pin] -fall_to [get_pin <capture_register_clock_pin>]
```

Example #3- Asynchronous Logic

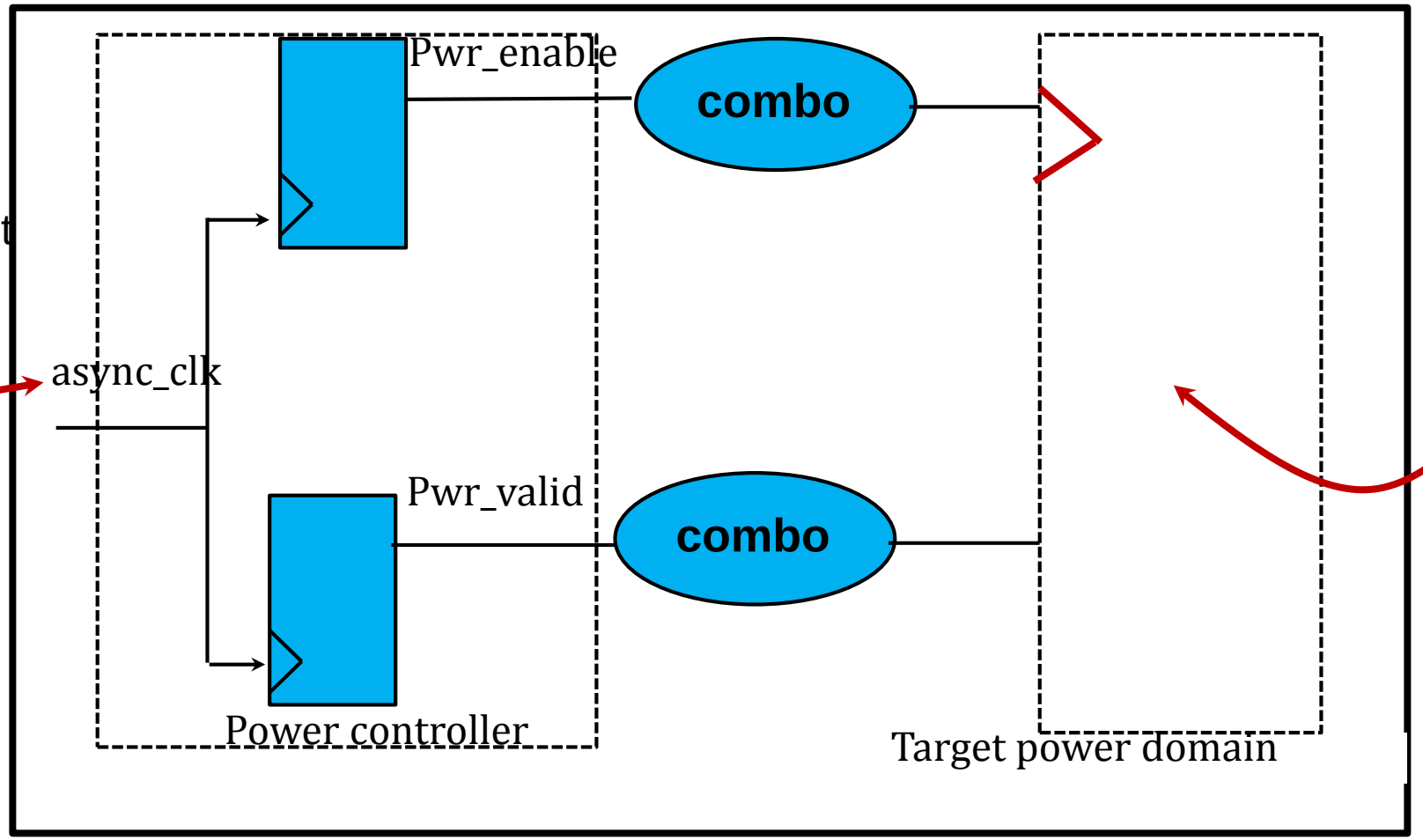
Example: Self Timed Logic

- Creative Engineers designed a self-timed power controller
- Then they wanted to verify that the enable/valid signals preserved their phasing at the target domain



Example: Self Timed Logic

- This can be verified with STA… if we apply Fundamental #2



Example: Self Timed Logic

```
set_clock_groups –asynchronous –group [get_clock async_clock] \
-group [ remove_from_collection [all_clocks] [get_clock async_clock]

set_data_check –setup –from [get_pin <pwr_valid pin> ] –to \
  [get_pin <pwr_enable_pin>]  <value>

set_data_check –hold –from [get_pin <pwr_valid pin> ] –to \
   [get_pin <pwr_enable_pin>]  <value>
```

Example #4: Implicit Clock Gating

Example: Implicit Clock Gating

- ICC said design was timing clean
- PrimeTime said the design had violations on the order of 1000ns

- Violation endpoints all looked like:

  …

  chip/block/block2/and3

  chip/block/block3/or2

  chip/block/block2/or3

  chip/block/block4/and2

  chip/block/block2/or3

  …

Example: Implicit Clock Gating

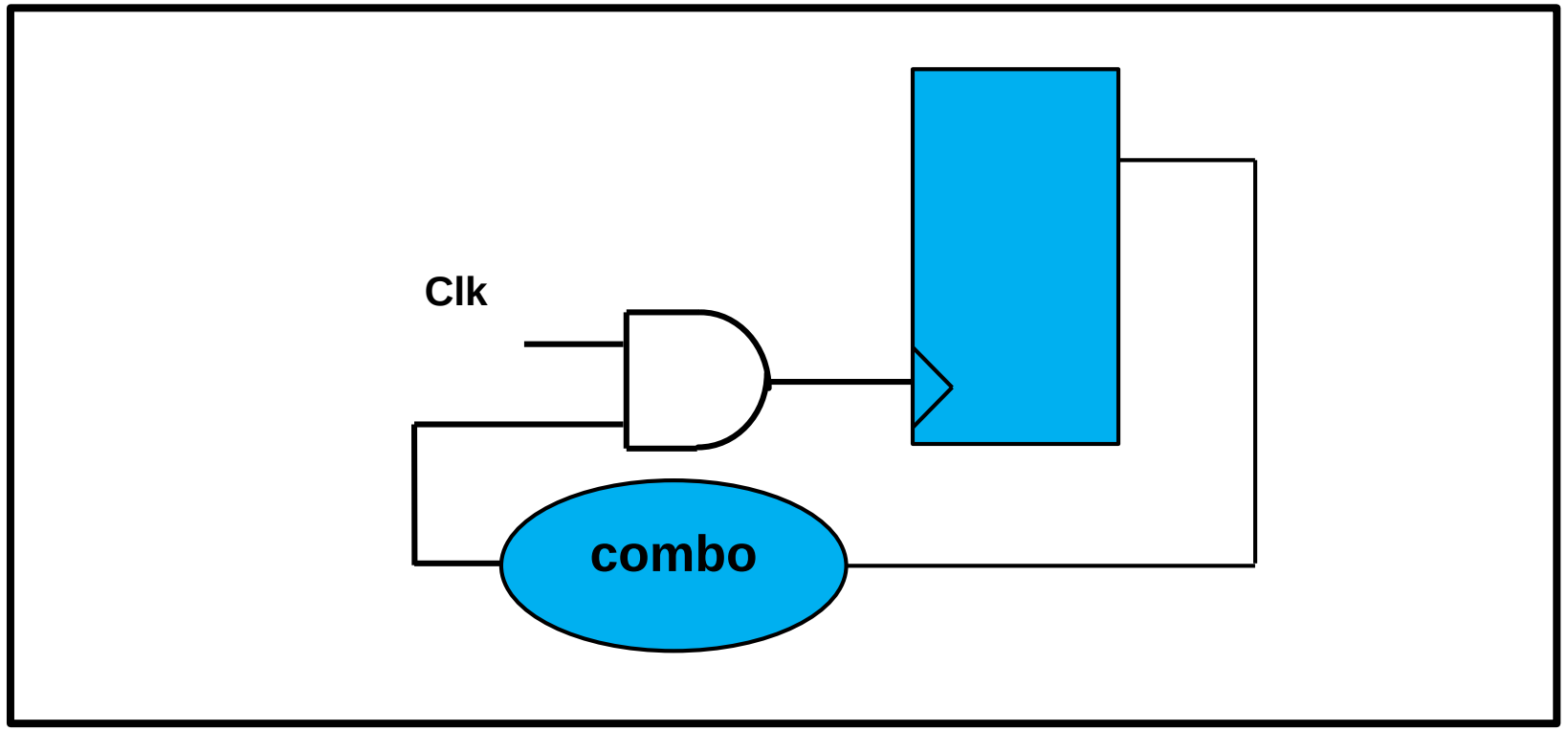
- Apply Fundamental #3 and go through the list of endpoints
- The only option is that PrimeTime decided these cells were clock gates

- PrimeTime was improperly inferring clock gating checks.
- A quick SolvNet search returned scripts for identifying and disabling the appropriate checks.

Example #5: Do you believe the tool

Example: Strange Tool Behavior

- PrimeTime said this path existed, ICC said “No Paths”



Example: Strange Tool Behavior

- The user believed the path existed – how to prove it?

- "No paths" means "You can't walk the graph from the startpoint to the endpoint".
  - all_fanin and all_fanout will walk a timing graph for you
  - Using all_fanout –from <startpoint> and searching for the endpoint showed we could walk the graph.
- One other possibility was that since a –group option was used, it was possible that the capture clock didn't reach the clock gate
  - Querying the clock attribute of the clock gate's clock pin showed it did
  - Using all_fanout –from <clock startpoint> showed that path was valid also
- The conclusion was that there was a problem with the tool

Conclusion

Static Timing analysis is Easy – If you remember the fundamentals

- #1: It's a graph based solution
- #2: Every path is the same
- #3: Know the start/end points

- Knowing these fundamentals will help you solve problems with any tool that uses Static Timing analysis

Thank You

Appendix

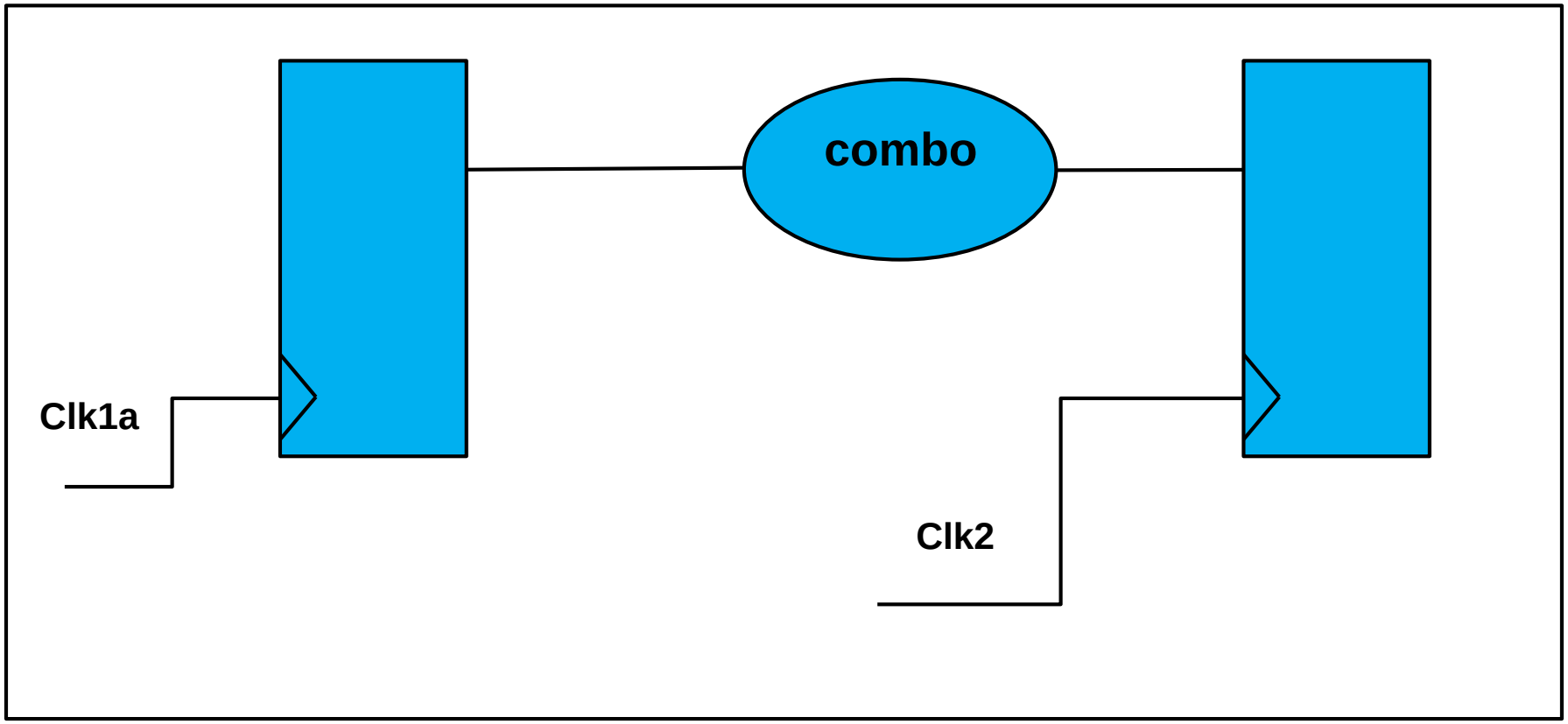
So What About Latches?

- Fundamental #2 says paths start at a Flop and end at a flop
- Fundamental #3 lists D inputs of a latch as a startpoint and an endpoint

- Why don’t these conflict?
  - Because latches are combinational and sequential.

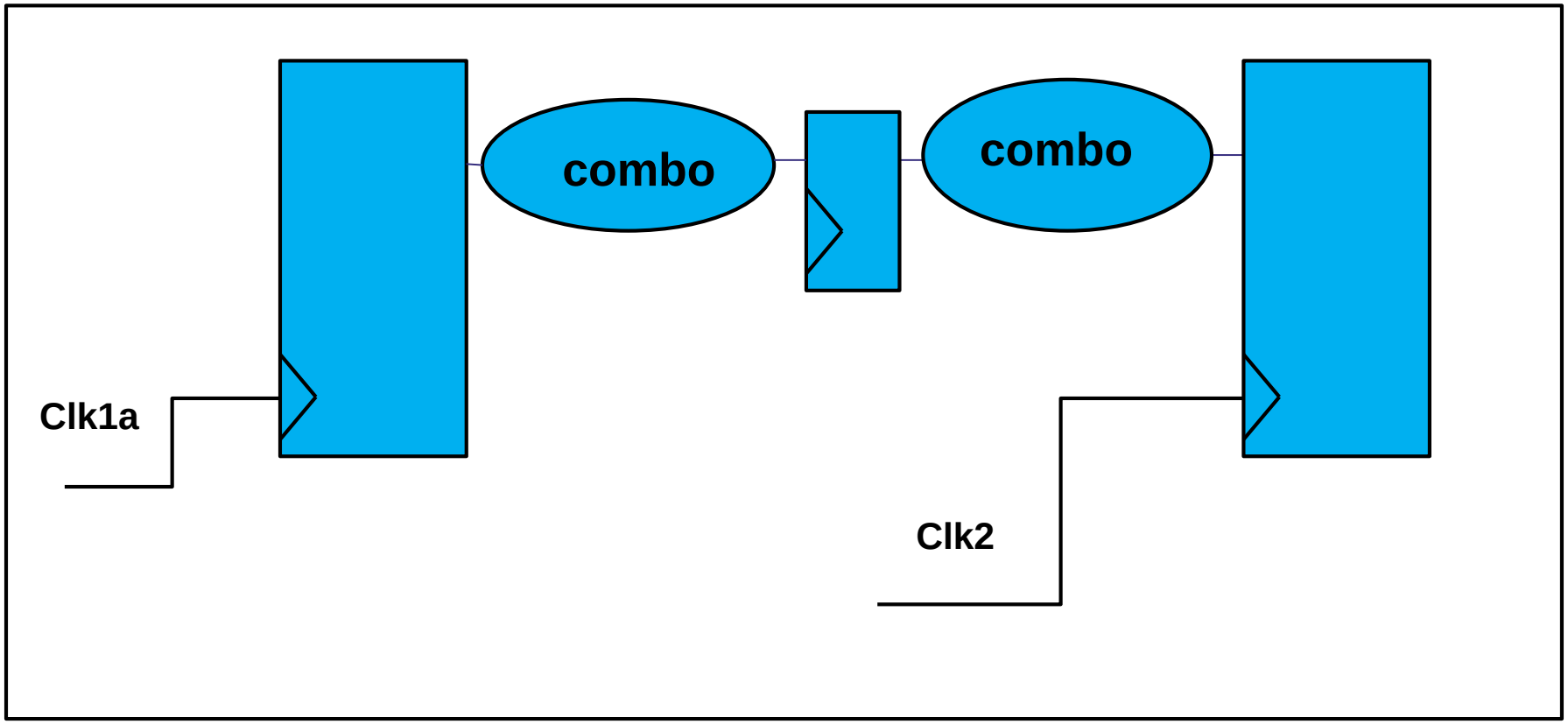
When Latches are Sequential

- The data arrives before the latch opens (no time borrowing)
  - They look just like regular FF's



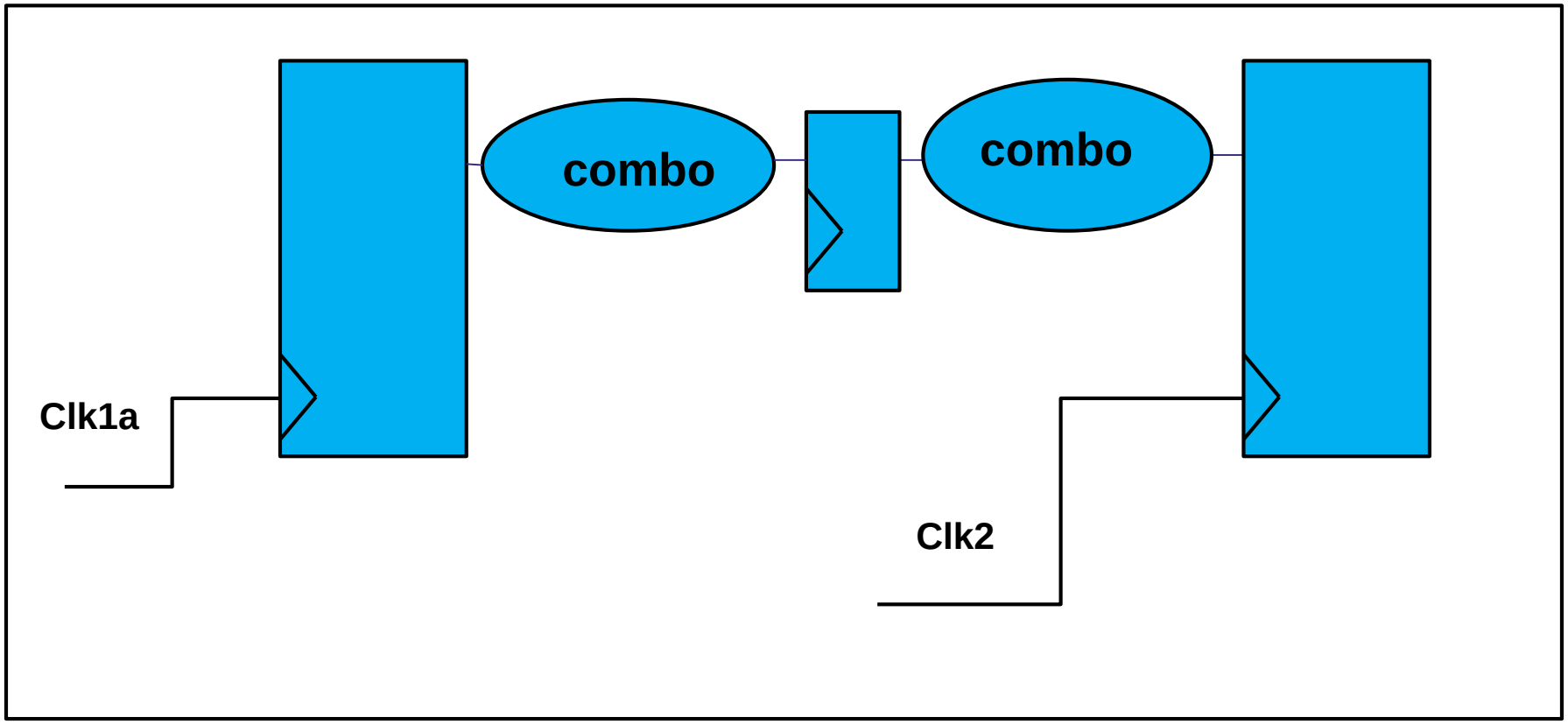
When Latches are Combinational

- The data arrives after the latch opens - time borrowing
  - The latch is now part of the combo cloud



When Latches are Combinational

- The data arrives after the latch opens - time borrowing
  - Tools split the path into 2 parts and time each part separately



Path Segmentation

- Ever wish you could time just a piece of a timing path?
- You Can- with Path Segmentation
- https://solvnet.synopsys.com/retrieve/015888.html?otSearchResultSrc=advSearch&otSearchResultNumber=2&otPageNum=1

- Remember, our startpoints include
  - Any pin with an input delay specified.
- Our endpoints include
  - Any pin with an output delay specified

Path Segmentation

- To make path segmentation more accurate by having the tool account for drive strength and net loading
  - Set the input delay on the output of a leaf cell
  - Set the output delay on the input of a leaf cell

- You'd typically use input and output delays of 0
- Remember to remove them once you've gotten the data you want.