inst.eecs.berkeley.edu/~eecs251b

EECS251B : Advanced Digital Circuits and Systems

Lecture 14 – Timing

Borivoje Nikolić

Data Centers Could Soon Break Lunar Ground

March 4, 2024, EETimes. When the world's first commercial lunar lander made its historic touchdown near the moon's south pole last week, the promise of another groundbreaking venture was also in the works. Stored within a software application loaded onto the craft's on-board computer were the makings of a lunar data center, along with the hopes of redefining deep-space computing.

One of six commercial payloads aboard the Intuitive Machines' Nova-C lander, the prototype belonged to a Florida-based company called Lonestar Data Holdings. Following a series of tests last week, the company said in prepared remarks that it had successfully tested the transmission, storage and receipt back of digital documents during lunar flight and again while on the surface of the moon.

Intuitive Machines' Nova-C lander touches down near the lunar south pole. Its on-board computer houses a software prototype for what could be the first lunar data center. (Source: Intuitive Machines)

Announcements

- Lab 5 still waiting on PDK correction
  - The newest fix brings it very close
- Start project phase 1
  - Spec doc due this week
- Homework 2 due this week
  - Quiz 2 on March 12

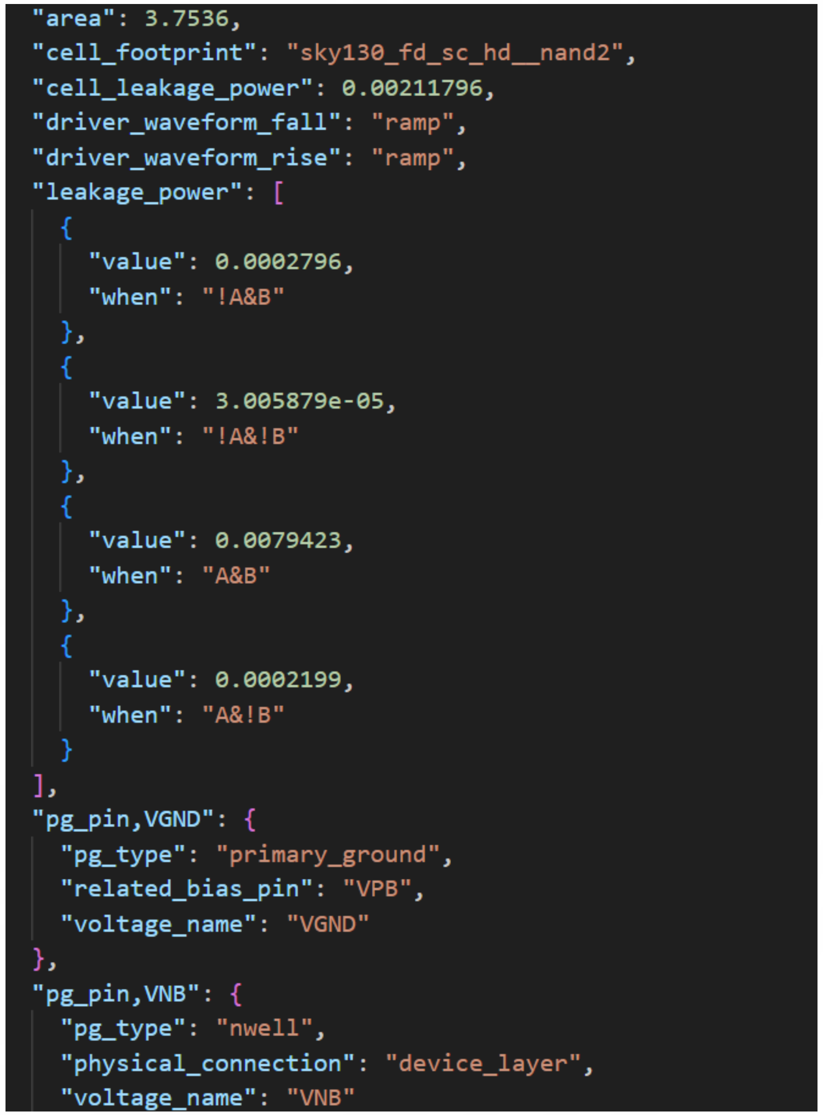
Standard Cell Library

Standard Cell Library

- Contains for each cell:
  - Functional information: cell = a *b * c
  - Timing information: function of
    - input slew
    - intrinsic delay
    - Input/output capacitance

    non-linear models used in tabular approach
  - Physical footprint (area)
  - Power characteristics
  - Noise sensitivity
- Wire-load models - function of
  - Block size
  - Fan-out

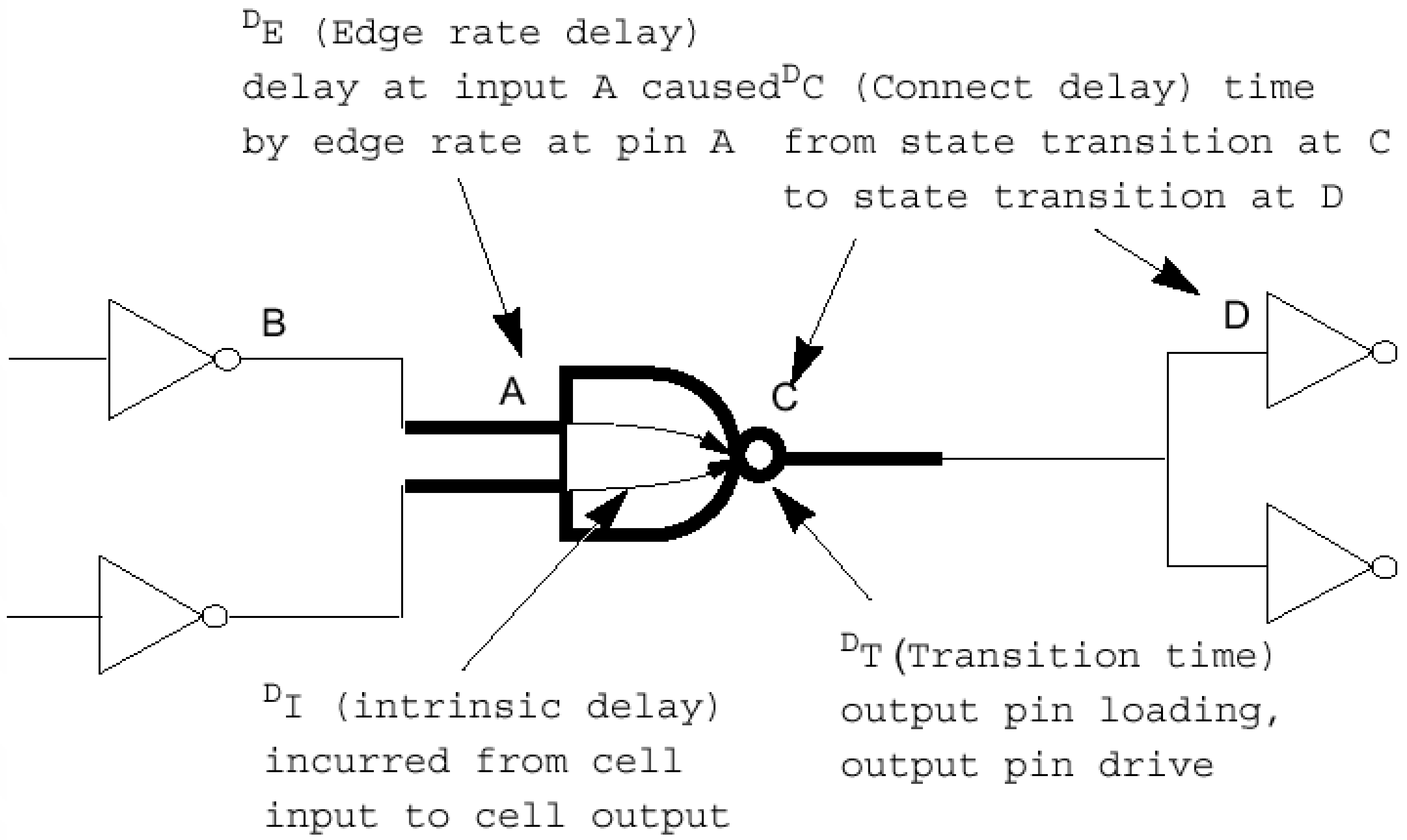
Example: NAND2



```
"area": 3.7536,
"cell_footprint": "sky130_fd_sc_hd__nand2",
"cell_leakage_power": 0.00211796,
"driver_waveform_fall": "ramp",
"driver_waveform_rise": "ramp",
"leakage_power": [
  {
    "value": 0.0002796,
    "when": "!A&B"
  },
  {
    "value": 3.005879e-05,
    "when": "!A&!B"
  },
  {
    "value": 0.0079423,
    "when": "A&B"
  },
  {
    "value": 0.0002199,
    "when": "A&!B"
  }
],
"pg_pin,VGND": {
  "pg_type": "primary_ground",
  "related_bias_pin": "VPB",
  "voltage_name": "VGND"
},
"pg_pin,VNB": {
  "pg_type": "nwell",
  "physical_connection": "device_layer",
  "voltage_name": "VNB"
```

Synopsys Delay Models

- Linear (CMOS2) delay model
  - Similar to what we have studied so far


D_E (Edge rate delay)
delay at input A caused by edge rate at pin A
D_C (Connect delay) time from state transition at C to state transition at D
B
A
C
D
D_I (intrinsic delay) incurred from cell input to cell output
D_T (Transition time) output pin loading, output pin drive

Example Cell Timing

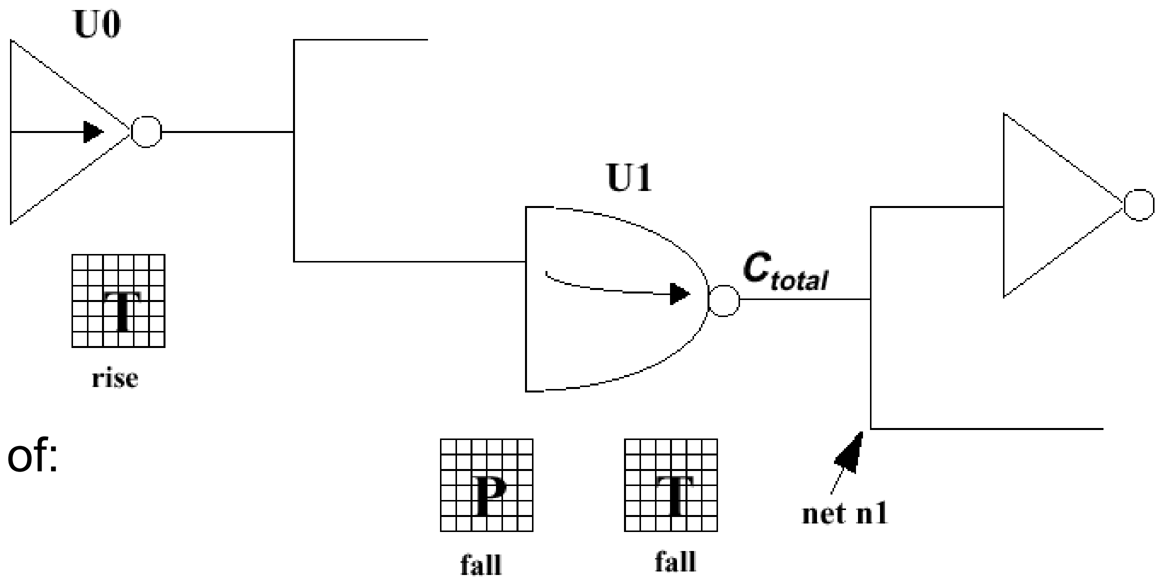
- From Synopsys training materials

```
From pin: U28/A
To pin: U28/Z

arc type :                      cell
arc sense :                     unate
Input net transition times:     Dt_rise = 0.1458, Dt_fall = 0.0653

Rise Delay computation:
rise_intrinsic                  0.48 +
rise_slope * Dt_rise            0 * 0.1458 +
rise_resistance * (pin_cap + wire_cap) / driver_count
0.1443 * (2 + 0) / 1
rise_transition_delay :         0.2886
-----------------------------------
Total                           0.7686
```

Cell Characterization (Linear Model)

```
cell(NAND2) {
  area : 1;
  pin(X) {
    function : "(A B)'";
    direction : output;
    edge_rate_rise : 0.24;
    edge_rate_fall : 0.14;
    edge_rate_load_rise : 5.4;
    edge_rate_load_fall : 3.4;
    timing() {
    intrinsic_rise  : 0.34;
    intrinsic_fall  : 0.24;
    rise_resistance : 3.4;
    fall_resistance : 1.4;
    edge_rate_sensitivity_r0 : 0.24;
    edge_rate_sensitivity_f0 : 0.14;
    edge_rate_sensitivity_r1 : 0.14;
    edge_rate_sensitivity_f1 : 0.04;
    related_pin : "A";
    }
    timing() {
    intrinsic_rise  : 0.34;
    intrinsic_fall  : 0.24;
    rise_resistance : 3.4;
    fall_resistance : 1.4;
    edge_rate_sensitivity_r0 : 0.24;
    edge_rate_sensitivity_f0 : 0.14;
    edge_rate_sensitivity_r1 : 0.14;
    edge_rate_sensitivity_f1 : 0.04;
    related_pin : "B";
    }
  }
  pin(A) {
    direction : input;
    capacitance : 0.10;
  }
  pin(B) {
    direction : input;
    capacitance : 0.10;
  }
}
```

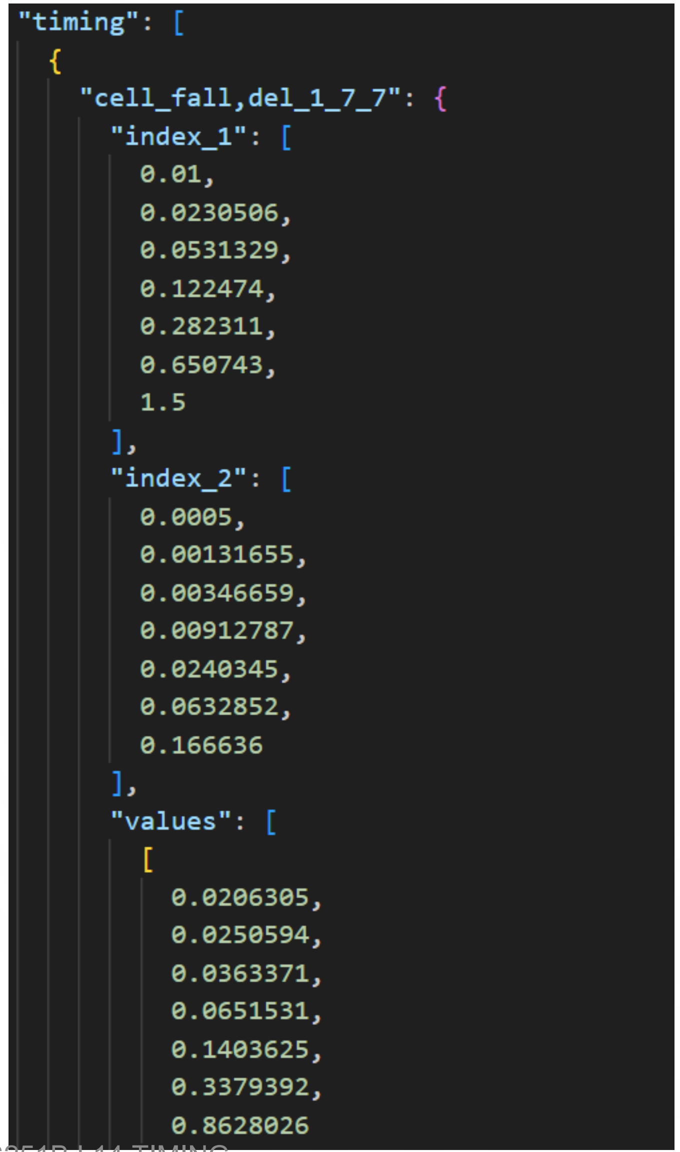
(Synopsys) Nonlinear Delay Model (NLDM)



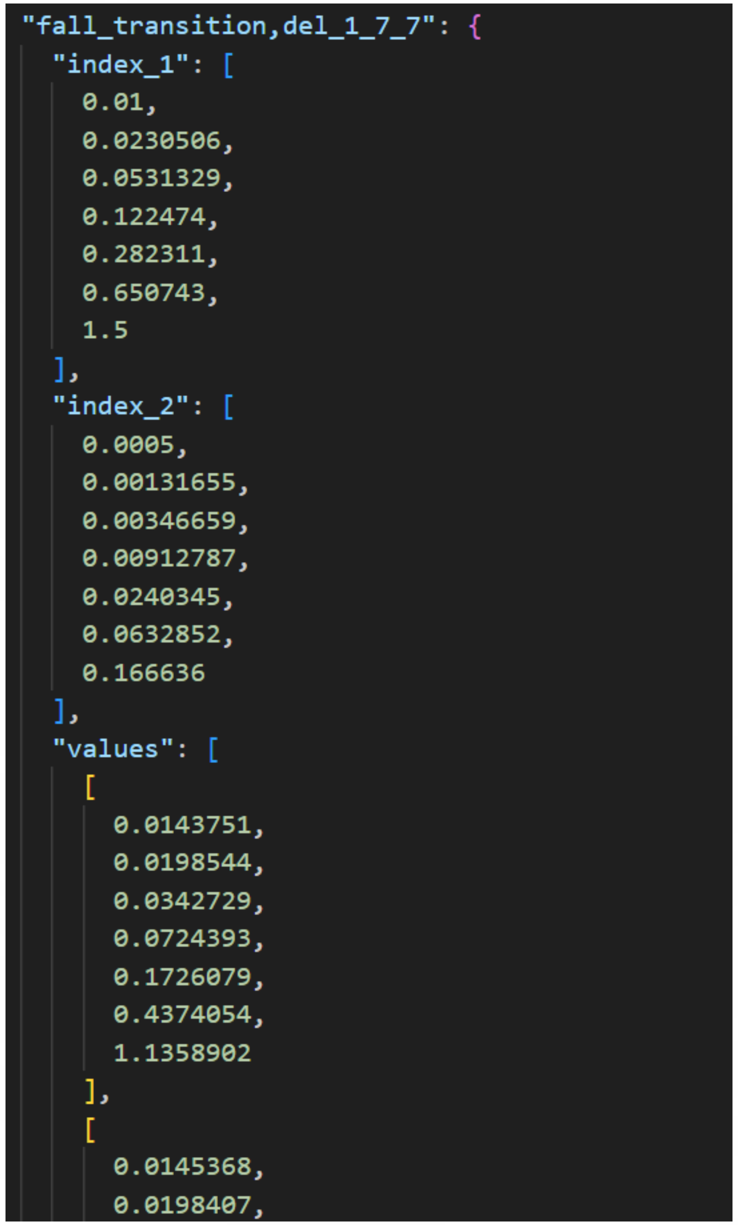
Delay is a function of:

- Rise propagation
- Cell rise
- Fall propagation
- Cell fall
- Rise transition
- Fall transition

NAND2 (Sky130)



```
"timing": [
  {
    "cell_fall,del_1_7_7": {
      "index_1": [
        0.01,
        0.0230506,
        0.0531329,
        0.122474,
        0.282311,
        0.650743,
        1.5
      ],
      "index_2": [
        0.0005,
        0.00131655,
        0.00346659,
        0.00912787,
        0.0240345,
        0.0632852,
        0.166636
      ],
      "values": [
        [
          0.0206305,
          0.0250594,
          0.0363371,
          0.0651531,
          0.1403625,
          0.3379392,
          0.8628026
```



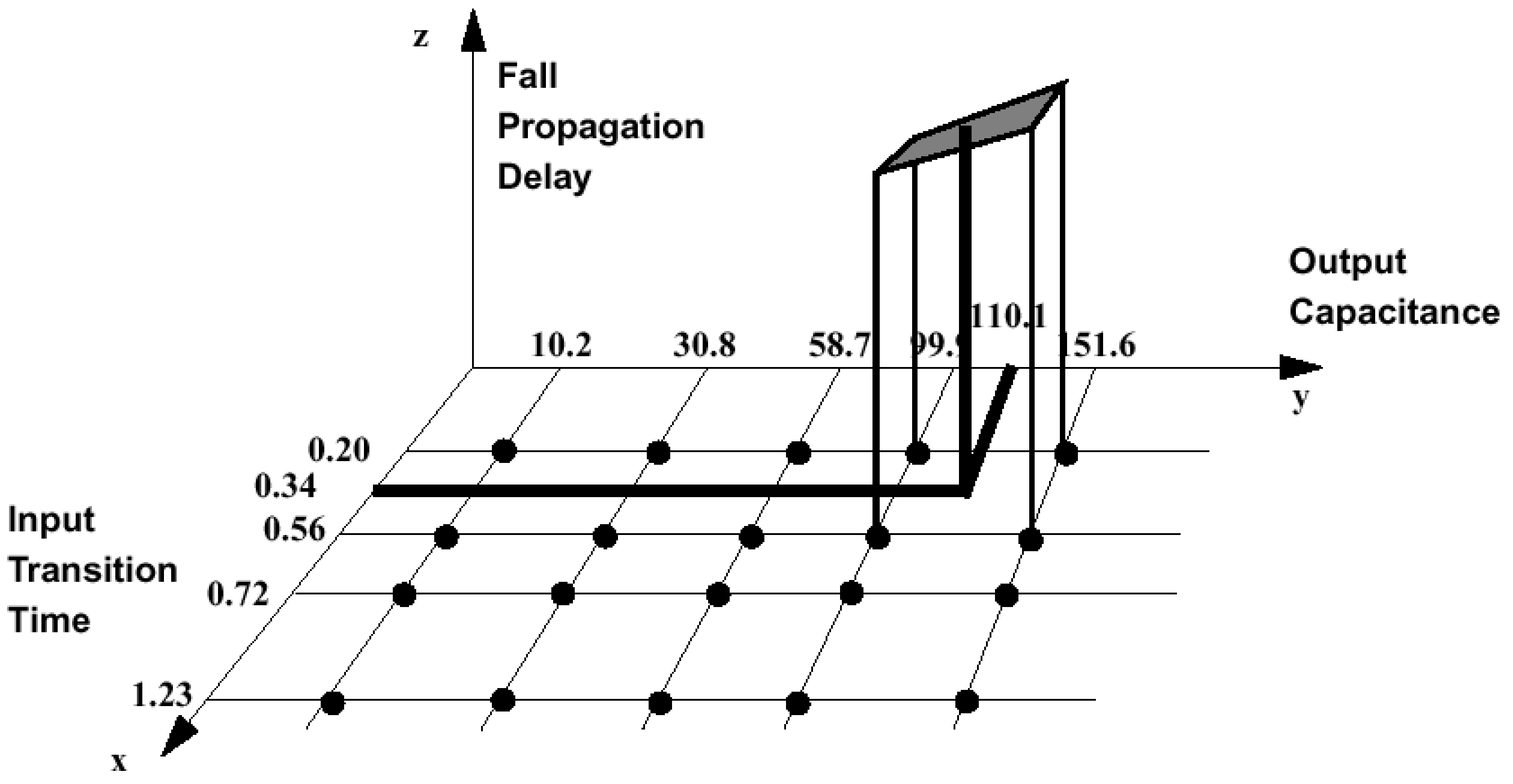
```
"fall_transition,del_1_7_7": {
  "index_1": [
    0.01,
    0.0230506,
    0.0531329,
    0.122474,
    0.282311,
    0.650743,
    1.5
  ],
  "index_2": [
    0.0005,
    0.00131655,
    0.00346659,
    0.00912787,
    0.0240345,
    0.0632852,
    0.166636
  ],
  "values": [
    [
      0.0143751,
      0.0198544,
      0.0342729,
      0.0724393,
      0.1726079,
      0.4374054,
      1.1358902
    ],
    [
      0.0145368,
      0.0198407,
```

Two-dimensional tables of pre-characterized delays/transition times as a function of input slope and output capacitance

Index1 – input transition
Index2 – load capacitance

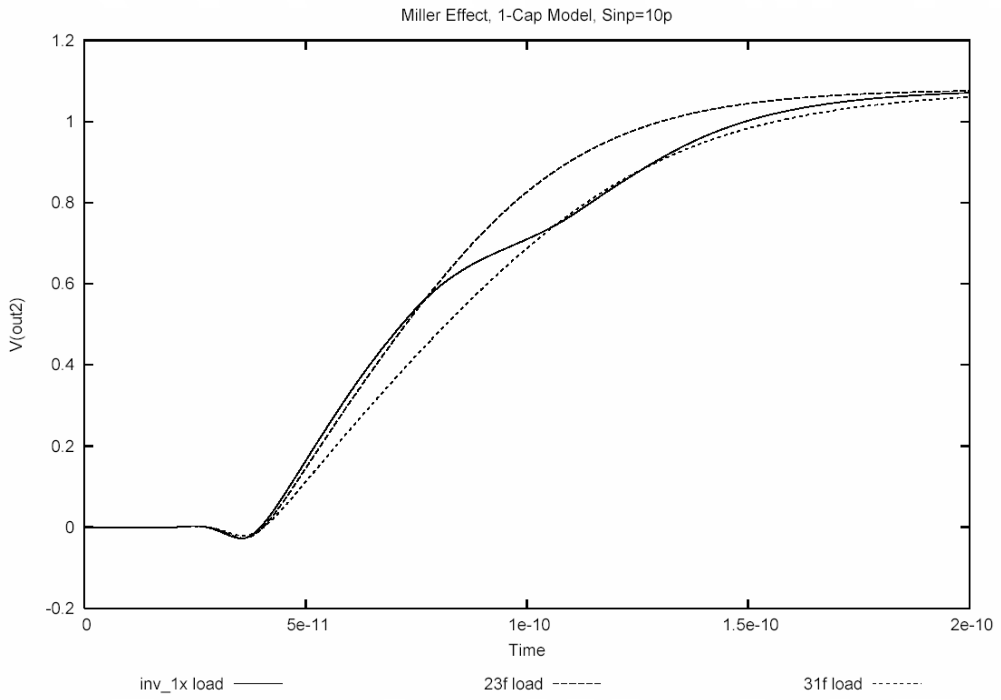
Nonlinear Delay Model (NLDM)

- Interpolates between characterization points

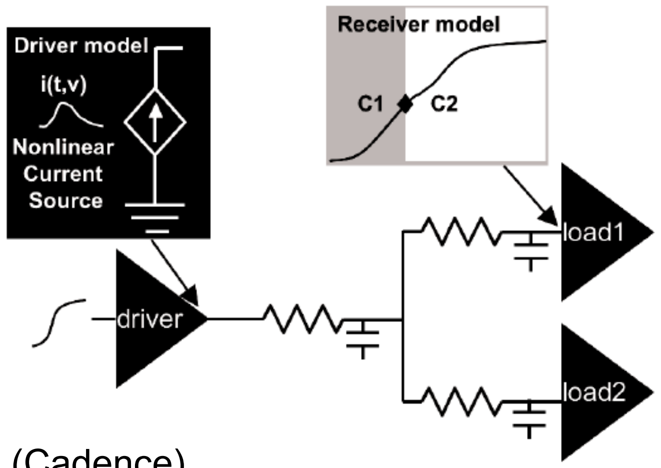


Composite Current Source (CCS) Model

- **Driver model**
  - Composite current source (time and voltage dependent)
- **Receiver model**
  - A set of capacitance models
  - Wire model
- **Interpolate**



Matches both delay and rise/fall times



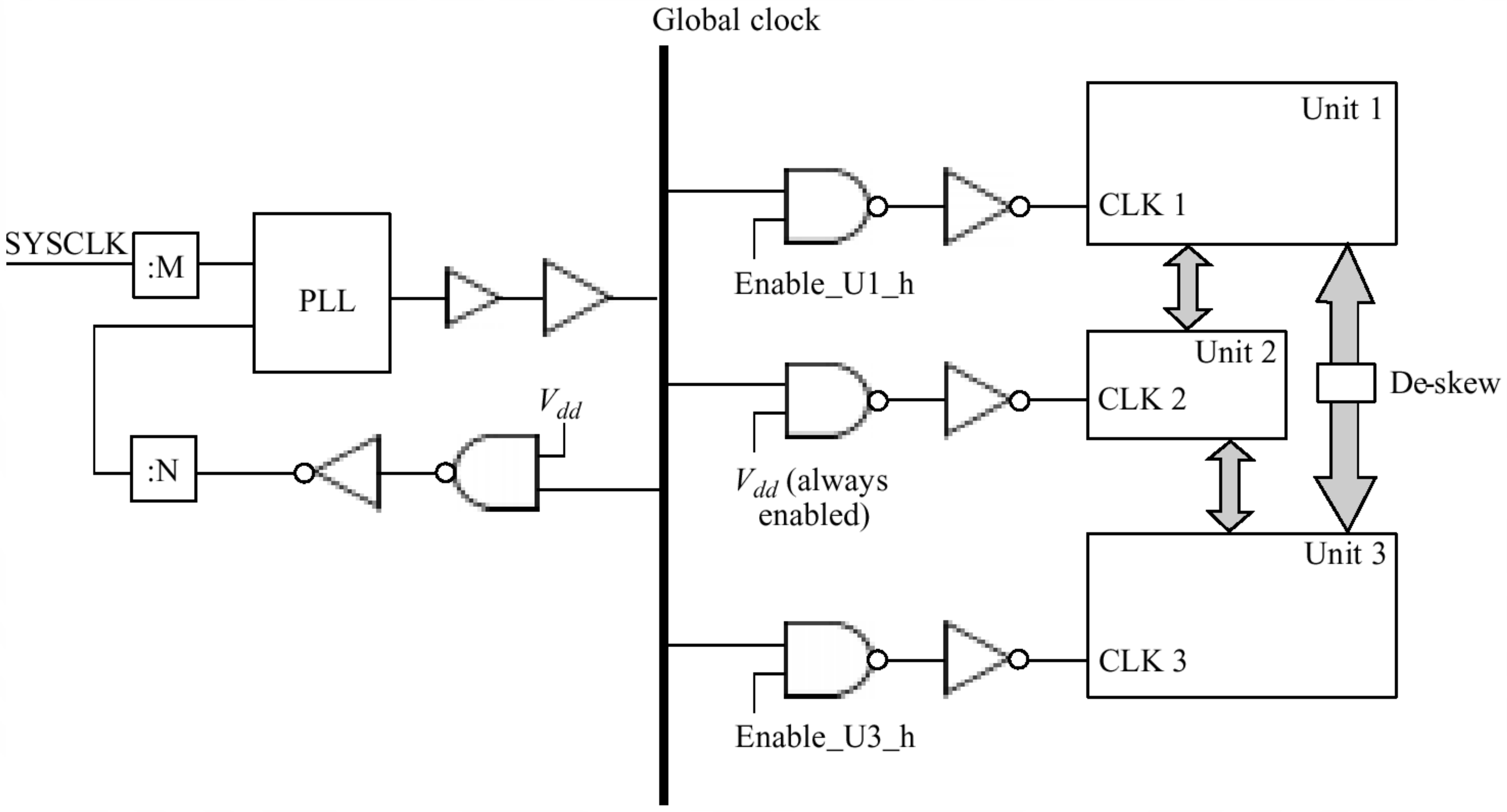
Synopsys

And then there is Effective Current Source Model … (Cadence)

Design for Performance

Flip-Flop-Based Timing

Example Clock System

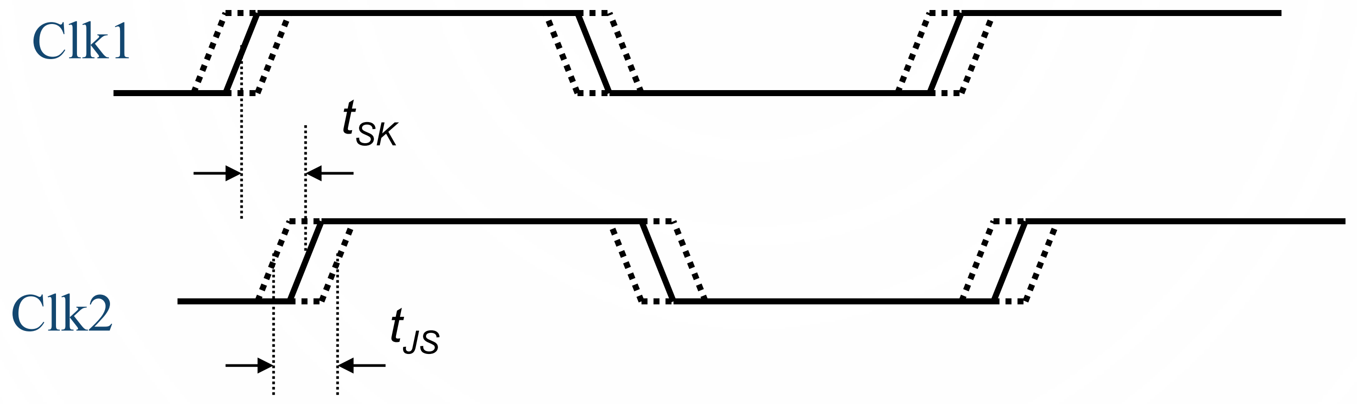


Courtesy of IEEE Press, New York. © 2000

Clock Nonidealities

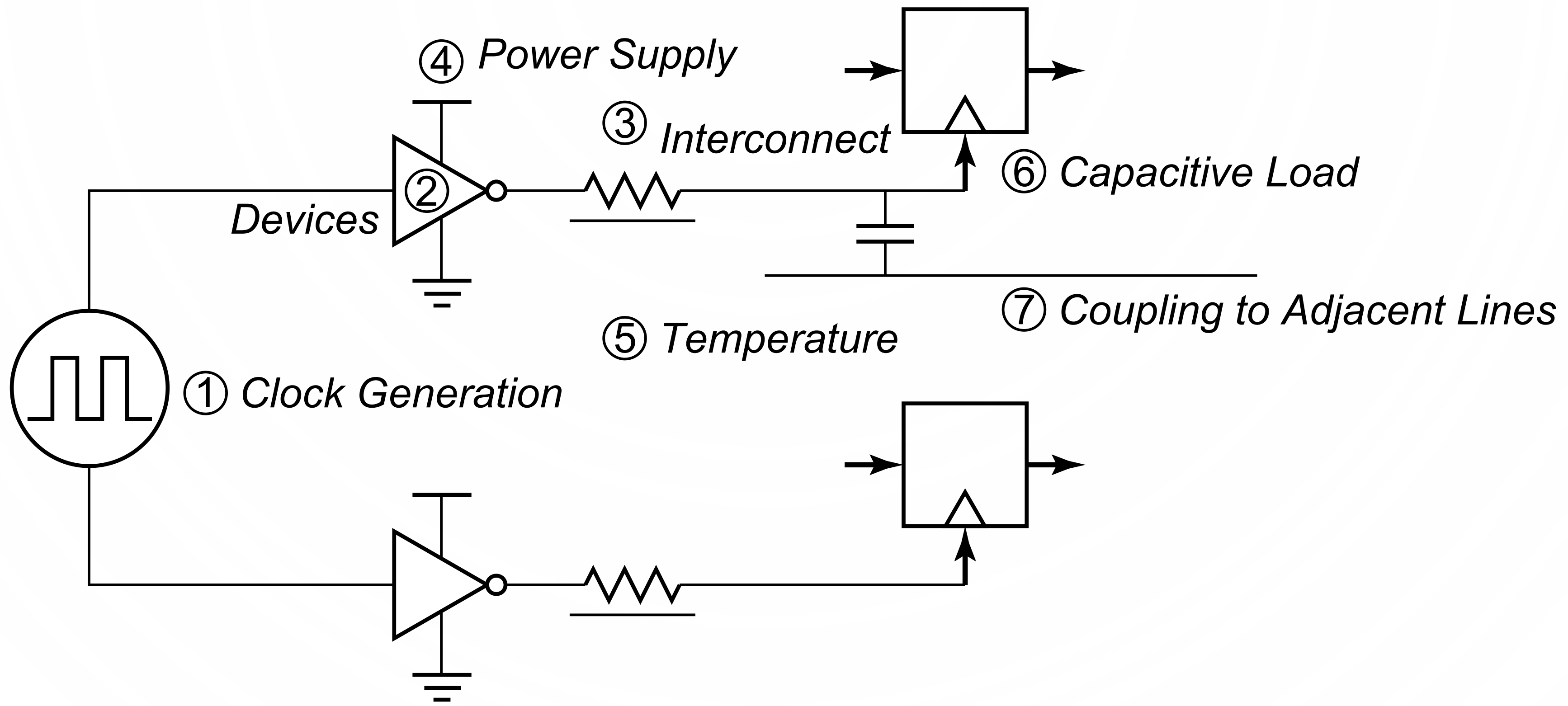
- Clock skew
  - Spatial variation in temporally equivalent clock edges; deterministic + random, t_{SK}
- Clock jitter
  - Temporal variations in consecutive edges of the clock signal; modulation + random noise
  - Cycle-to-cycle (short-term) - t_{JS}
  - Long-term - t_{JL}
- Variation of the pulse width
  - for level-sensitive clocking

Clock Skew and Jitter



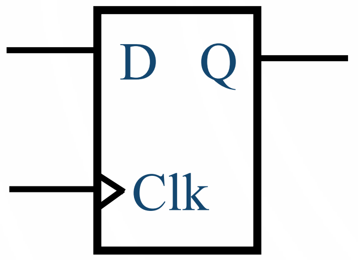
- Both skew and jitter affect the effective cycle time
- Only skew affects the race margin, if jitter is from the source
  - Distribution-induced jitter affects both

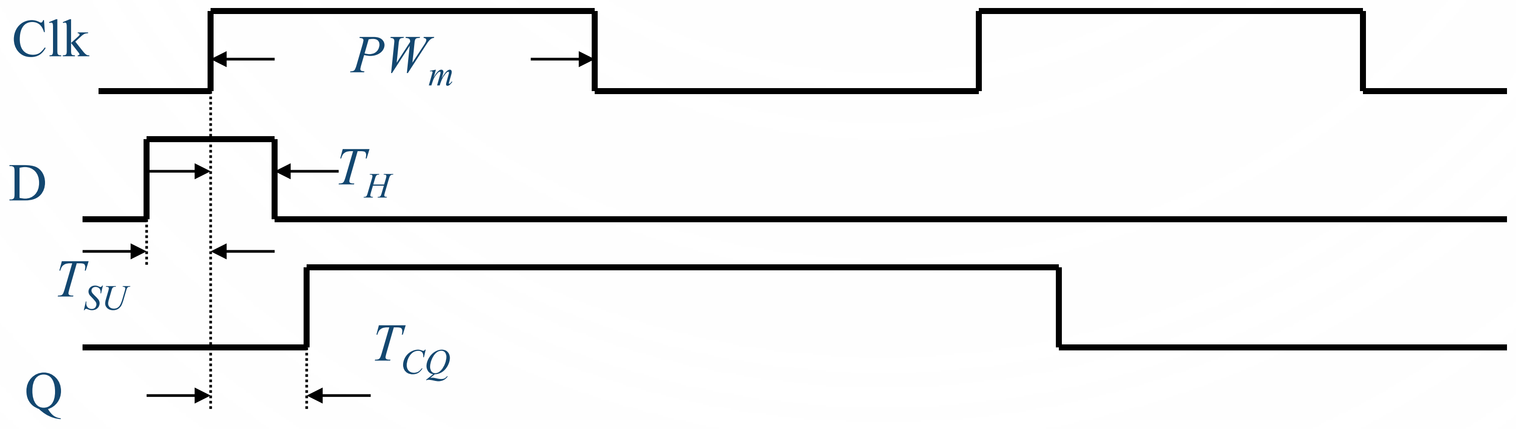
Clock Uncertainties



Sources of clock uncertainty

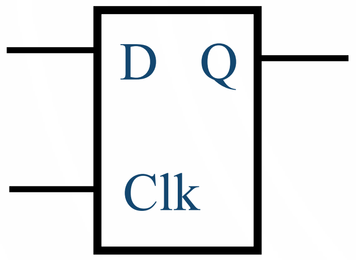
Flip-Flop Parameters



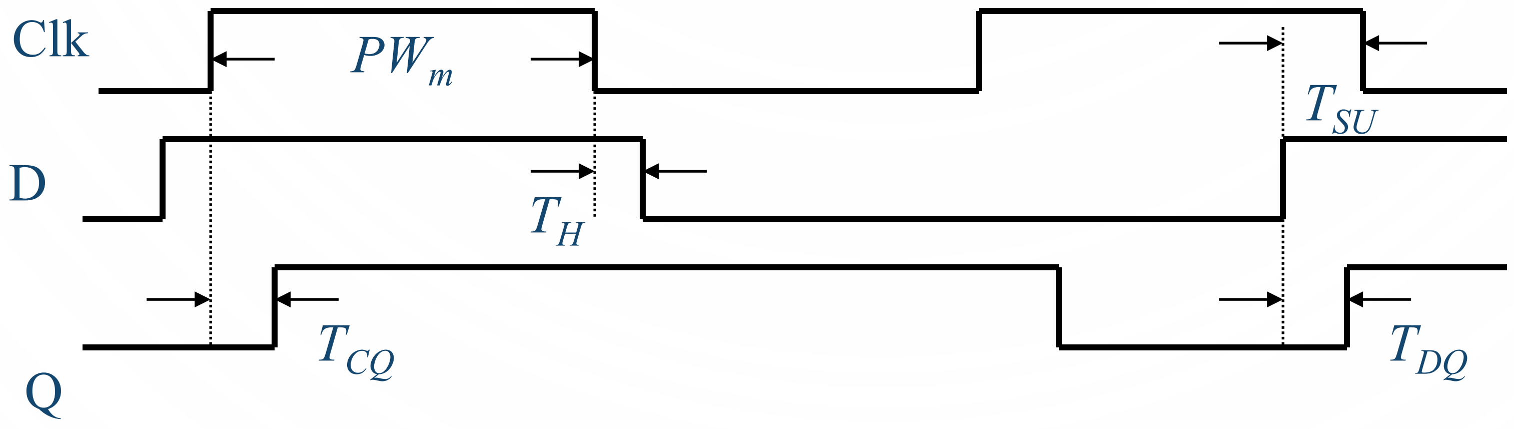


Delays can be different for rising and falling data transitions

Latch Parameters

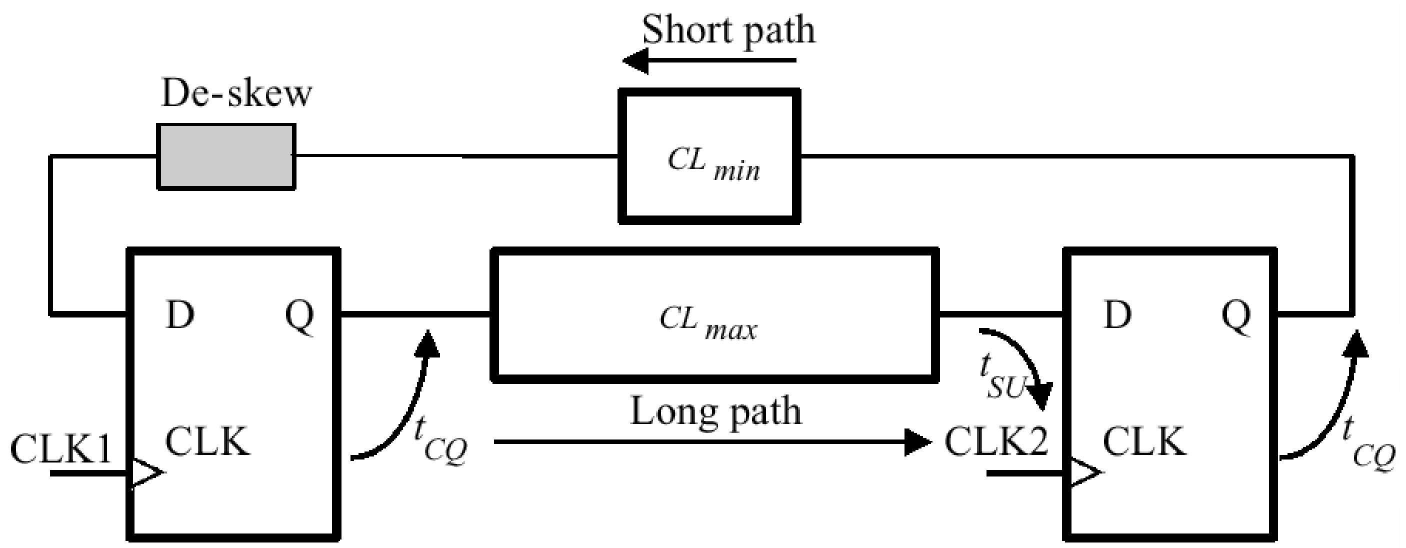


Unger and Tan
Trans. on Comp.
10/86



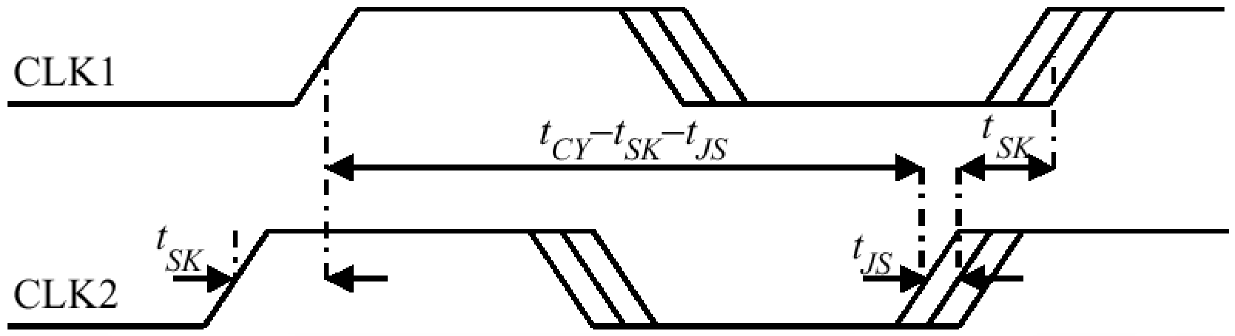
Delays can be different for rising and falling data transitions

Clock Constraints in Edge-Triggered Systems



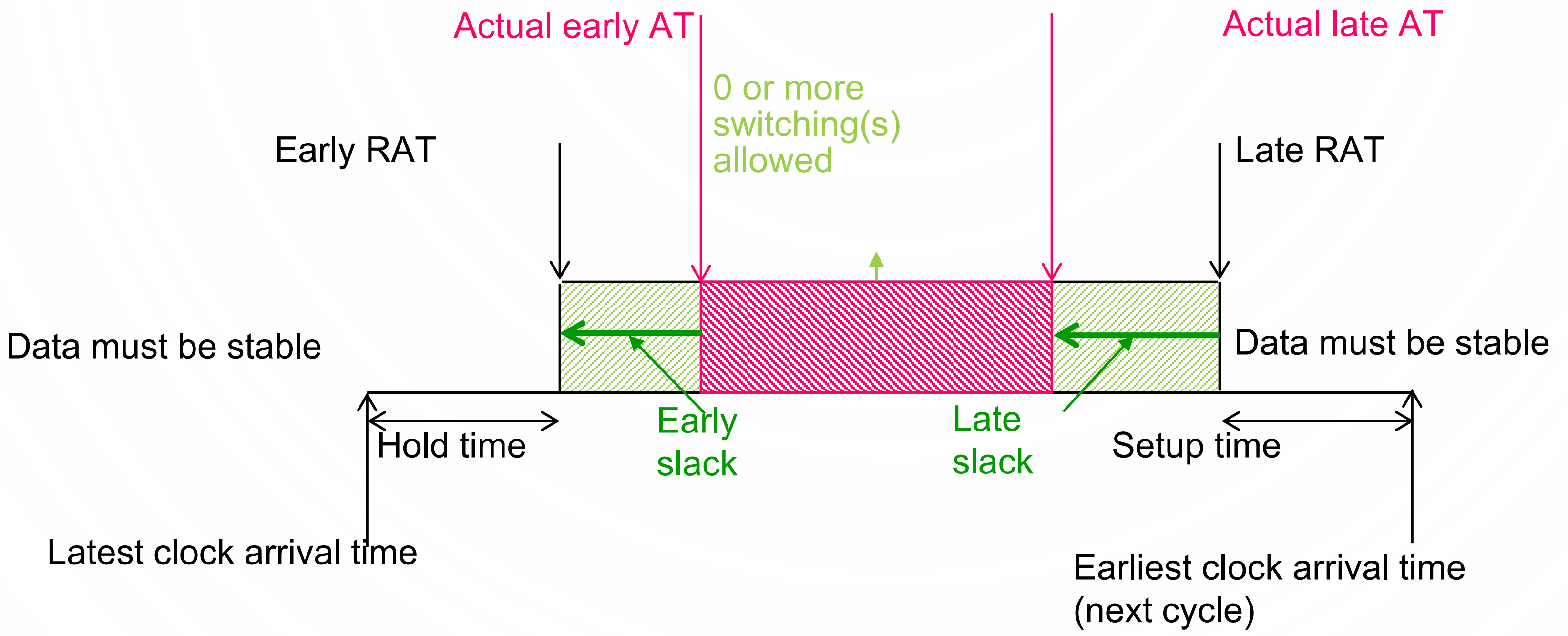
$$t_{CL} \le (t_{CY} - t_{SK} - t_{JS}) - (t_{SU} + t_{CQ})$$

$$t_{CL} \ge t_{SK} + (t_H - t_{CQ})$$



Courtesy of IEEE Press, New York. © 2000

Pictorial View of Setup and Hold Tests



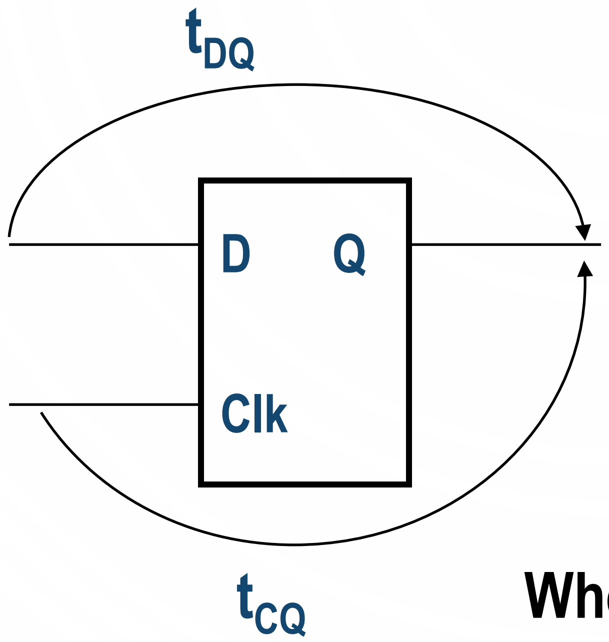
ICCAD '07 Tutorial Chandu Visweswariah

Latch Timing

Key Point

- Latch-based sequencing can improve performance, but is more complicated
  - Timing analysis not limited to a consecutive pair of latches

Latch Timing



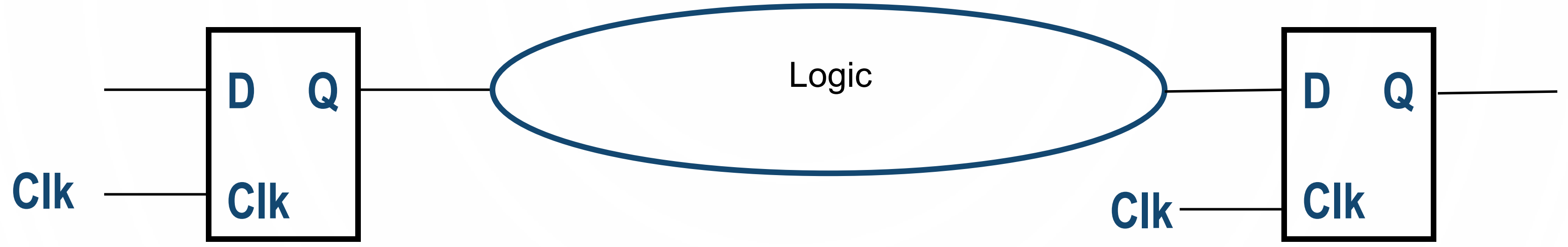
When data arrives
to transparent latch

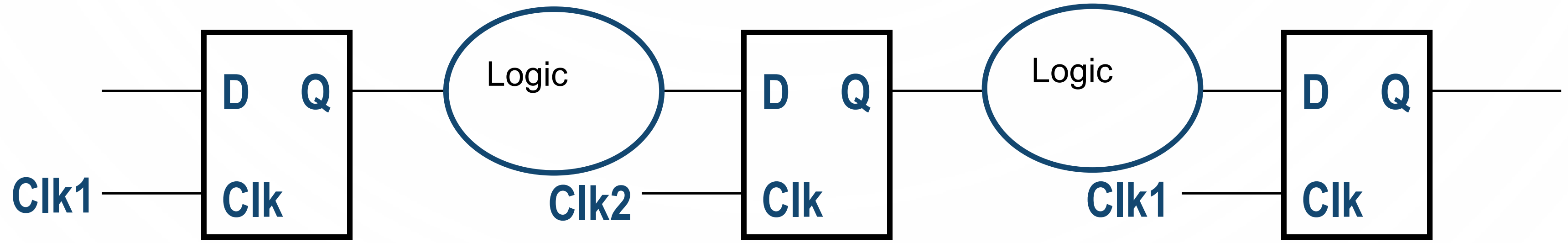
Latch is a 'soft' barrier

When data arrives
to non-transparent latch

Data has to be 're-launched'

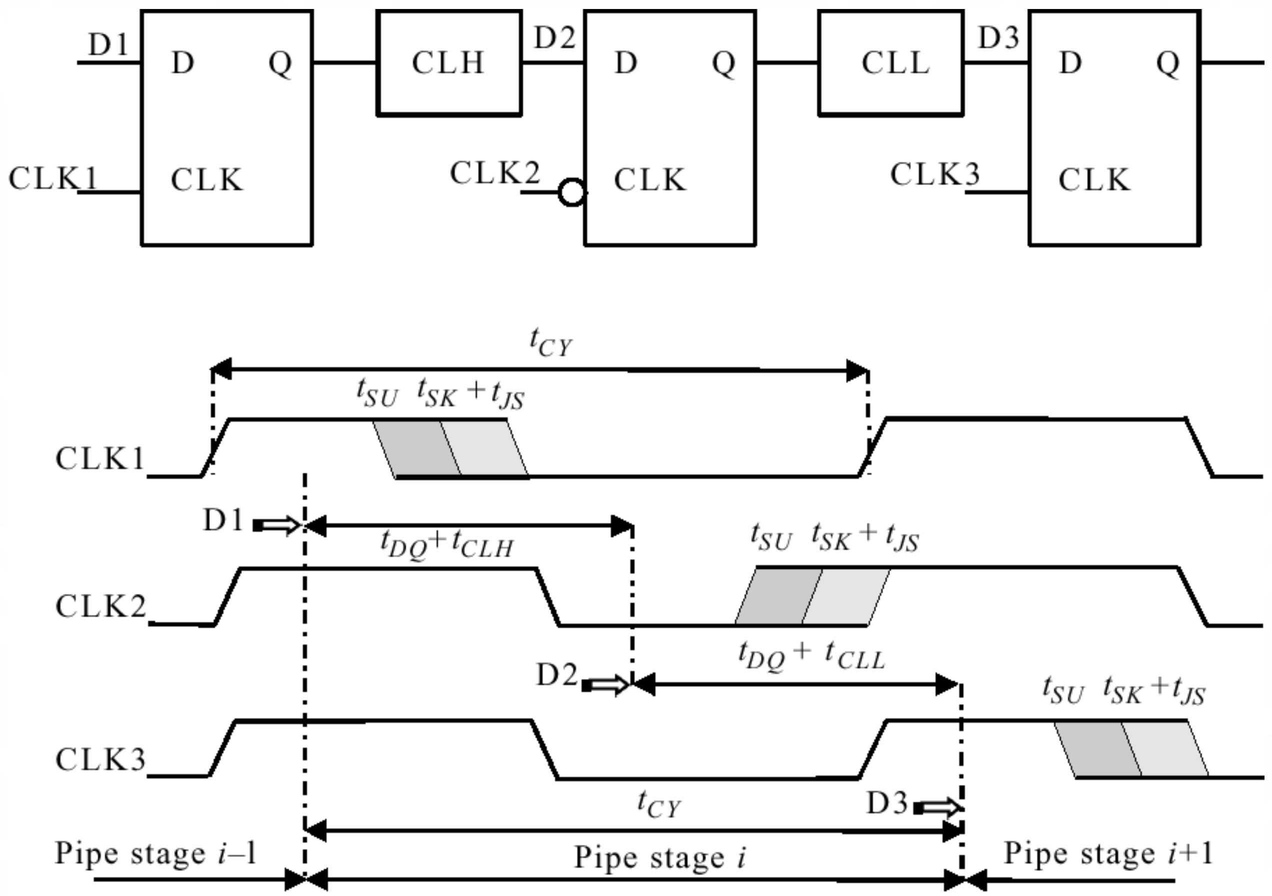
Latch Sequencing





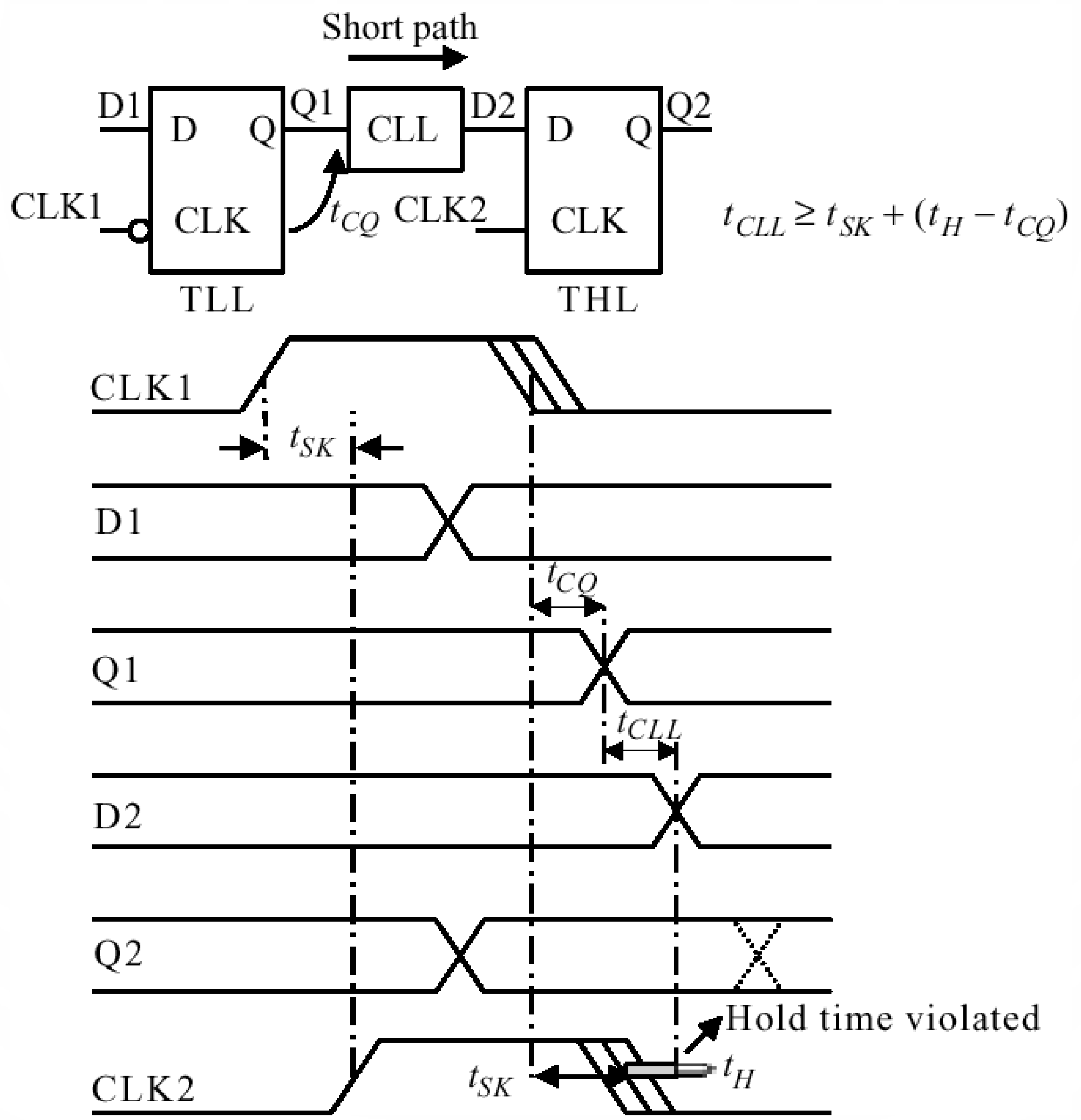
Latch-Based Timing

- Single-phase, two-latch



As long as transitions are within the assertion period of the latch, no impact of position of clock edges

Latch Design and Hold Times



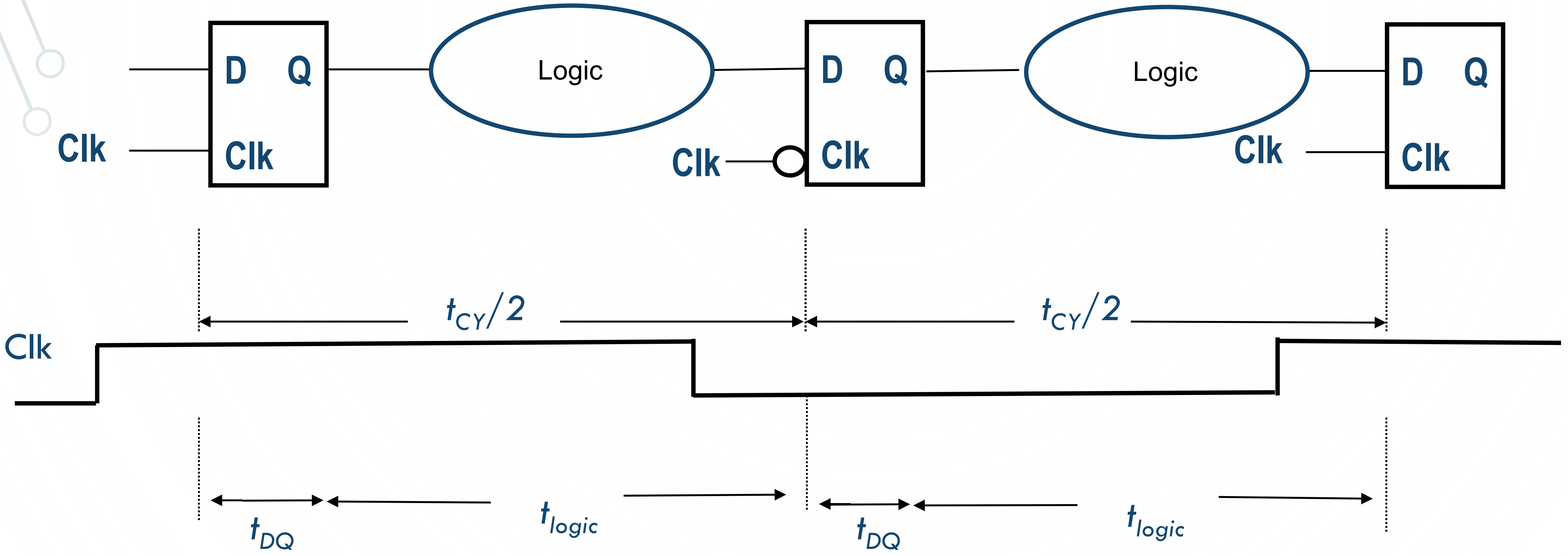
$$t_{CLL} \geq t_{SK} + (t_H - t_{CQ})$$

Soft-Edge Properties of Latches

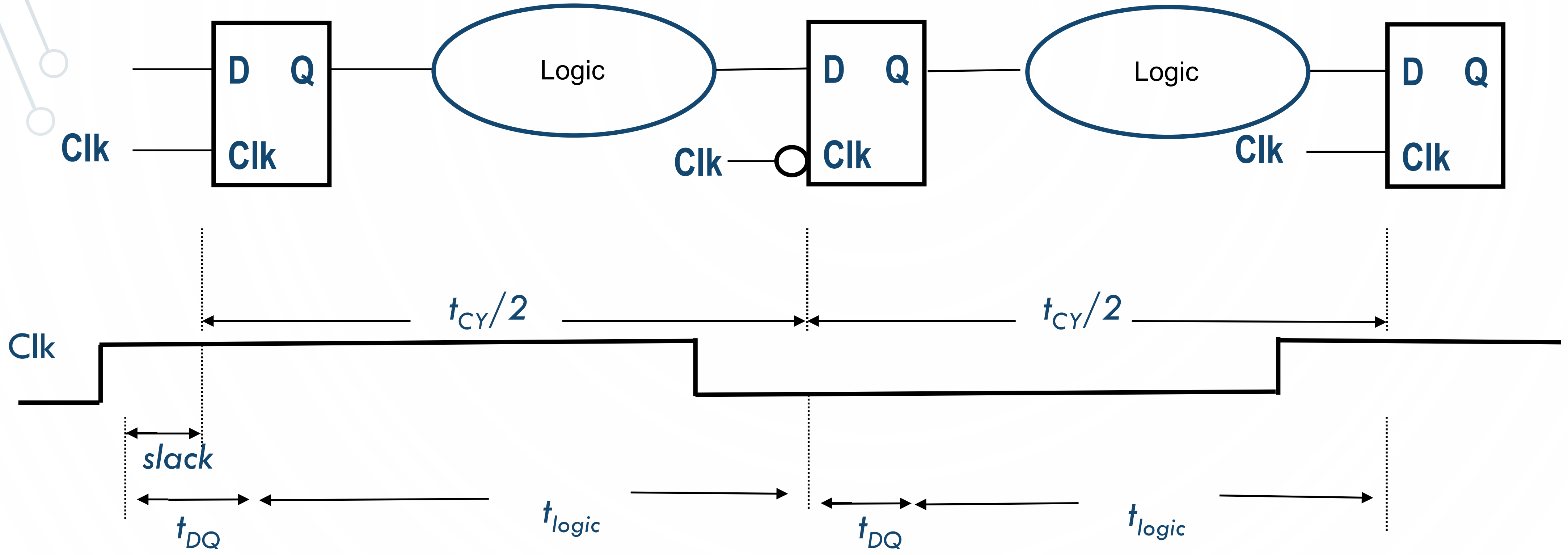
- Slack passing – logical partition uses left over time (slack) from the *previous* partition
- Time borrowing – logical partition utilizes a portion of time allotted to the *next* partition
- Makes most impact in unbalanced pipelines

Bernstein et al, Chapter 8, Chandrakasan, Chap 11 (by Partovi)

Slack Passing and Time Borrowing

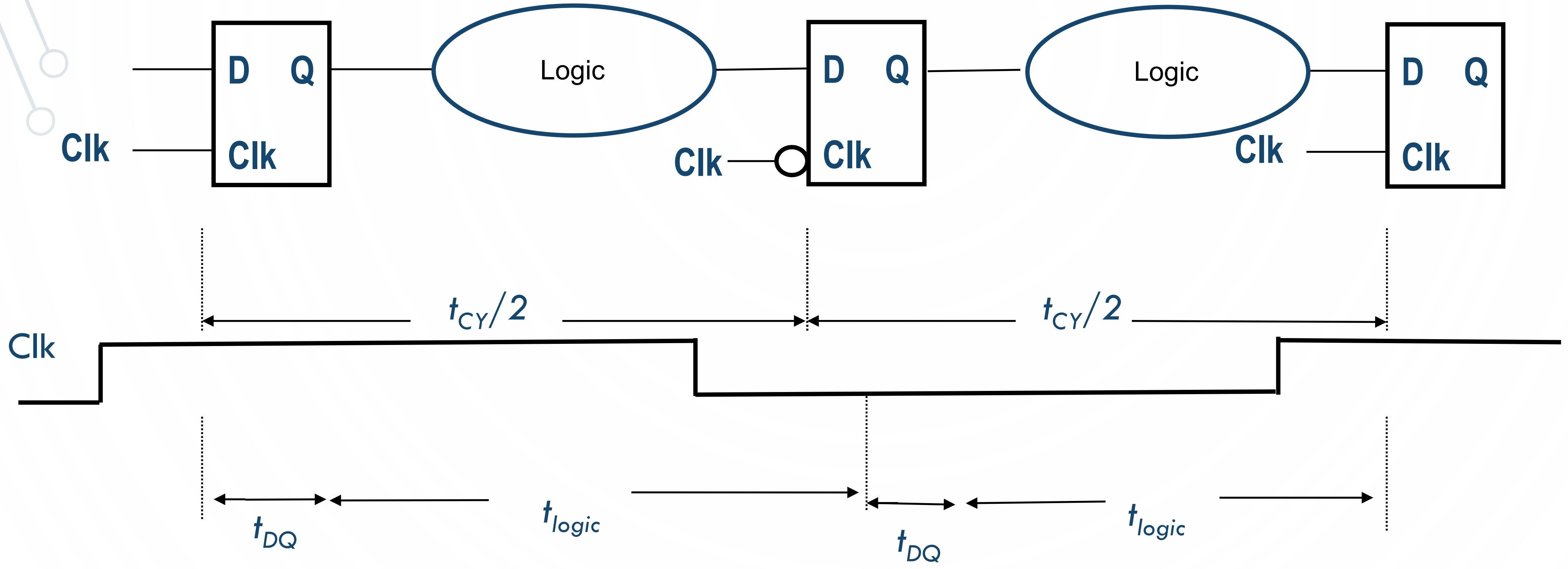


Slack Passing and Time Borrowing



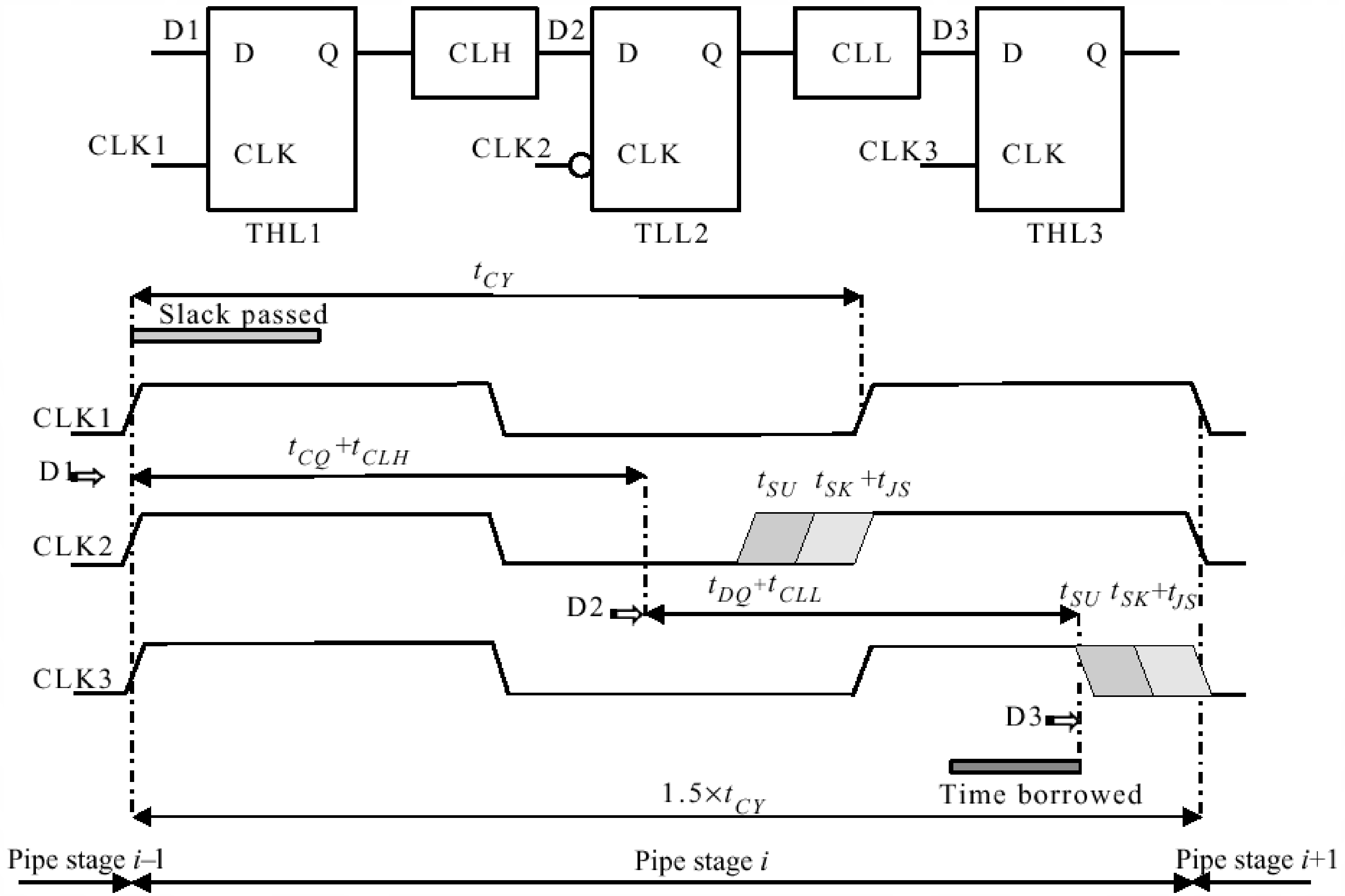
- Slack passed

Slack Passing and Time Borrowing



- Time borrowed

Slack-Passing and Cycle Borrowing



For N stage pipeline, overall logic delay should be < N Tcl

Summary

- Standard cell libraries
  - Linear model (not used anymore)
  - NLDM
  - CCS
  - Lots of options, details abstracted
- Timing
  - Flip-flop-based taming
  - Latch-based timing is more complex

Next Lecture

- Latches and flip-flops