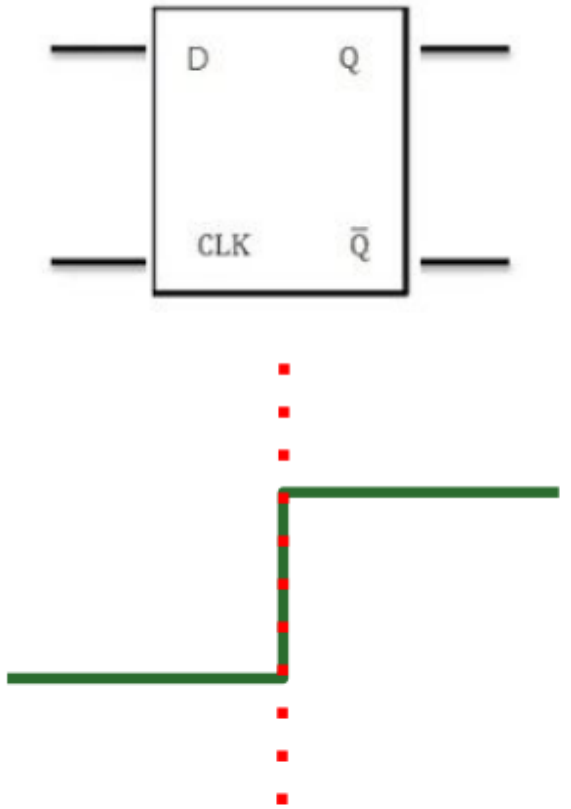
CARP Meeting 2

MULTIPLIER/DIVIDER

RV32I → RV32IM

Ideal Register / Flip Flop

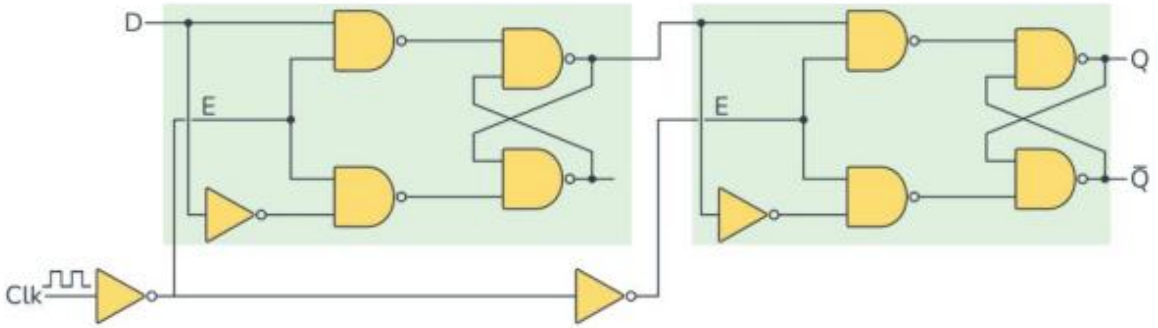
- Samples Data perfectly on clock edge
  - Instantaneous Capture on Perfect Edge
- Immune to noise before and after edge
- Always holds a known binary value
  - 0 or 1
- Data is instantly made available to output



Real Register Part 1

- Takes some time to sample data

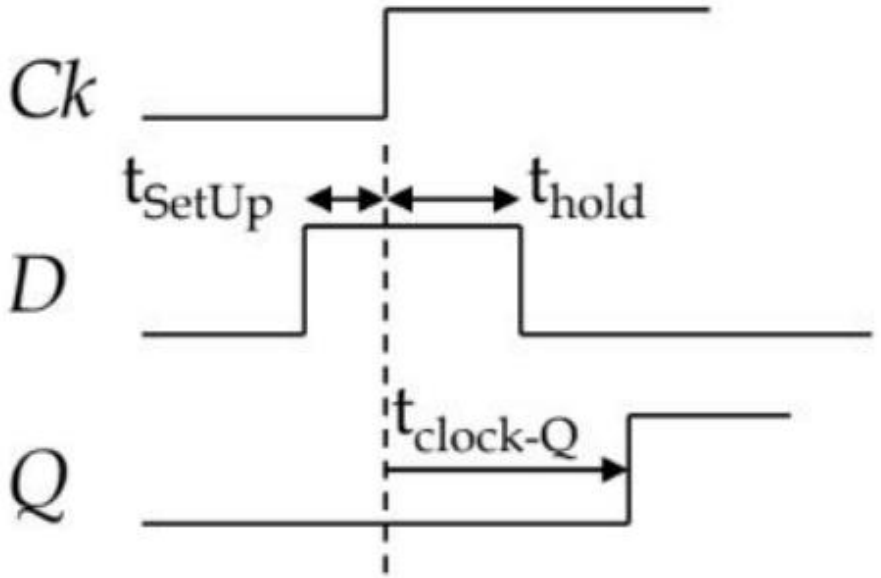
- Requires data to be stable during that time
  - **Setup Time**: min time before the edge that data must remain stable
  - **Hold Time**: min time after the edge that data must remain stable



Glossary

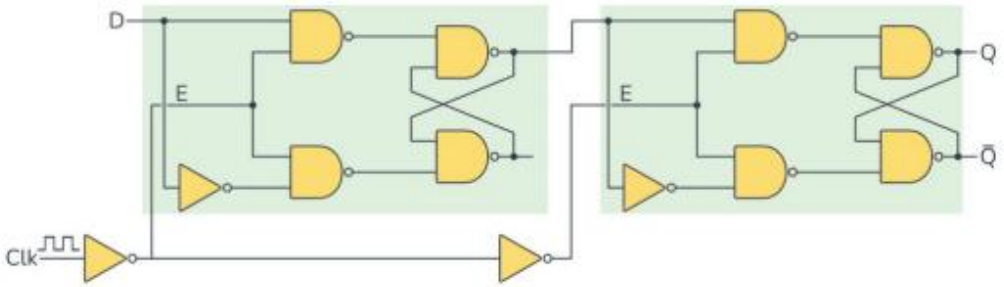
Setup Time t_s: data must be stable t_s before edge

Hold Time t_H: data must be stable t_H after edge



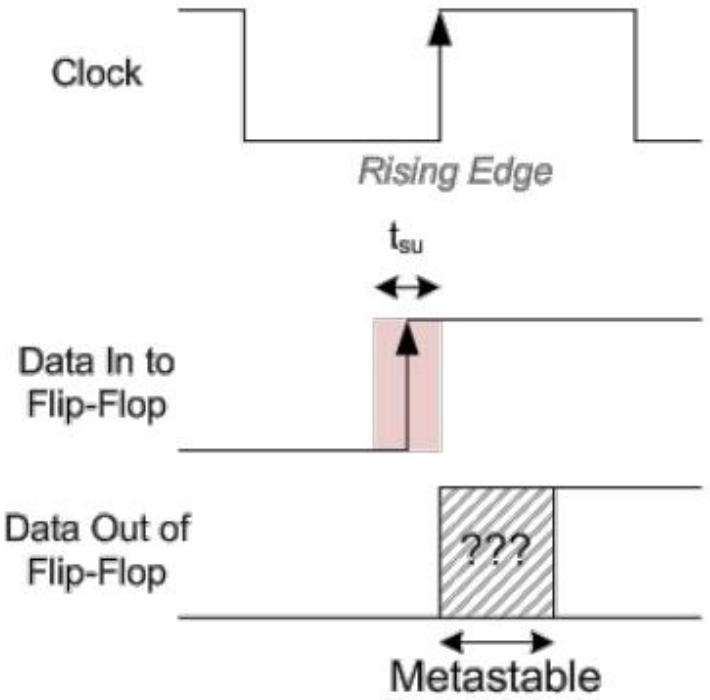
Real Register Part 2

- If signal is not stable, risk **metastability**
  - Register will risk incorrect data, or even worse get stuck
- Once sampled, takes some time to propagate to the output
  - t_{CQ}: clock-to-Q propagation delay



Glossary

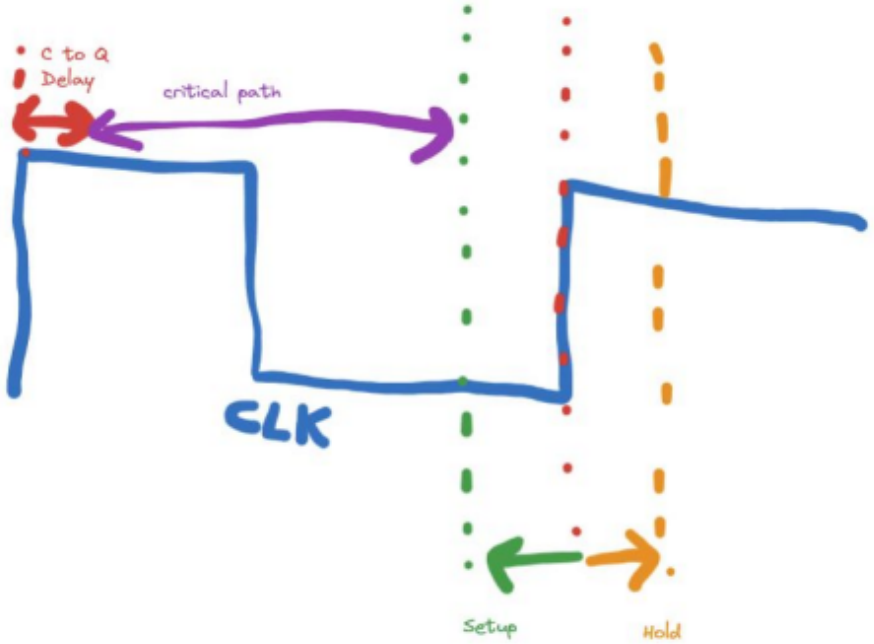
Metastability: register gets stuck between 0 and 1 states for unknown amount of time

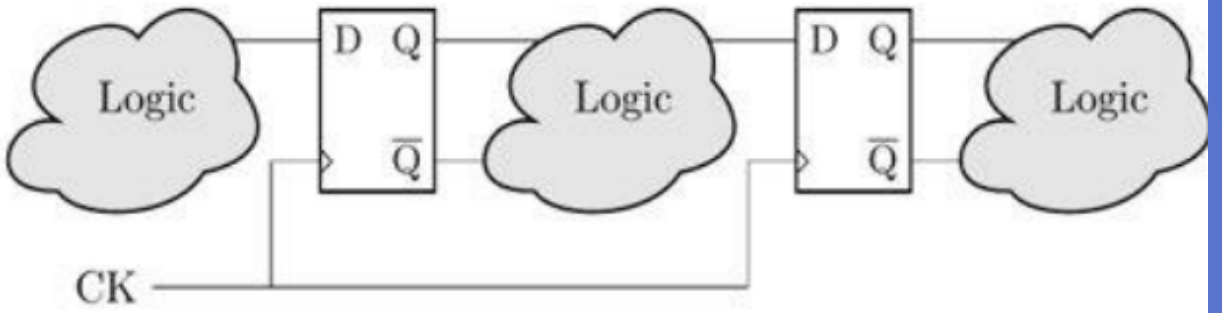


Max Path: Setup Time Violations

- Max Path asks the question:
  - Is your critical path logic too slow to keep up with your clock?
- **Setup Time Violation**: logic is too slow such that data changes past setup time period
  - Fixable by lowering clock frequency
  - Or use faster logic
- Usable Portion of Clock Period:

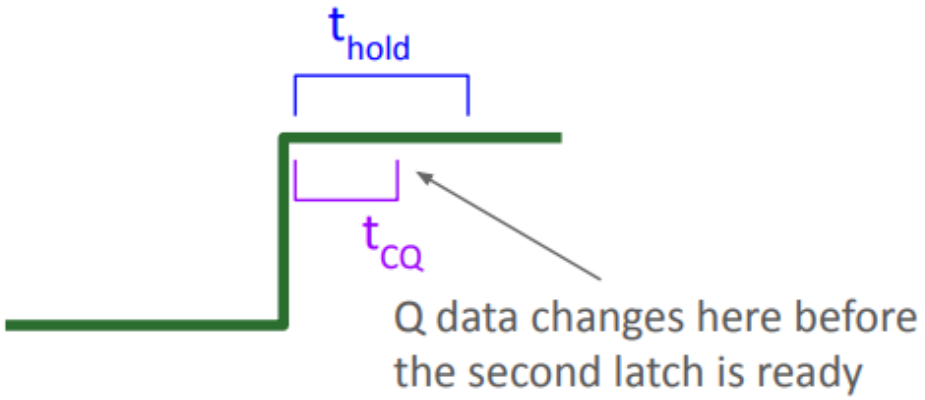
$$t_{logic} \leq t_{clock} - t_{setup} - t_{cq}$$

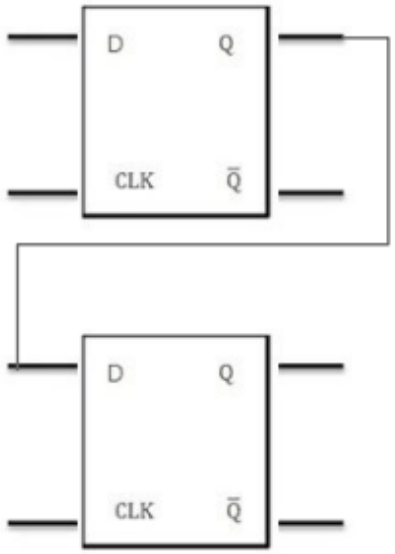




Min Path: Hold Time Violations

- Min Path asks the question:
  - Is your minimum logic path so fast that one register's output will change before the next one has completed hold time?
- **Hold Time Violation:** input to register will change before register is done with its hold time
- Not affected by Clock Speed, only by design
  - **NOT FIXABLE AFTER TAPEOUT**
  - Have to tell the tools (Openlane) to try harder to fix





Slack

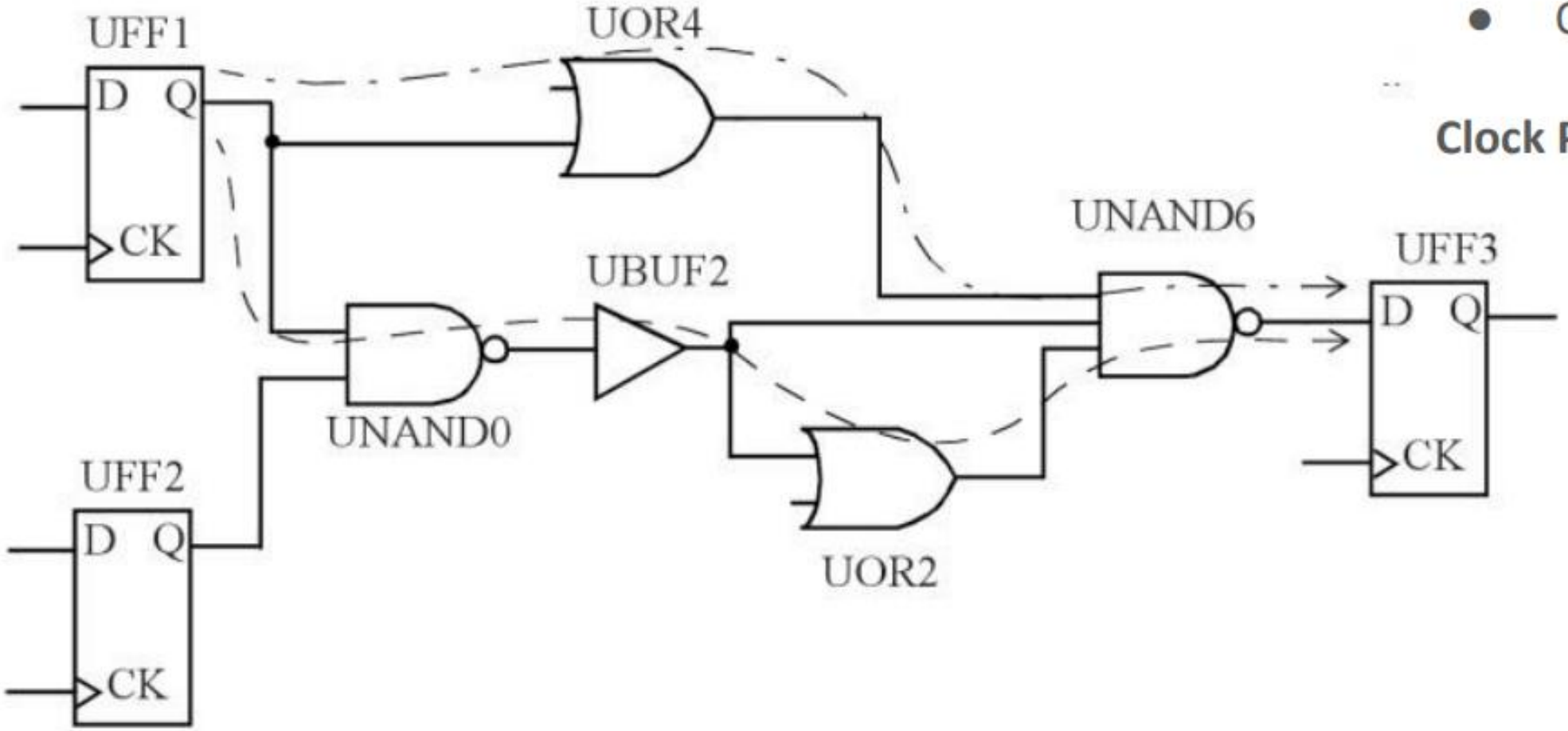
- **Slack** is the amount of leeway between your design's timing and a design constraint
- **Max Path**:
  - **Positive Slack** means you are passing your timing requirement.
    - With a positive slack of 3, you could speed up your clock frequency by up to 3
  - **Zero Slack** means you are passing but with no wiggle room
  - **Negative Slack** means you are violating setup time.
    - To fix, slow down your clock frequency by the amount of negative slack
- **Min Path** is not affected by clock frequency
  - **Negative Slack** in your min path is a hold time violation, which the tools will need to fix by inserting dummy logic
- Computing: Slack = Required Time - Arrival Time
  - S = R - A

STA Example

Delays:

- Buffer: 1
- NAND: 1
- 3-Input NAND: 2
- OR: 2

Clock Period: 5



HARDWARE PERFORMANCE TRADEOFFS FOR DUMMIES

COMBINATIONAL = MORE HARDWARE (do everything at once)

- Critical Path = longest logic chain between registers
- Setup Time = time to get through all the logic to the register
- Hold Time = time to hold the data while register captures it

SEQUENTIAL = LESS HARDWARE (do everything in steps, reuse hardware)

- IPC goes way down depending on # of cycles wasted in calculation
- Sometimes worth the cost if hardware too complex

REMEMBER: Critical Path determines CLK frequency

LETS BUILD HARDWARE LIKE A 3rd GRADER!

How does a child multiply two numbers? → REPEATED ADDITION

How does a child divide two numbers? → REPEATED SUBTRACTION

BINARY MULTIPLICATION:

Algorithm: SHIFT-ADD

01001 (9)
X 011 (3)

BINARY DIVISION

01001/ (9)
011 (3)

How to handle SIGNED

STEPS

1. CONVERT SIGNED INPUTS TO UNSIGNED (take absolute value)

2. MULTIPLY/DIVIDE UNSIGNED

3. NEGATE THE RESULT AT THE END

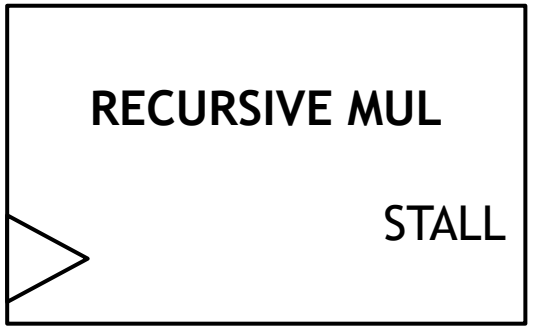
How to avoid the extremes

A combinational multiply and divide is a lot of hardware cost, especially when we're considering a hardware limited ASIC implementation for our design.

Let's get the best of both worlds

- Perform multiplication in steps
- Perform each step combinationally

INSTEAD OF a big combinational multiplier, or a recursive multiplier with a stall..



PIPELINE!!!!

DO A COMBINATIONAL MULTIPLY ACROSS 3 STAGES!!
31 possible shifts, 32 possible adds
32 bits handled across 3 stages = 10.67 shift-adds per stage (11)
Results in best performance vs logic depth