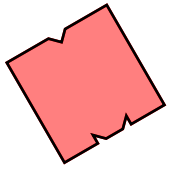
ENEE 359a
Digital VLSI Design

System Timing: Conventions, Problems, Solutions

Prof. Bruce Jacob

blj@ece.umd.edu

Credit where credit is due:
Slides contain original artwork (© Jacob 2004) as well as material taken liberally from Irwin & Vijay's CSE477 slides (PSU), Schmit & Strojwas's 18-322 slides (CMU), Dally's EE273 slides (Stanford), Wolf's slides for *Modern VLSI Design*, and/or Rabaey's slides (UCB). Asynchronous circuits: Erik Brunvand.

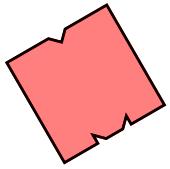


Bruce Jacob

University of
Maryland
ECE Dept.

Overview

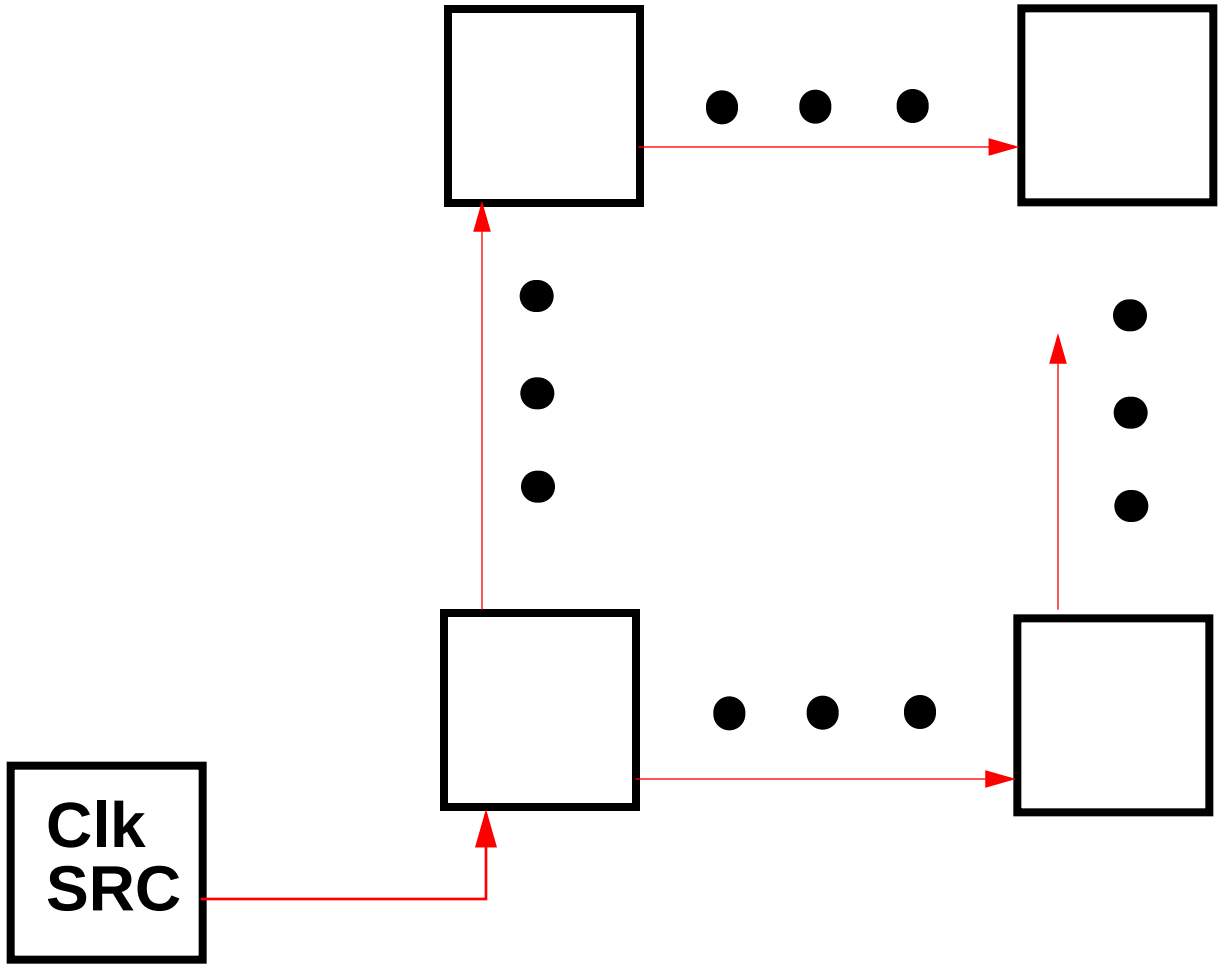
- **Motivation: What Needs To Be Done? (and Why It Matters) (Elmore primer)**
- **Timing Conventions**
- **Synchronous & Source-Synchronous**
- **Self-Timed Circuits**
- **Dealing with Problems of Global Clocks**
- **Balanced Trees**
- **The Wonderful World of DLLs/PLLs**



Bruce Jacob

University of
Maryland
ECE Dept.

What Needs To Be Done?



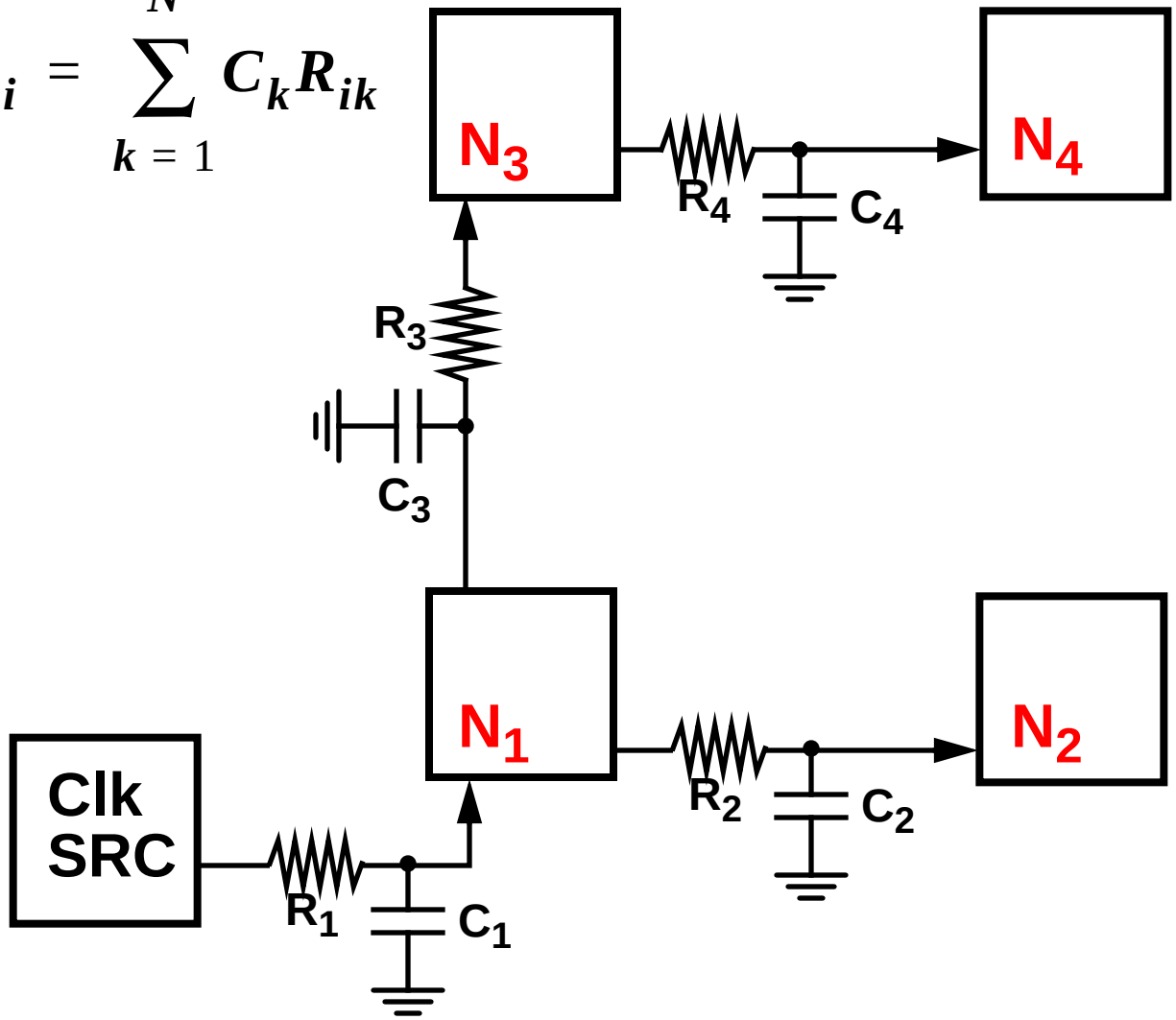
By the way: “global” clock scheme

Bruce Jacob

University of
Maryland
ECE Dept.

Background: Elmore Delays

$$\tau_{Di} = \sum_{k=1}^{N} C_k R_{ik}$$

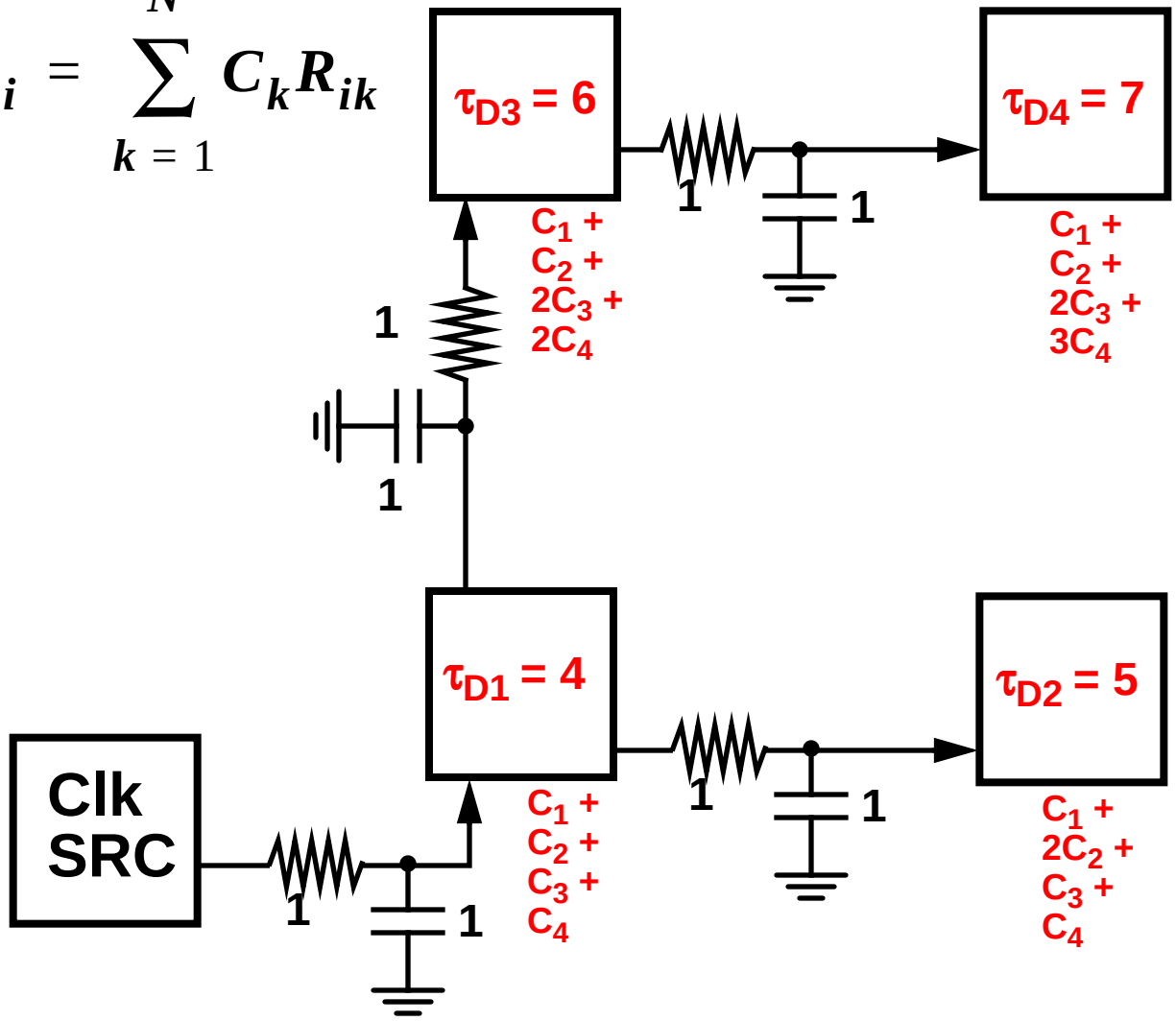


Bruce Jacob

University of
Maryland
ECE Dept.

Background: Elmore Delays

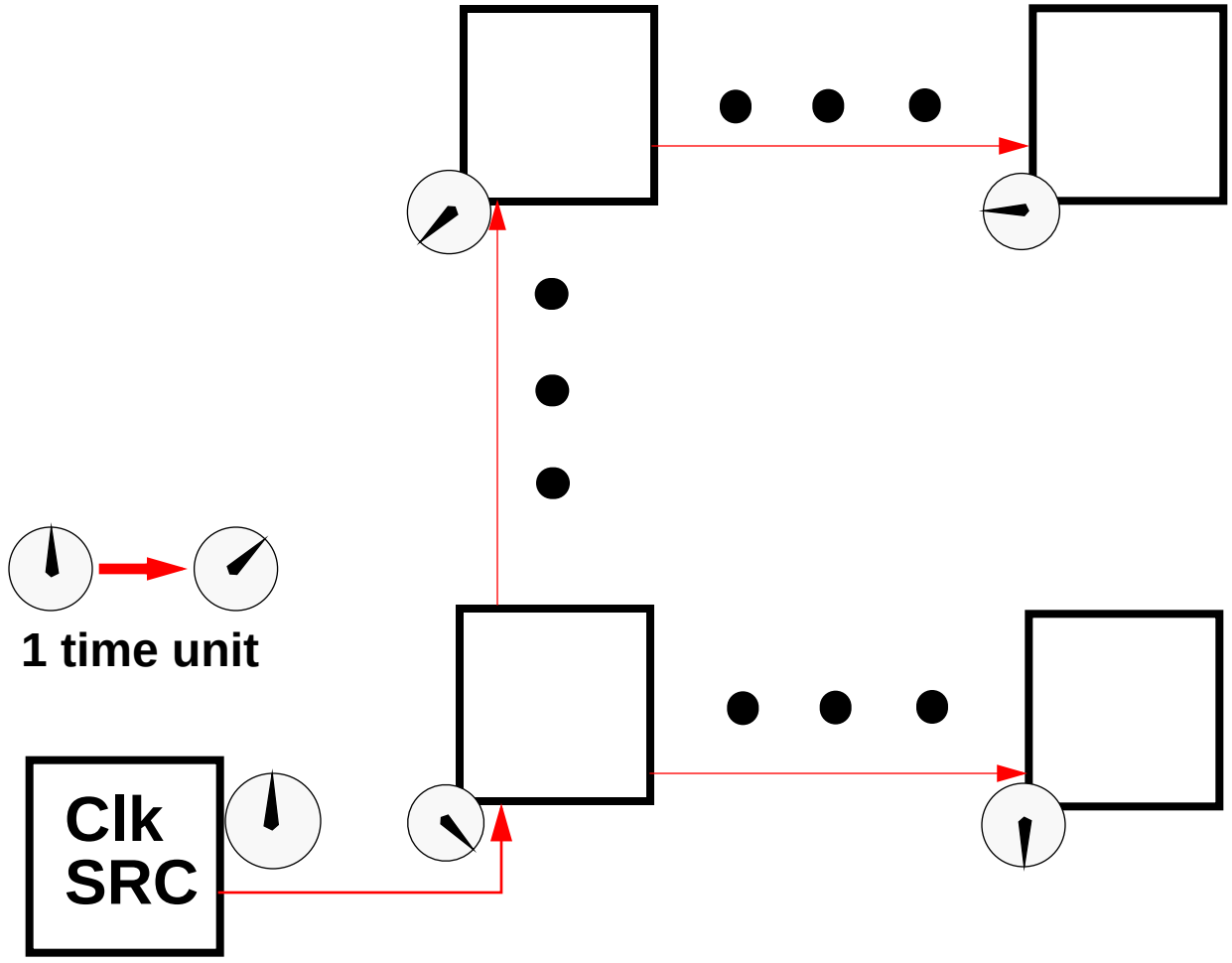
$$\tau_{Di} = \sum_{k=1}^{N} C_k R_{ik}$$



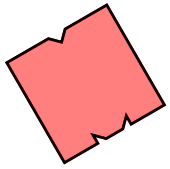
Bruce Jacob

University of
Maryland
ECE Dept.

End Result: *Clock Skew*



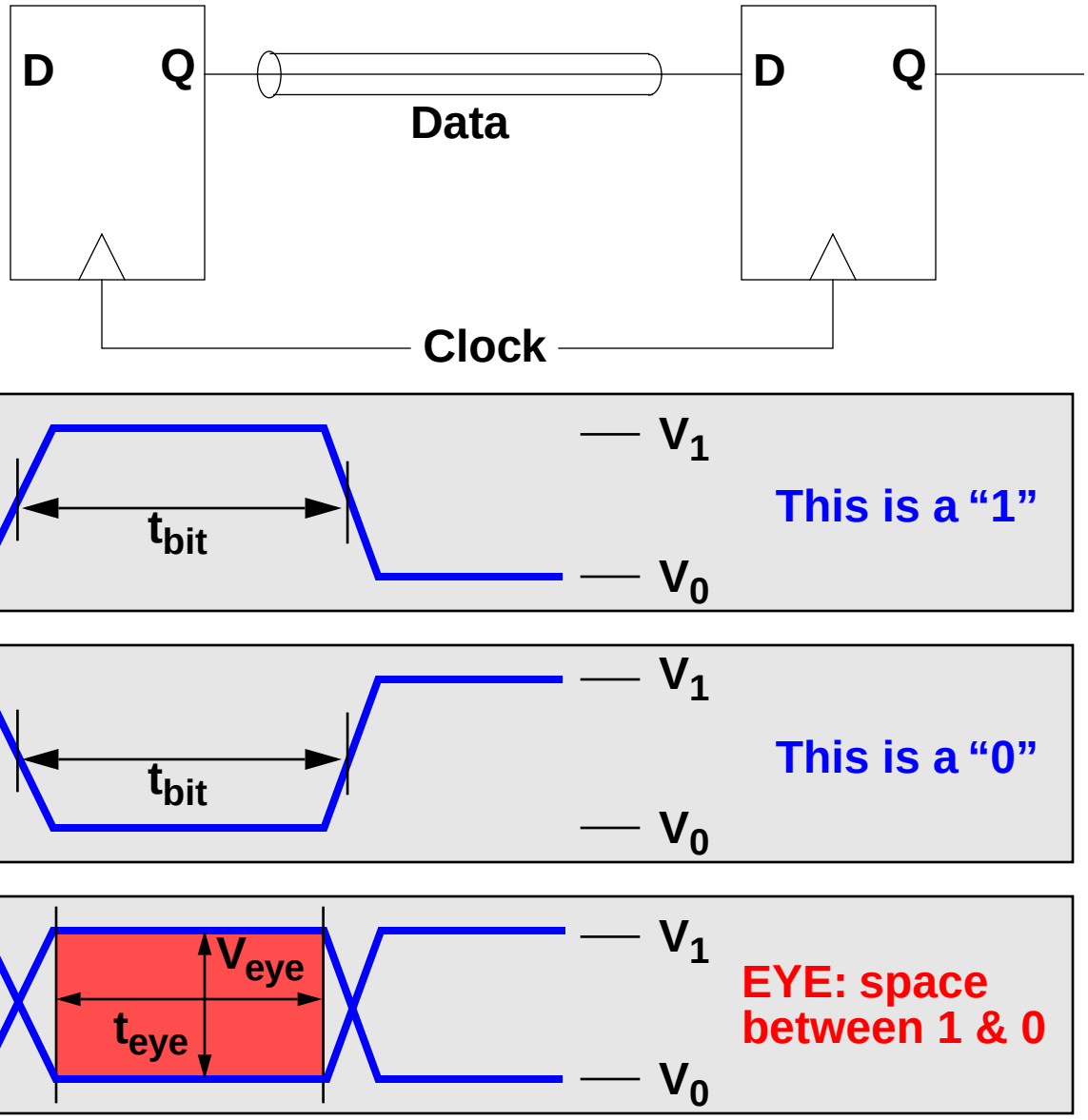
In general, even nearby clocks not in synch



Bruce Jacob

University of
Maryland
ECE Dept.

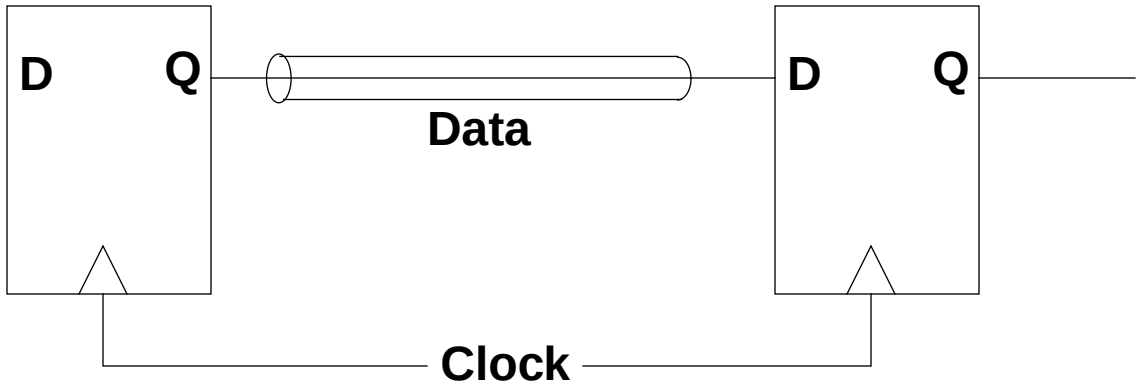
Idea: Detect “0” vs. “1”

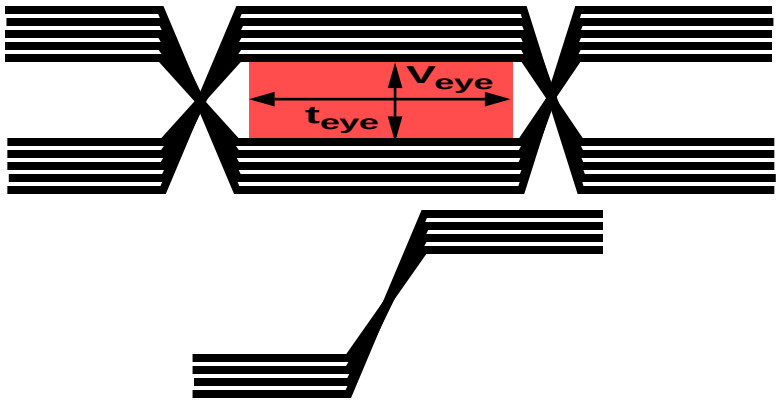


Bruce Jacob

University of
Maryland
ECE Dept.

Basic Problem: Voltage Noise

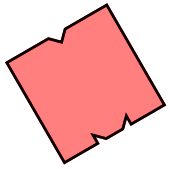




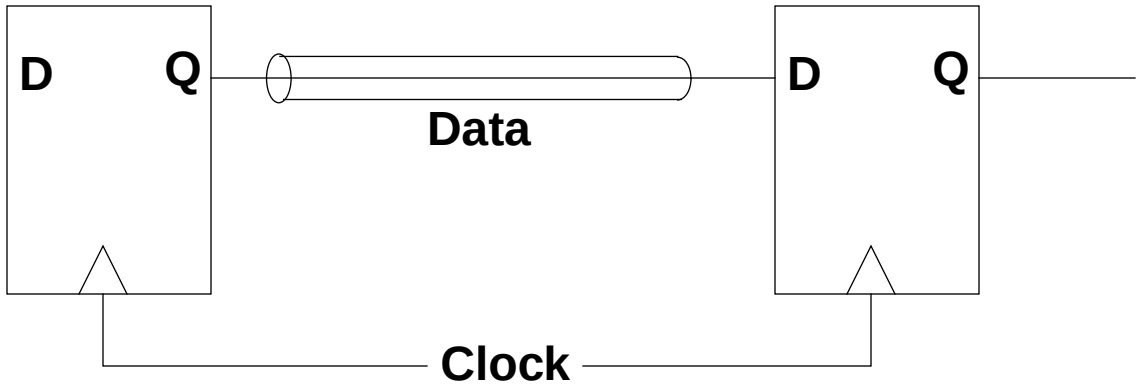
Data signal with voltage noise

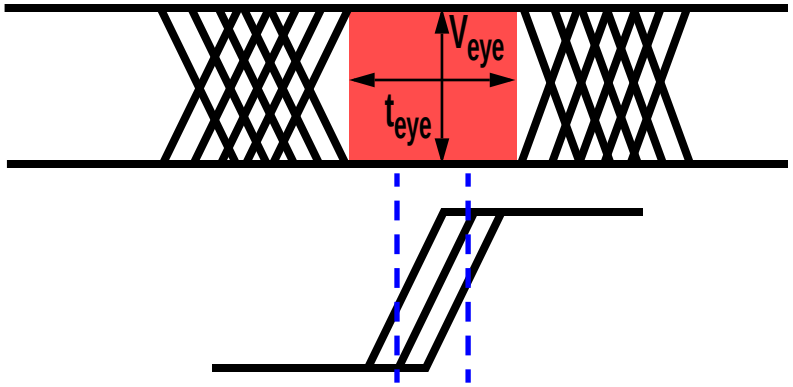
Clock signal with voltage noise

Voltage noise reduces operating margins



Basic Problem: Timing Noise

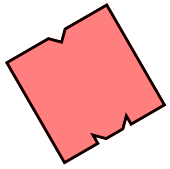




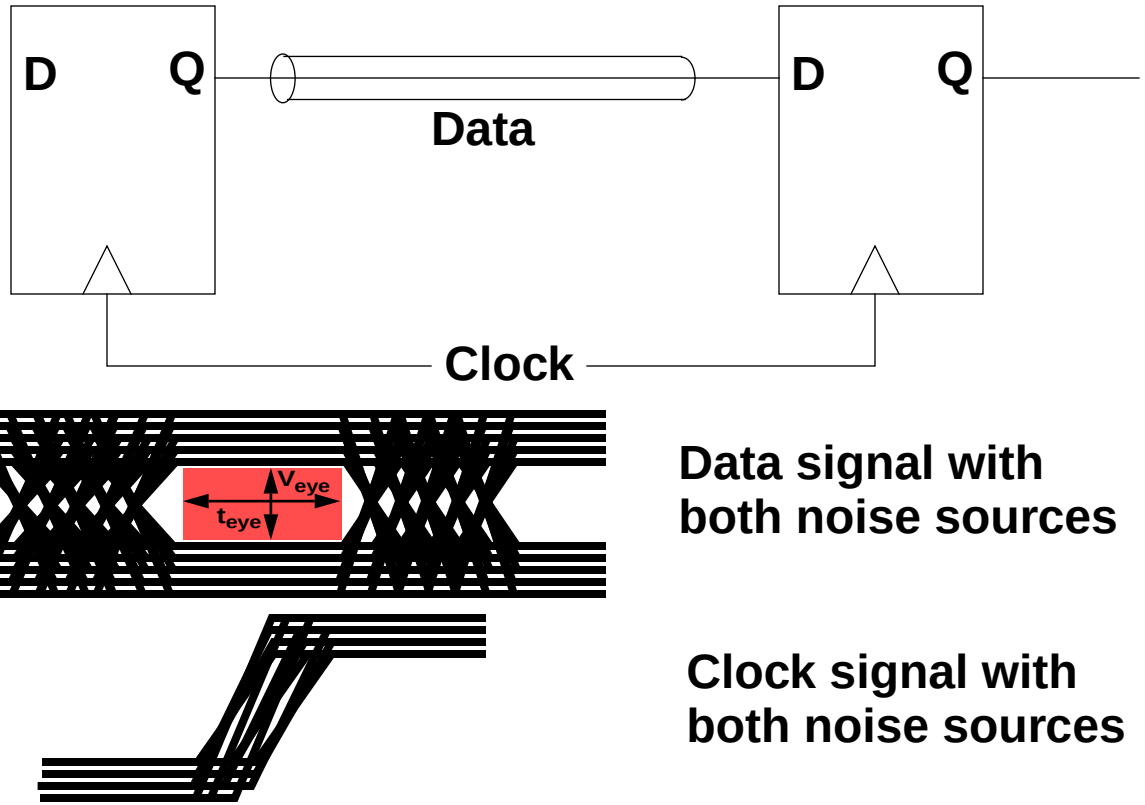
Data signal with timing noise

Clock signal with timing noise

Timing noise reduces operating frequency



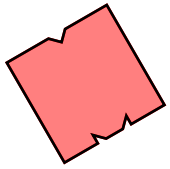
Basic Problem: Both



Data signal with both noise sources

Clock signal with both noise sources

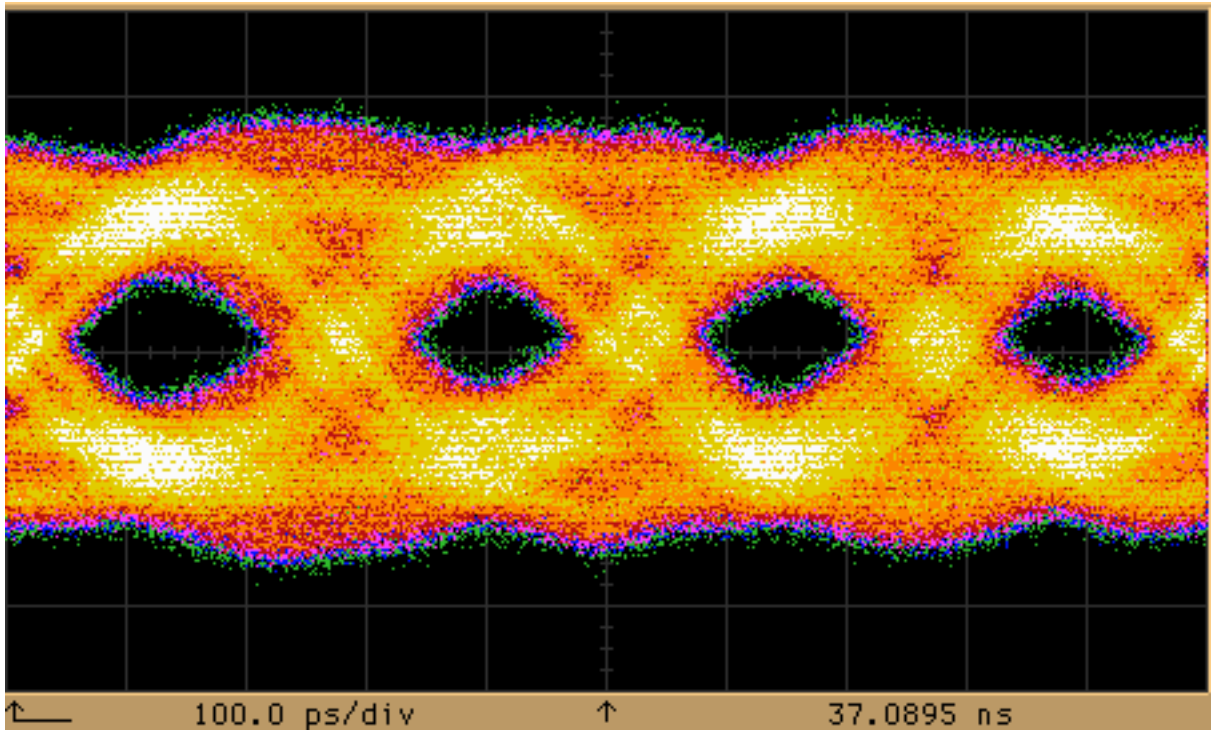
Note: Clock signal is just another signal subject to same constraints of voltage noise, skew and jitter as data signals



Bruce Jacob

University of
Maryland
ECE Dept.

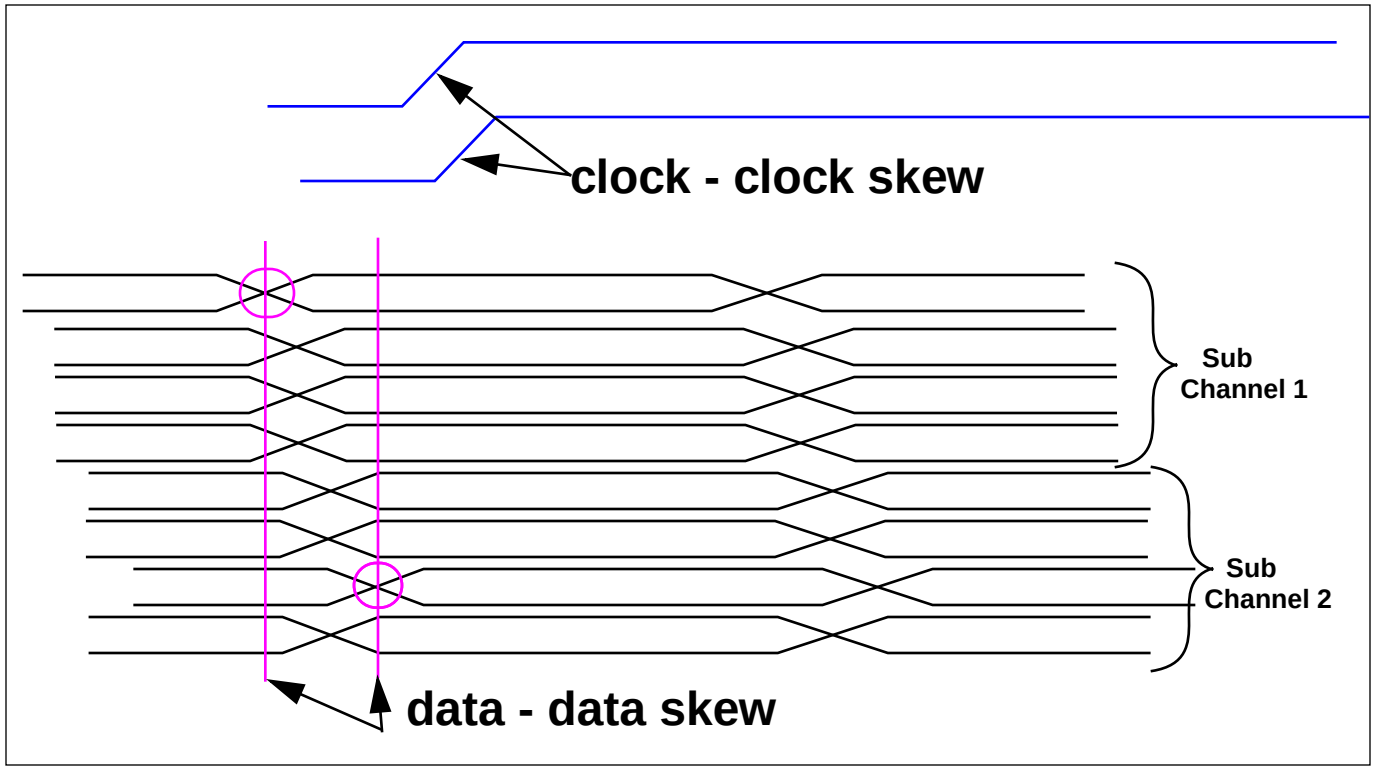
Eye Diagram



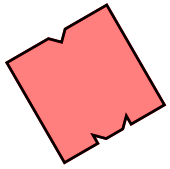
Yes, there really is that much voltage noise;
Yes, there really is that much timing noise …

Life sucks; deal with it.

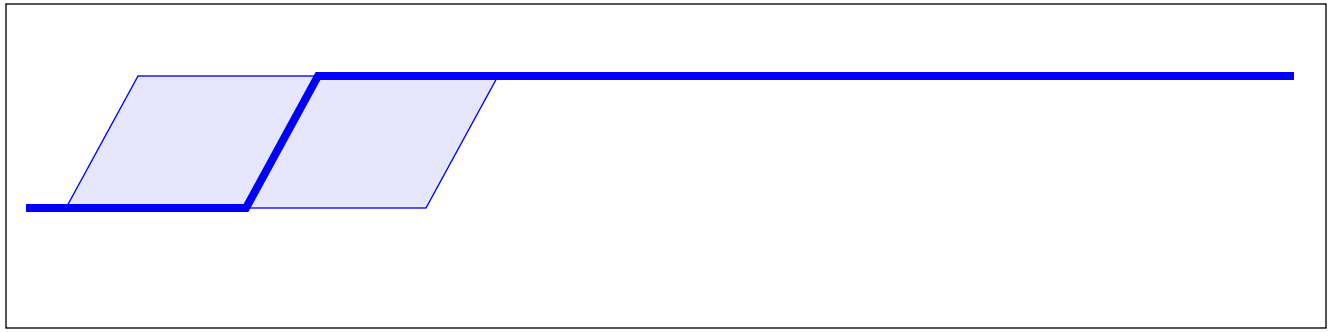
Definitions: *Skew*



- Static timing displacement from ideal design
- Caused by differences in signal path characteristics
- Total timing budget must take data-data skew, data-clock skew as well as clock-clock skew into cycle budget consideration

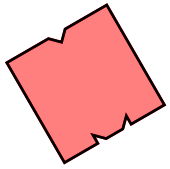


Definitions: *Jitter*

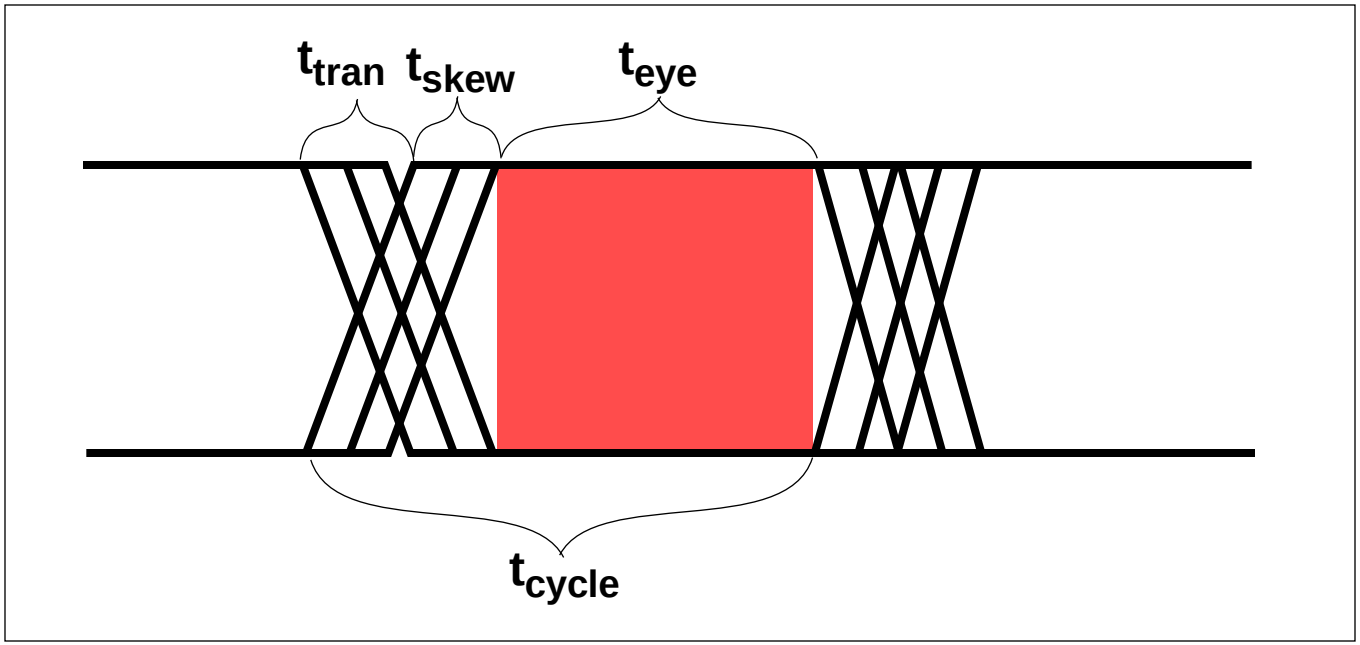


- Dynamic timing displacement from nominal timing characteristics

- Magnitude and offset of timing displacement could depend on: previous signal state(s), current signal state(s), supply voltage level(s), crosstalk, variations in thermal characteristics. Perhaps even phases of the moon (not proven).



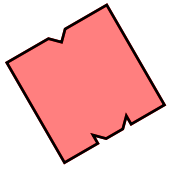
Cycle Budget



t_{skew} = max(skew) + max(jitter)

t_{tran} = Edge transition time = max(rise_time, fall_time)

$t_{eye} \leq t_{cycle} - t_{tran} - t_{skew}$



Timing: Clock ∩ Data

Define *when* a value is present on a line …

How many 1's? How many 0's?

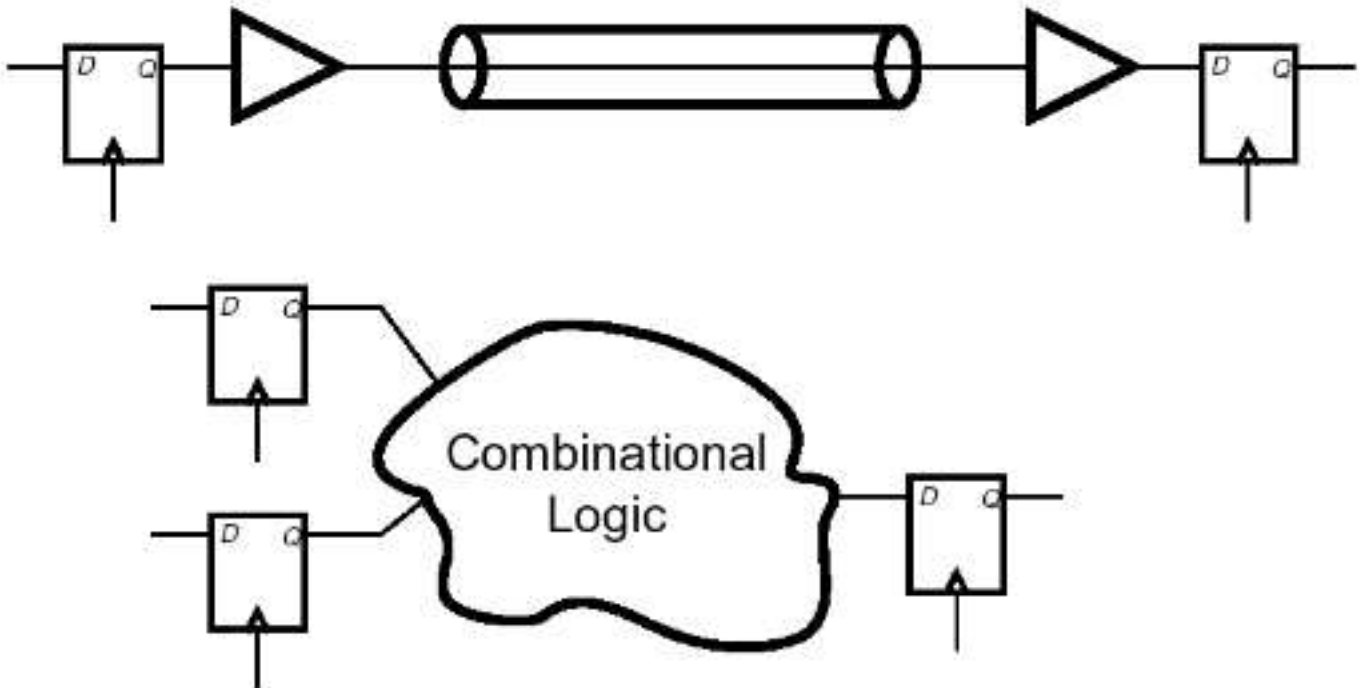
Need convention to distinguish where one '1' ends and next '1' begins … conventions typically mark boundaries w/ *TRANSITIONS*

- **of the signal itself**
- **of an associated *clock* signal**

(original definition of "synchronous")

Uncertainty in timing limits operating speed

Timing Conventions

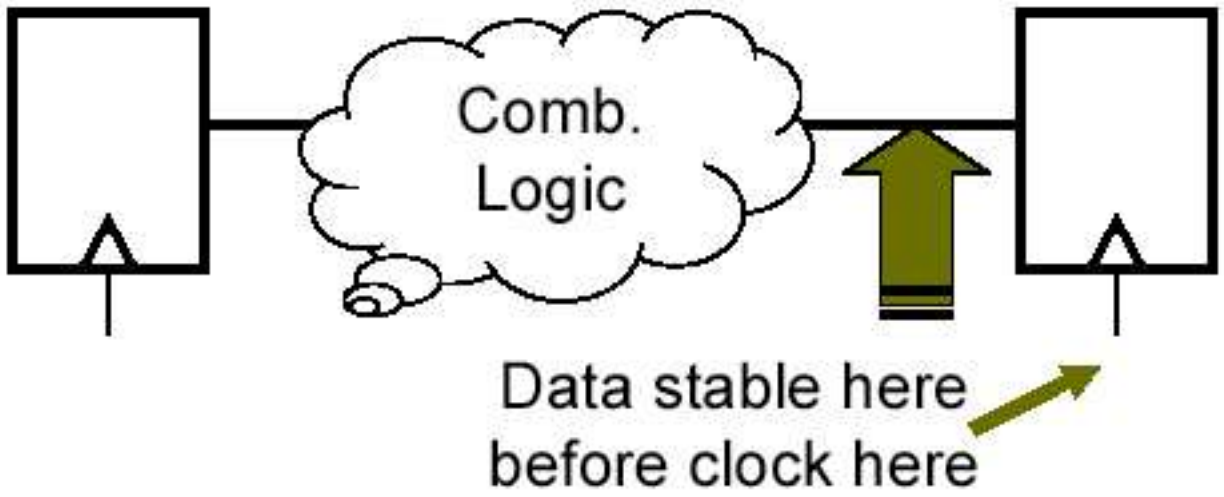


Apply to both inter- *and* intra-chip signaling

Setup Time

Required time for input to be stable

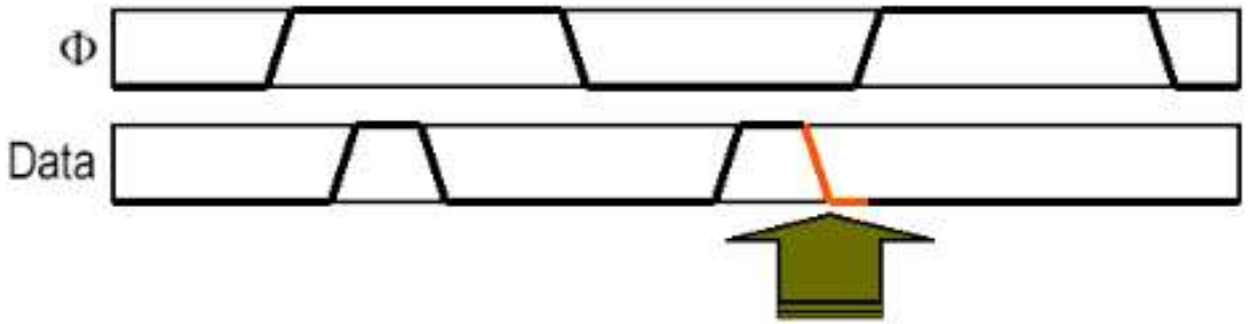
BEFORE CLOCK EDGE



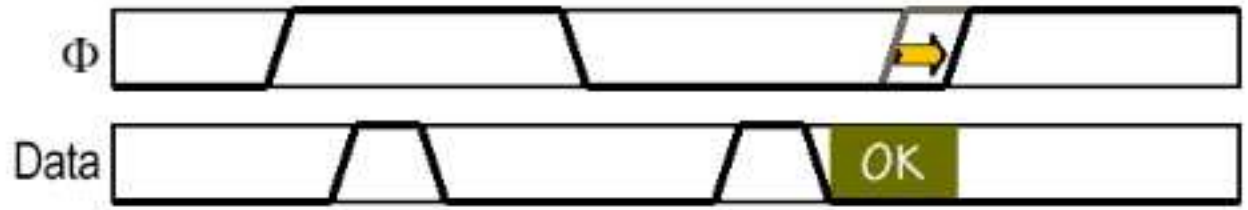
Bruce Jacob

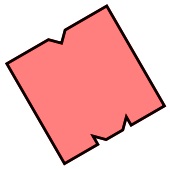
University of
Maryland
ECE Dept.

Setup Time Fix



This violation can be fixed by stretching the clock cycle

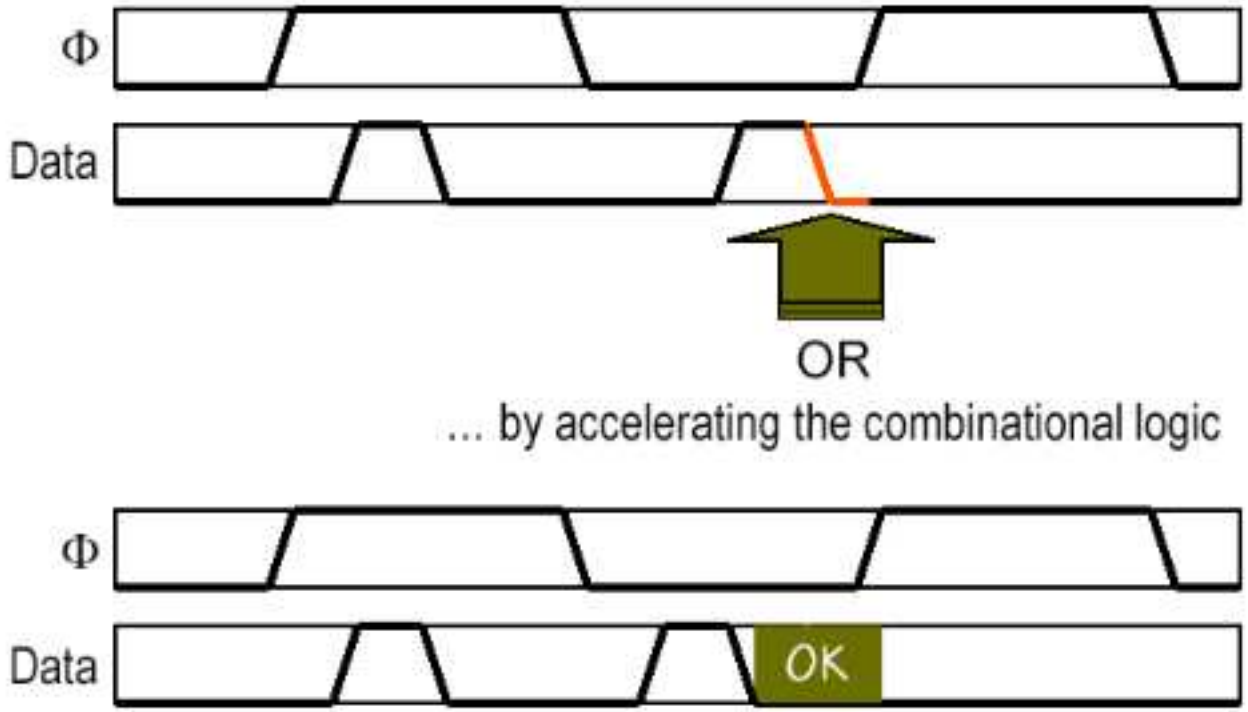


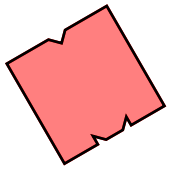


Bruce Jacob

University of
Maryland
ECE Dept.

Setup Time Fix II

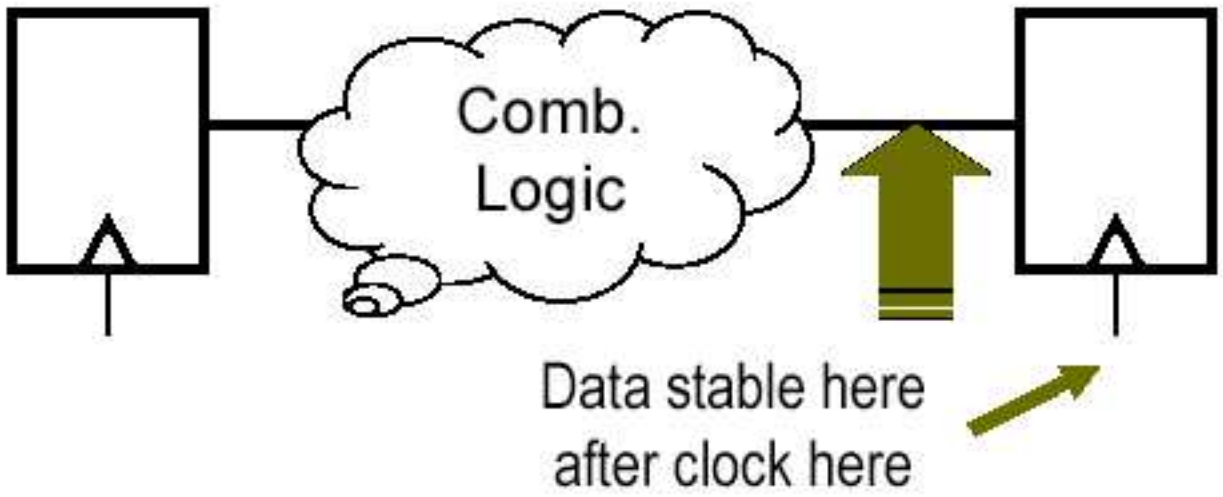




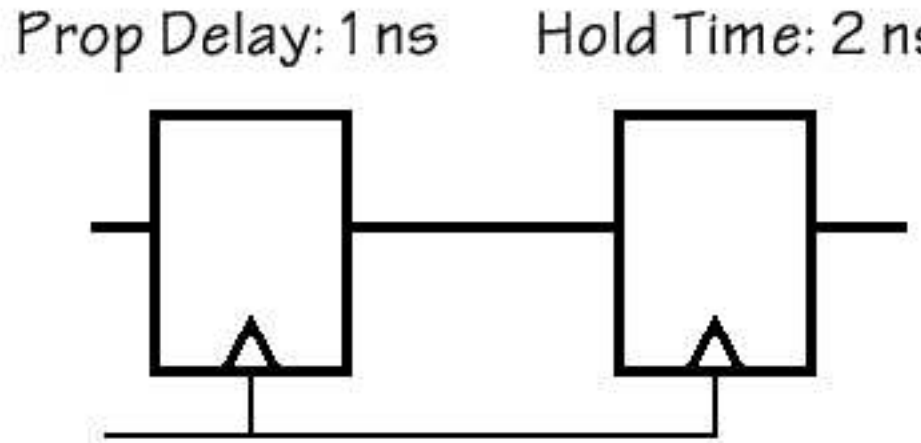
Hold Time

Required time for input to be stable

AFTER CLOCK EDGE



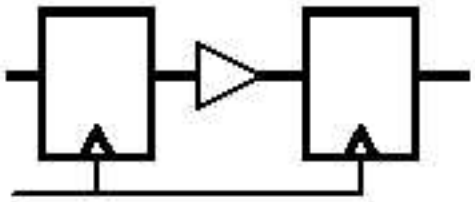
Hold Time Violations



Hold time violations are caused by "short paths"

Cannot be fixed by slowing down the clock!!!

Fixed by slowing down fast paths



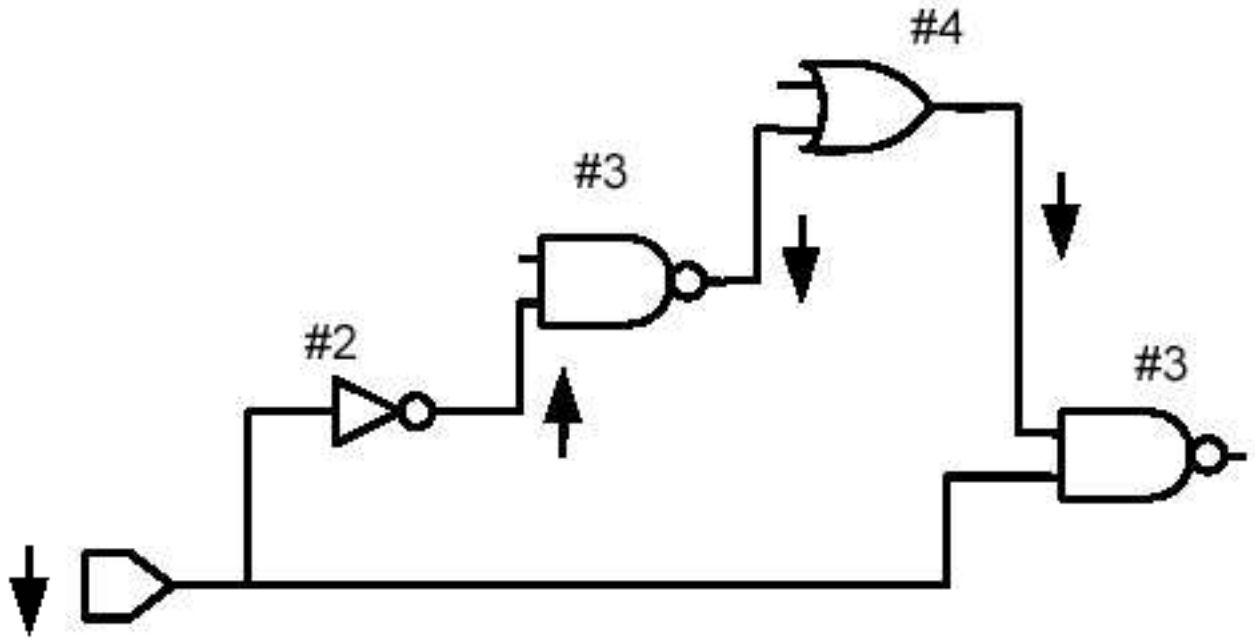
Bruce Jacob

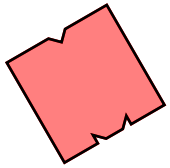
University of
Maryland
ECE Dept.

Basic Timing Analysis

Look for *longest* path: clock speed
Look for *shortest* path: check hold time

Difficult problem, e.g. False Paths





Bruce Jacob

University of
Maryland
ECE Dept.

A Tale of Two (or more) Timing Conventions

Synchronous: global clock

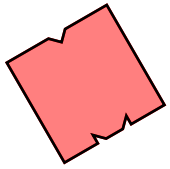
Synchronous: source-synchronous I

("open-loop" meaning no control loop)

Synchronous: source-synchronous II

("closed loop" meaning feedback control)

Asynchronous: self-timed

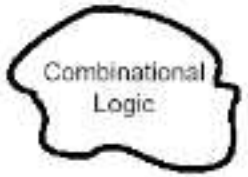


System Building Blocks

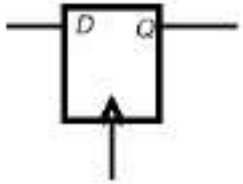
DELAY ELEMENTS

- Nominal delay
- Timing uncertainty, skew, jitter

COMBINATIONAL LOGIC

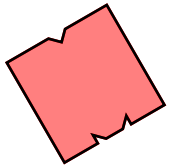


- Contamination delay
- Propagation delay



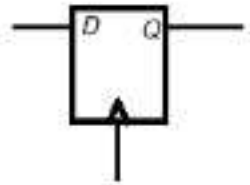
CLOCKED STORAGE ELEMENTS

- Align signal to a clock
- Signal waits to be sampled by clock
- Output held steady until next clock

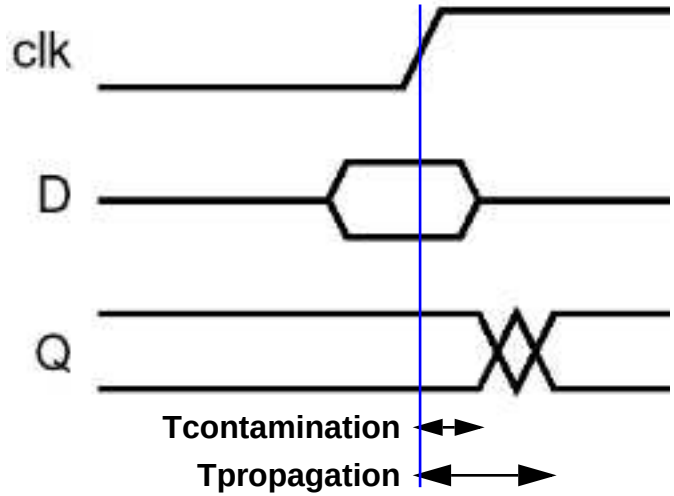


Edge-Triggered Register

Samples data on rising edge of CLK

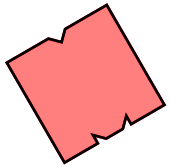


- **Data must remain valid during an *aperture* of time during sampling**



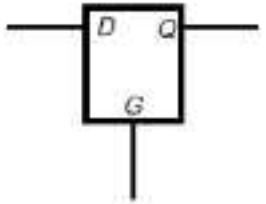
Output held steady until next CLK edge

- **Output is held until a *contamination delay* following CLK edge**
- **Output has a correct value after a *propagation delay* following CLK edge**

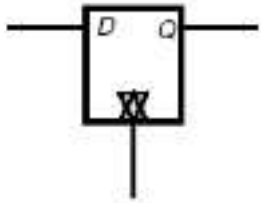


Other Storage Elements

LEVEL-SENSITIVE LATCH

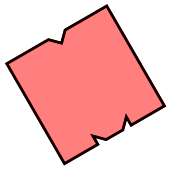


- **Passes data through when enable (clock) is high**
- **Holds data stable when enable (clock) is low**



DUAL-EDGE-TRIGGERED FLIP-FLOP/REGISTER

- **Samples data at both edges of the clock**
- **Internally two interleaved flip-flops (1 posedge FF + 1 negedge FF)**
- **Allows CLK to run at same speed as data**

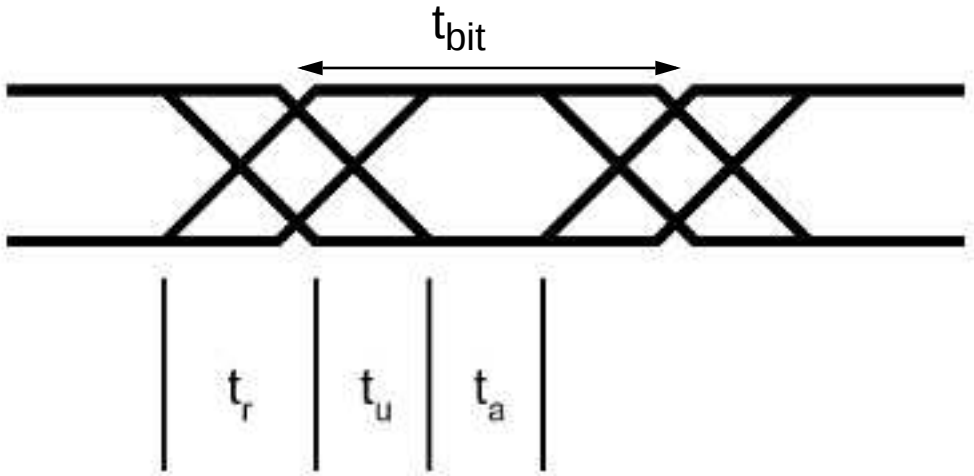


Eye Diagram, Redux

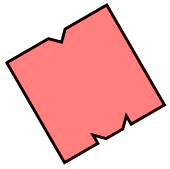
MAXIMUM OPERATING RATE

(i.e., signaling speed)

LIMITED BY THREE FACTORS:

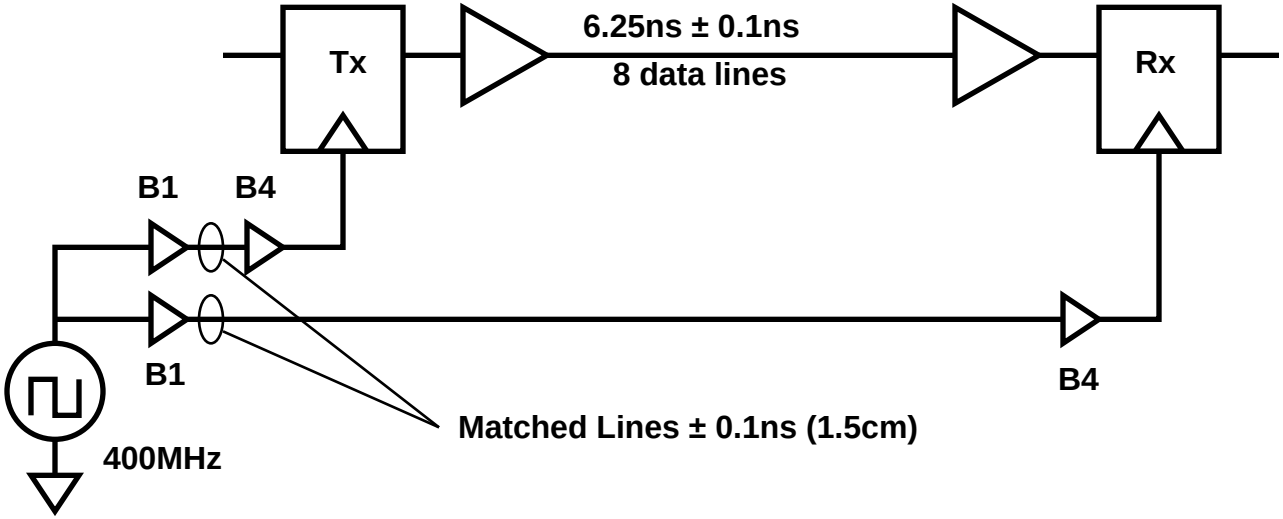


- **Tr — Transition time (rise/fall time)**
- **Tu — Timing uncertainty, skew, jitter**
- **Ta — Aperture time**
- **Tbit ≥ Tr + Tu + Ta *(but not that simple …)***

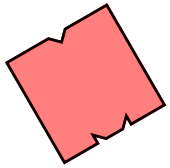


Synchronous Timing I

GLOBAL CLOCK (Conventional) EXAMPLE

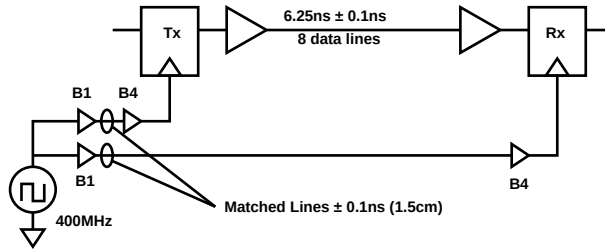


Parameter	Symbol	Nominal	Skew	Jitter
Bit Cell (data period)	Tbit	2.5 ns		
Transmitter Rise Time	Tr	1.0 ns		
Cable Delay	Twire	6.25 ns	100 ps	
Receiver Aperture	Ta	300 ps	100 ps	50 ps
Transmitter Delay		500 ps	150 ps	50 ps
Buffer Stage Delay	B#	250 ps	100 ps	50 ps



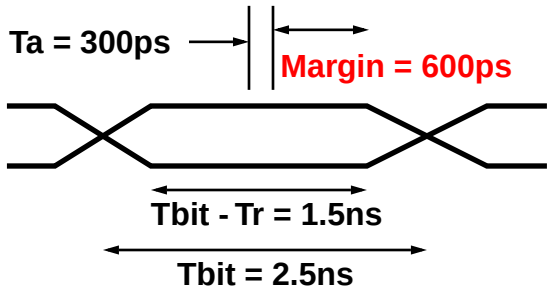
Synchronous Timing I

Sources of uncertainty:

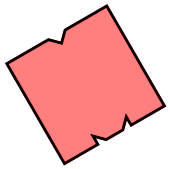


- **Skew (across multiple lines) of line delay**
- **Jitter of Tx, Rx, and line delay**
- **Skew and jitter of global clock (usually large due to high fan-out)**

For best performance, center sampling edge on data eye

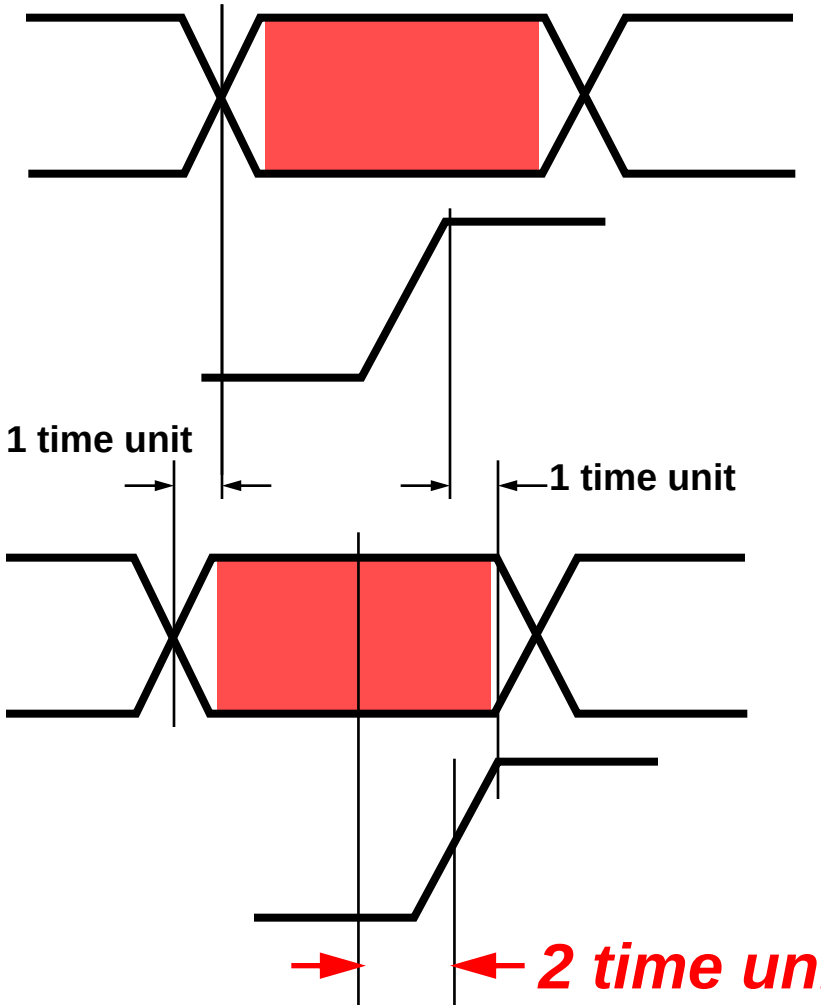


Gross timing margin
= 1/2 [Tbit - Tr - Ta] = ± 600ps = *1/2 Tu?*



An Aside ...

Why do we simply add up the uncertainties?
And why does each effectively count twice?



DATA w/ no skew/jitter

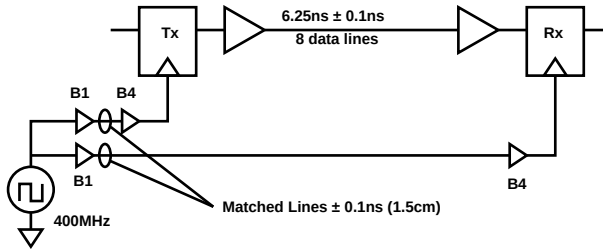
CLOCK w/ no skew/jitter

DATA *early* by 1 time unit

CLOCK *late* by 1 time unit

Synchronous Timing I

TIMING ANALYSIS



Clock Skew: 100ps lines + 100ps B1 + 400ps B4

Clock Jitter: 50ps B1 + 200ps B4

[CLKs times 2: one for xmit, one for recv]

Transmitter: 150ps skew, 50ps jitter

Receiver: 100ps skew, 50ps jitter

Data Cable: 100ps skew

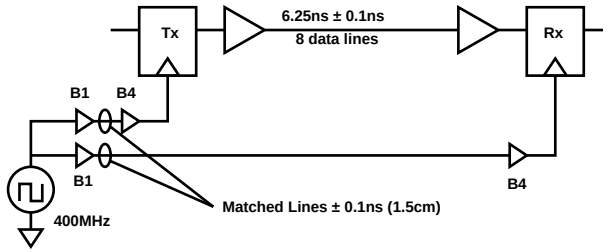
TOTAL: 1550ps skew, 600ps jitter (BAD)

Bruce Jacob

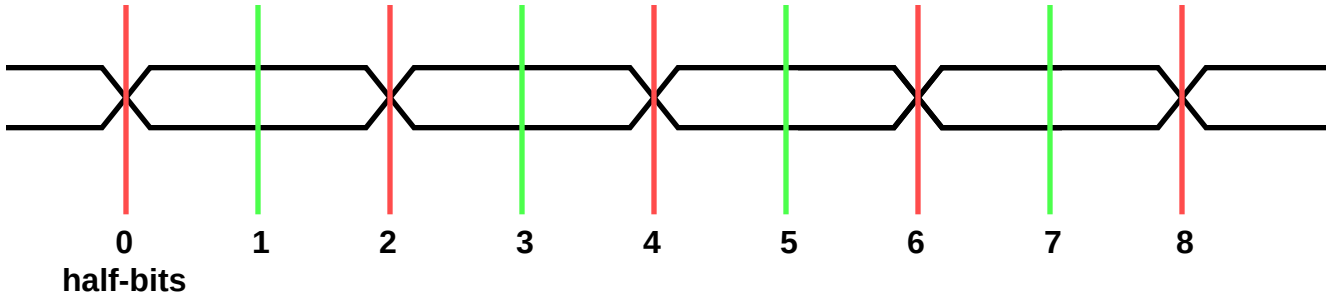
University of
Maryland
ECE Dept.

Synchronous Timing I

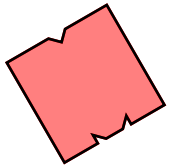
LIMITS TO LINE DELAY & DATA FREQUENCY



Conventional Wisdom:
Line Delay Must Be
ODD NUMBER of HALF-BITS … WHY?

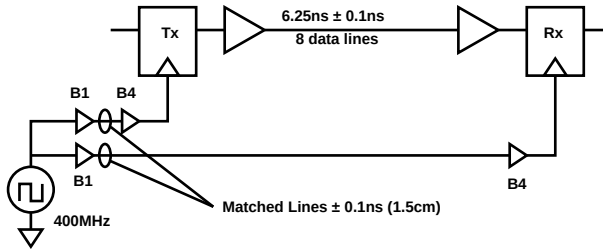


For fixed line length and tight margins, this limits the bus speeds that can be used



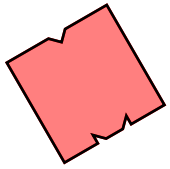
Synchronous Timing I

SUMMARY



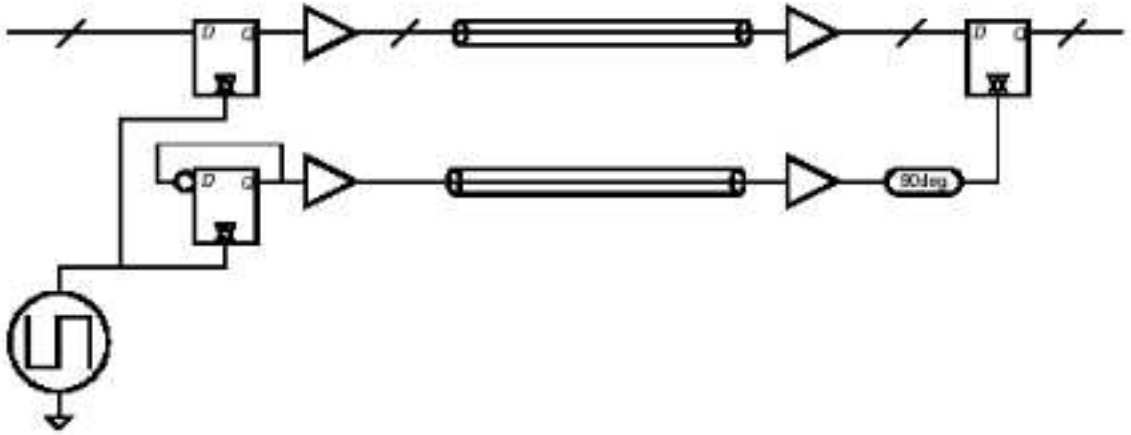
GLOBALLY SYNCHRONOUS DESIGN:

- **For long wires and high speeds, only a handful of frequencies work**
- **Impractical to control uncertainties**
- **Cannot switch frequencies**



Synchronous Timing II

PIPELINED TIMING: BASIC IDEA



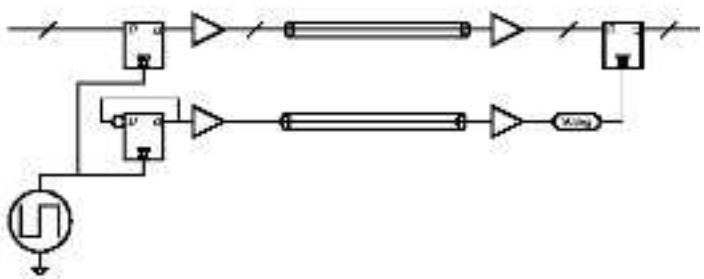
Delay the clock by the same amount as data … PLUS half a bit-cell

System will work from DC to maximum theoretical frequency 1/(Tr + Tu + Ta)

Defines new clock domain at receiving end

Synchronous Timing II

SOURCES OF UNCERTAINTY

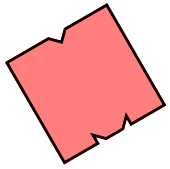


SKEW:

- Between CLK & Data line
- Fixed differences in FF, Tx, Rx delays
- Different CLK delays to different FFs
- Aperture offset in Rx FF
- Extra offset in the delayed CLK line

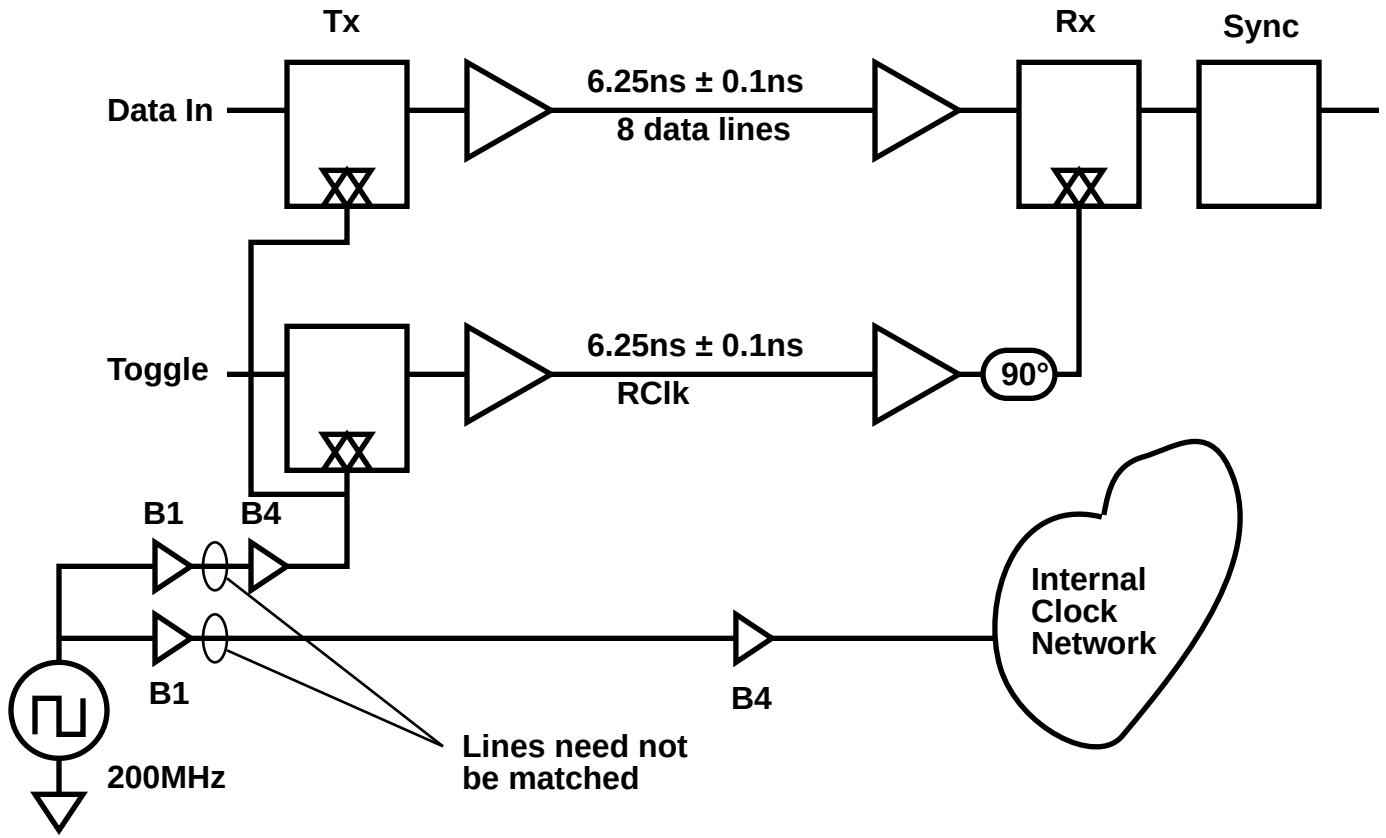
JITTER:

- In Tx clock
- In FF, Tx, Rx delays



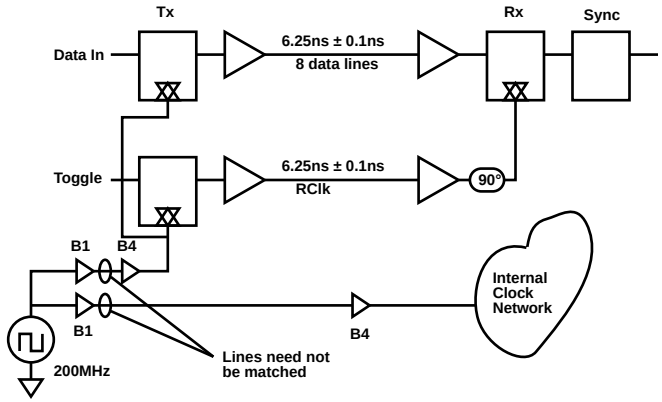
Synchronous Timing II

OPEN-LOOP PIPELINED EXAMPLE



Synchronous Timing II

TIMING ANALYSIS



Xmit data: 150ps skew, 50ps jitter

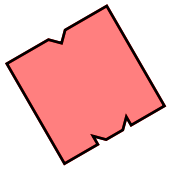
Xmit toggle: 150ps skew, 50ps jitter

Receiver: 100ps skew, 50ps jitter

Data cable: 100ps skew

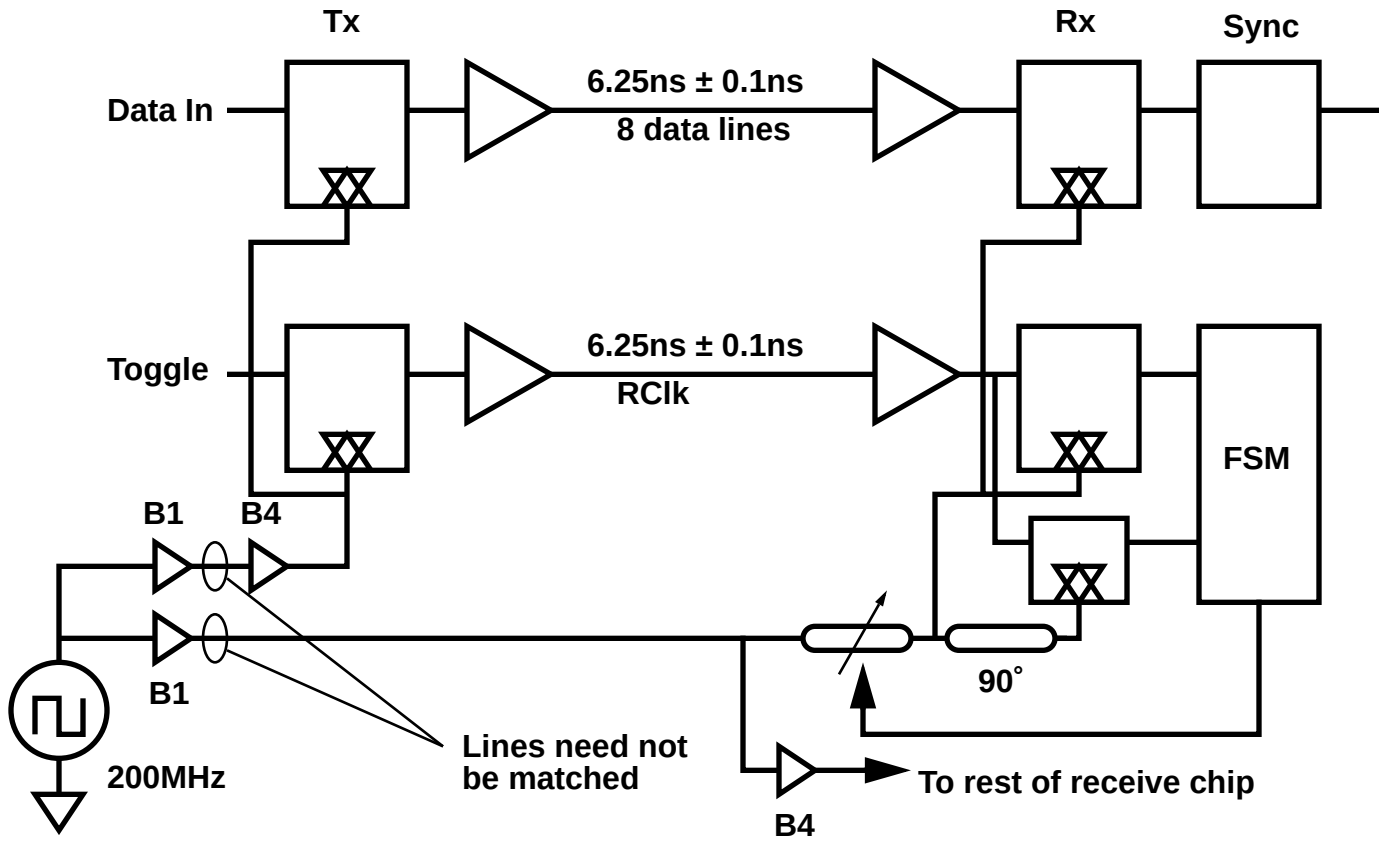
Toggle clock cable: 100ps skew

TOTAL: 600ps skew, 150ps jitter (BETTER)



Synchronous Timing III

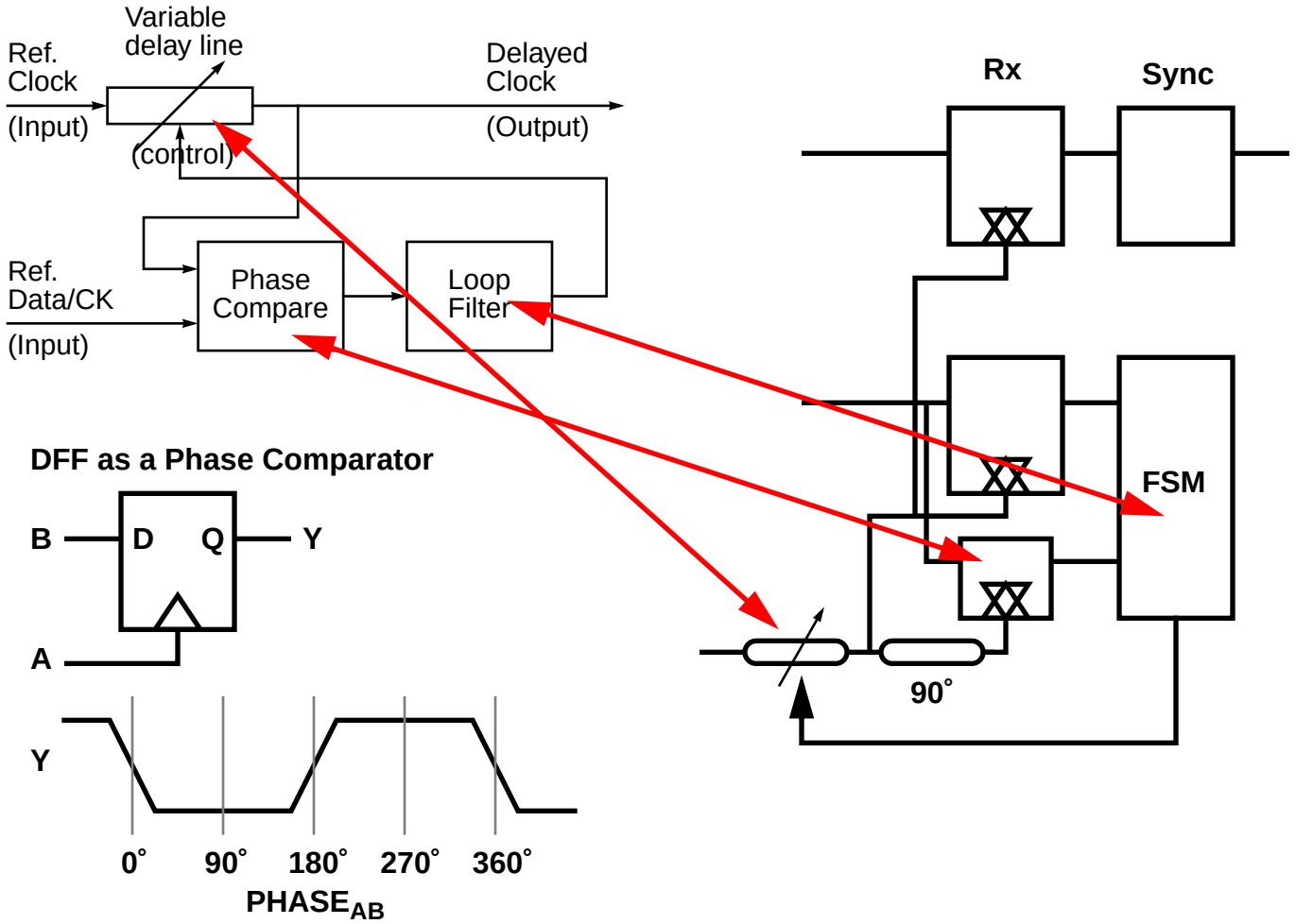
CLOSED-LOOP PIPELINED EXAMPLE



Variable delay line can cancel ALL SKEW

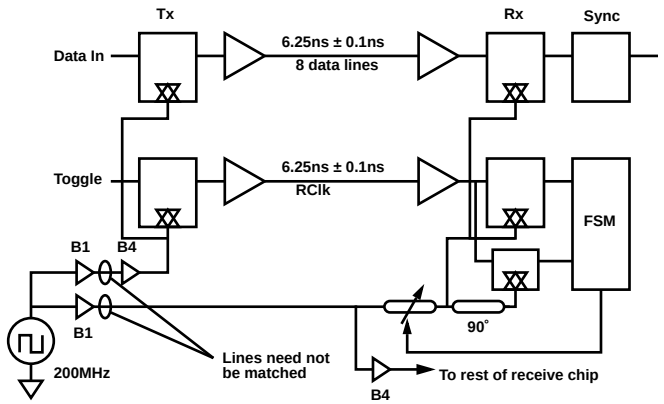
Synchronous Timing III

COMPONENTS of CONTROL LOOP (DLL)



Synchronous Timing III

TIMING ANALYSIS



Xmit data: 50ps jitter

Recv data: 50ps jitter

Xmit toggle: 30ps
skew data/toggle (0?), 50ps jitter

Recv toggle: 20ps skew (0?), 50ps jitter

Data cable: 100ps skew

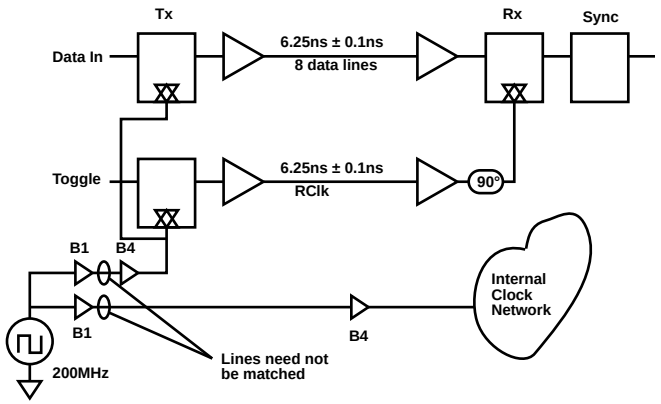
Toggle clock cable: 100ps skew

TOTAL: 250ps skew, 200ps jitter (GOOD!)

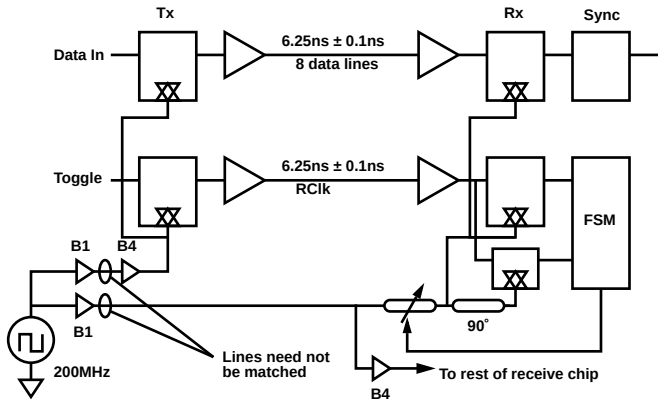
Synchronous Timing II & III

LIMITS TO LINE DELAY & DATA FREQUENCY

None.



Only limiter to bus frequency is the rate at which you can successfully transmit & receive data (e.g. Taperture + Tuncertainty + Ttransmit)

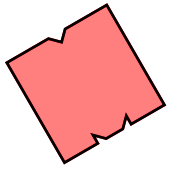


Asynchronous Timing

Basic Idea: no clocks

ADVANTAGES:

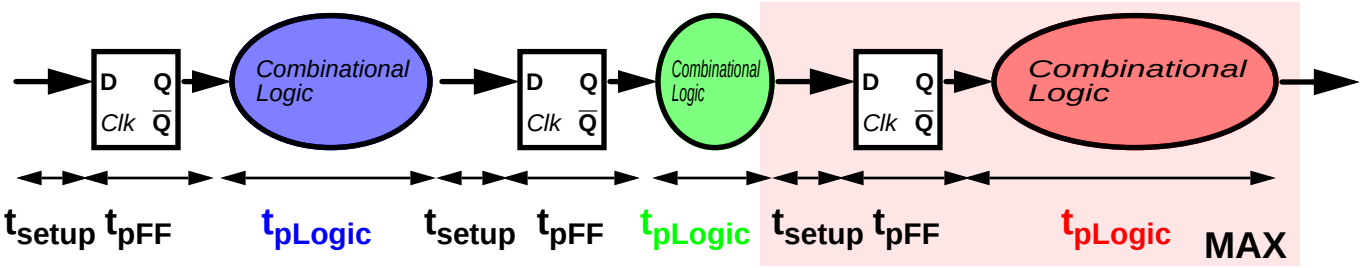
- **Achieve average-case performance**
  (good if difference between average & worst case is large)
- **Consume power only when needed**
  (i.e., only when actually processing data)
- **Easy modular composition**
  (designer focuses on local issues, not global issues)
- **No clock alignment required**
  (no expensive DLLs/PLLs)
- **No clock distribution headaches**
  (saves design time, power consumption, chip area)
- **Robust in the face of parameter variations**
  (e.g., temperature/voltage fluctuations, process variations)
- **Global synchrony is a fallacy anyway!**
  (i.e., face the problem head-on)



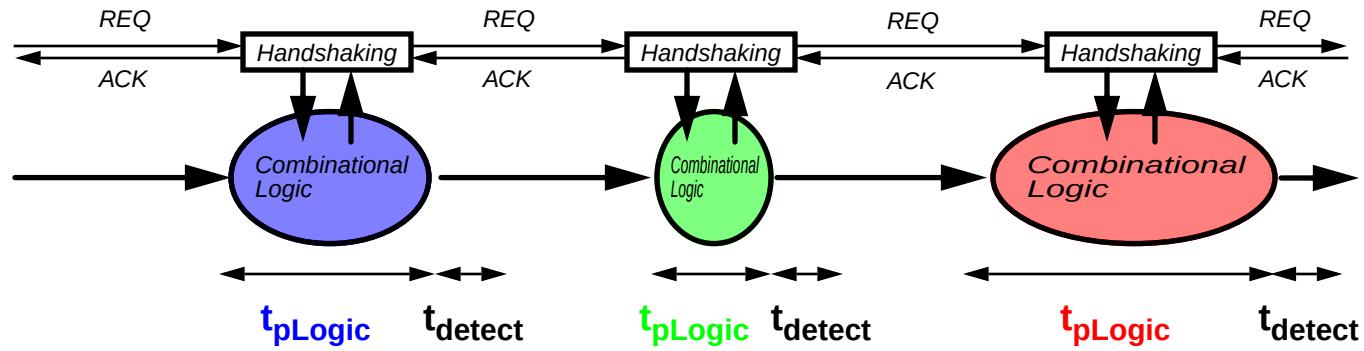
Asynchronous Timing

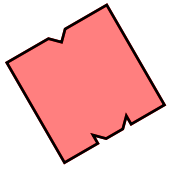
SELF-TIMED CIRCUITS

Worst-Case Delay: sets clock period in synchronous designs (time: N * max delay)



Average-Case Delay: asynchronous designs achieve it (time: $delay_1$ + $delay_2$ + $delay_3$ …)



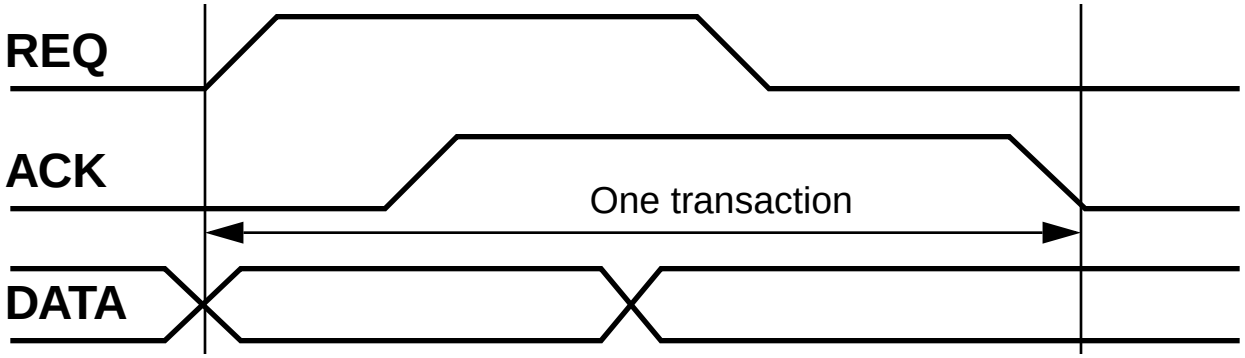


Asynchronous Timing

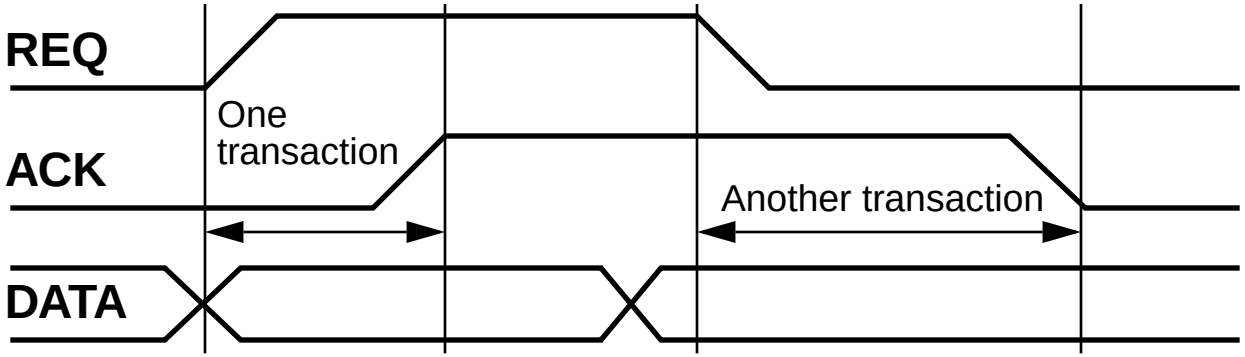
HANDSHAKING PROTOCOLS

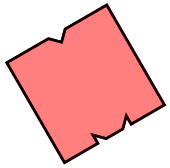
Four-Phase / RTZ / Level Signaling

- specific protocol determines data-release point



Two-Phase / NRTZ / Transition Signaling

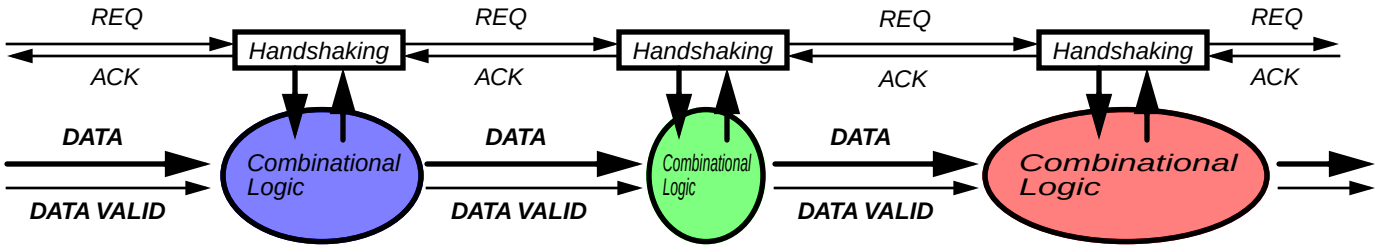




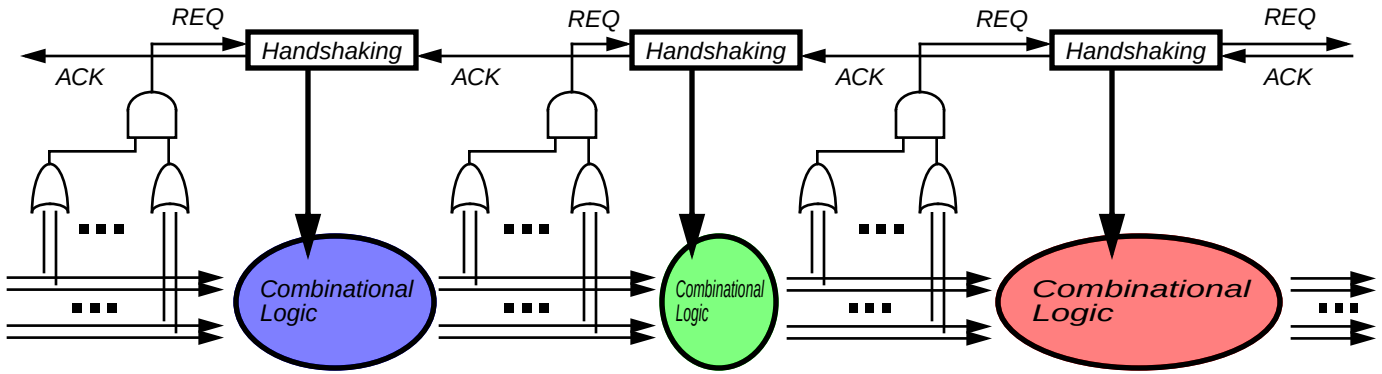
Asynchronous Timing

DATA SIGNALING

Bundled Data: "normal" data wires, one per bit, with associated "valid" signal



Dual-Rail Data: two wires per bit, encoded (00 = no data, 01 = 0, 10 = 1, 11 = error)

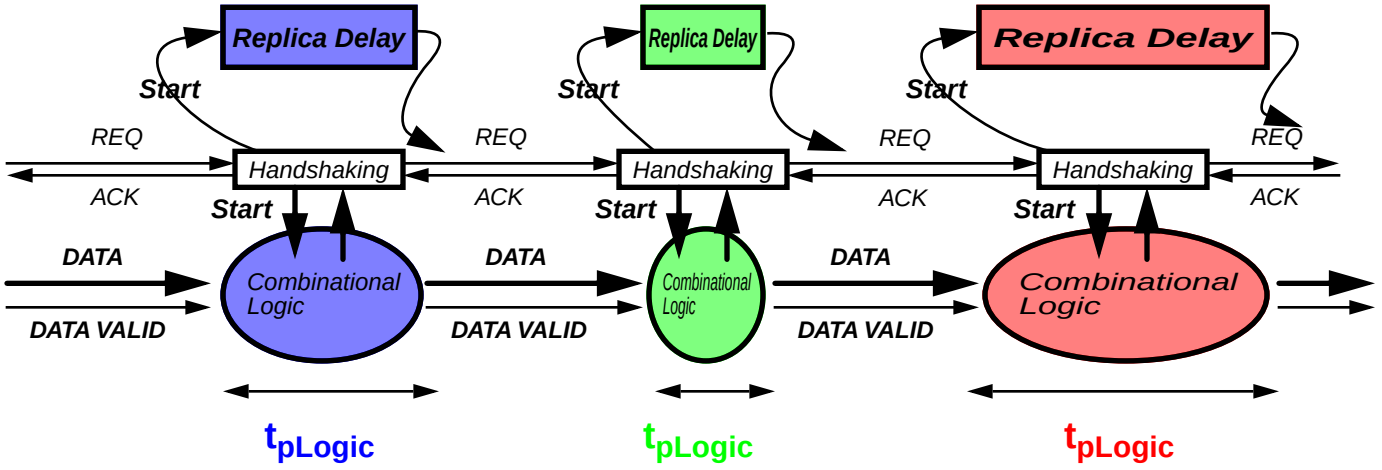


Asynchronous Timing

GOTCHA: Glitches on output (REQ line)

One Solution — CRITICAL PATH REPLICAS

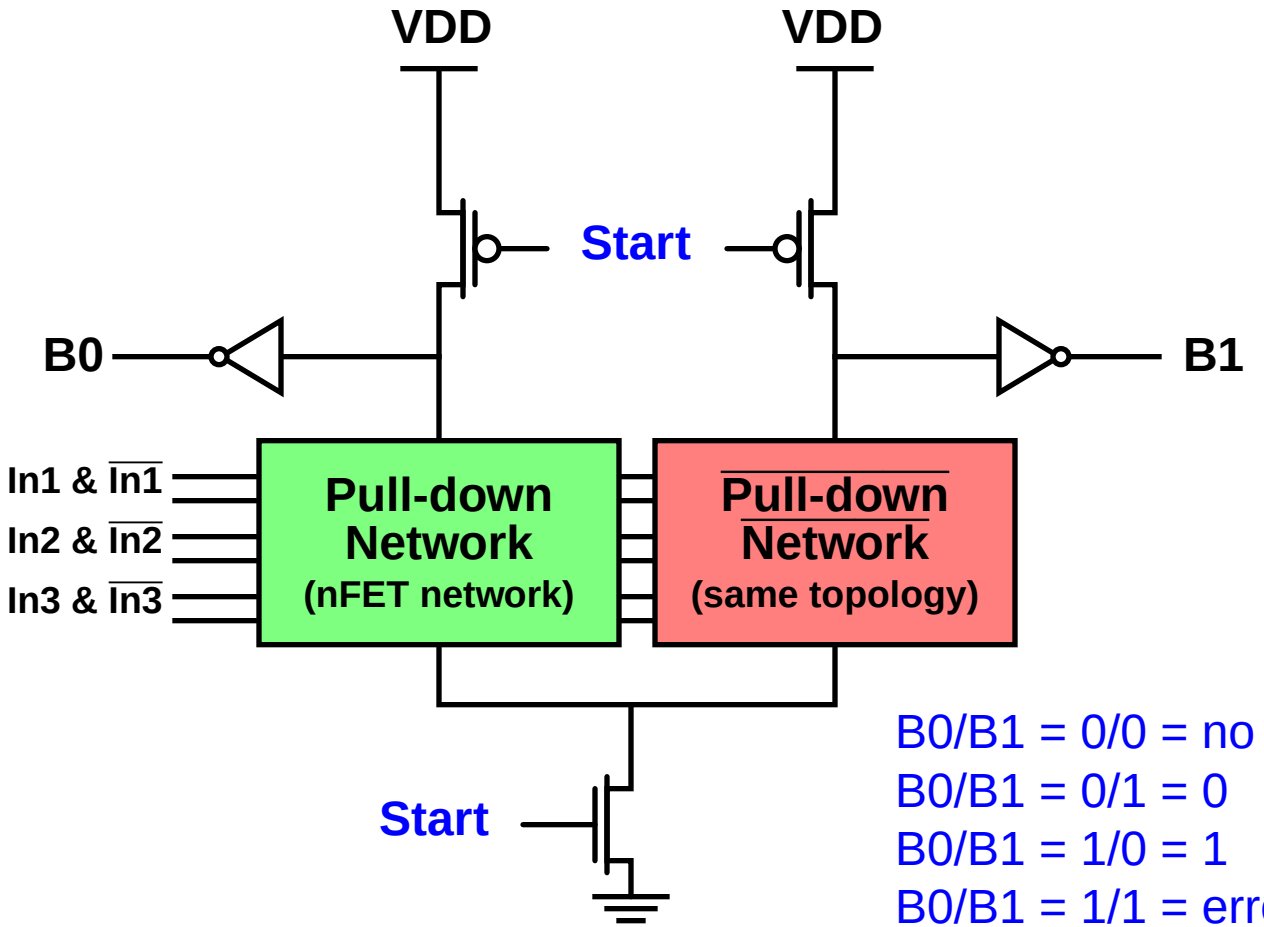
- Match the critical-path delay through the combinational logic block
- When a "start" signal indicates to begin processing, send signal through REPLICA DELAY block
- Combinational logic block is done processing when signal exits REPLICA DELAY



Asynchronous Timing

GOTCHA: Glitches on output (REQ line)

Another Solution — DUAL-RAIL CODING



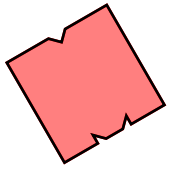
B0/B1 = 0/0 = no data
B0/B1 = 0/1 = 0
B0/B1 = 1/0 = 1
B0/B1 = 1/1 = error

A Tale of Three Pipelines

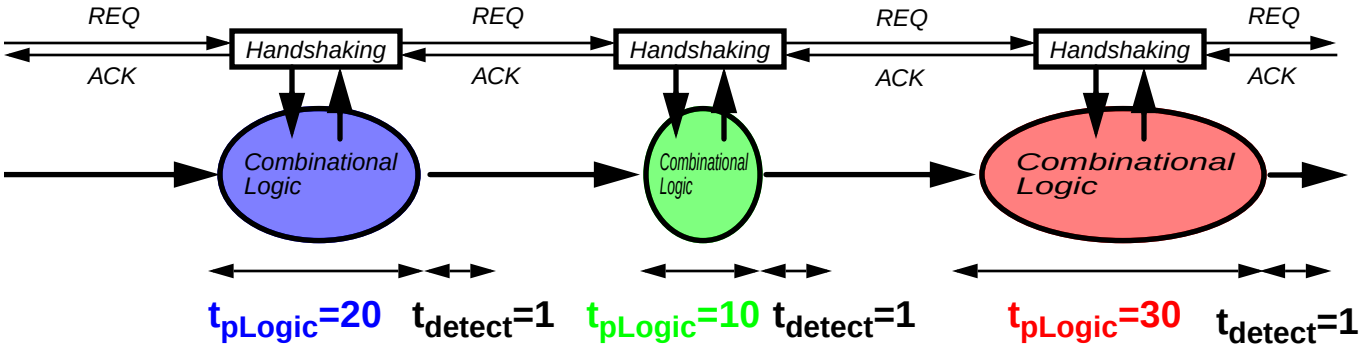
Asynchronous: self-timed

Synchronous: global clock I
(“normal” design: unbalanced pipe)

Synchronous: global clock II
(intentionally skews clock to balance pipe, uses wave pipelining)



Asynchronous Pipeline



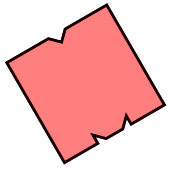
Say best/worst case = 0.5 (e.g. best=10/5/15)

On average, graduate one result every

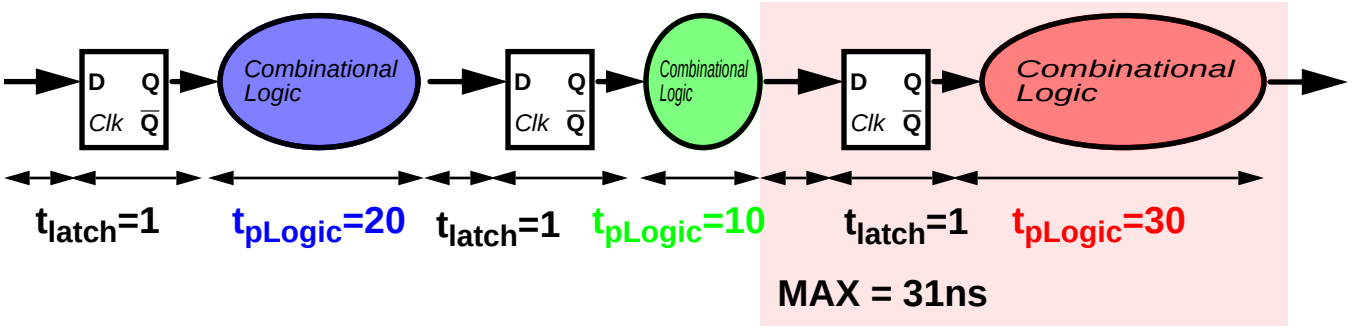
(15+1 + 30+1) ÷ 2

time units (say ns): roughly every 23.5ns, for an effective speed of 43MHz

Delay through pipe for single item = 33/63ns



Synchronous Pipeline I

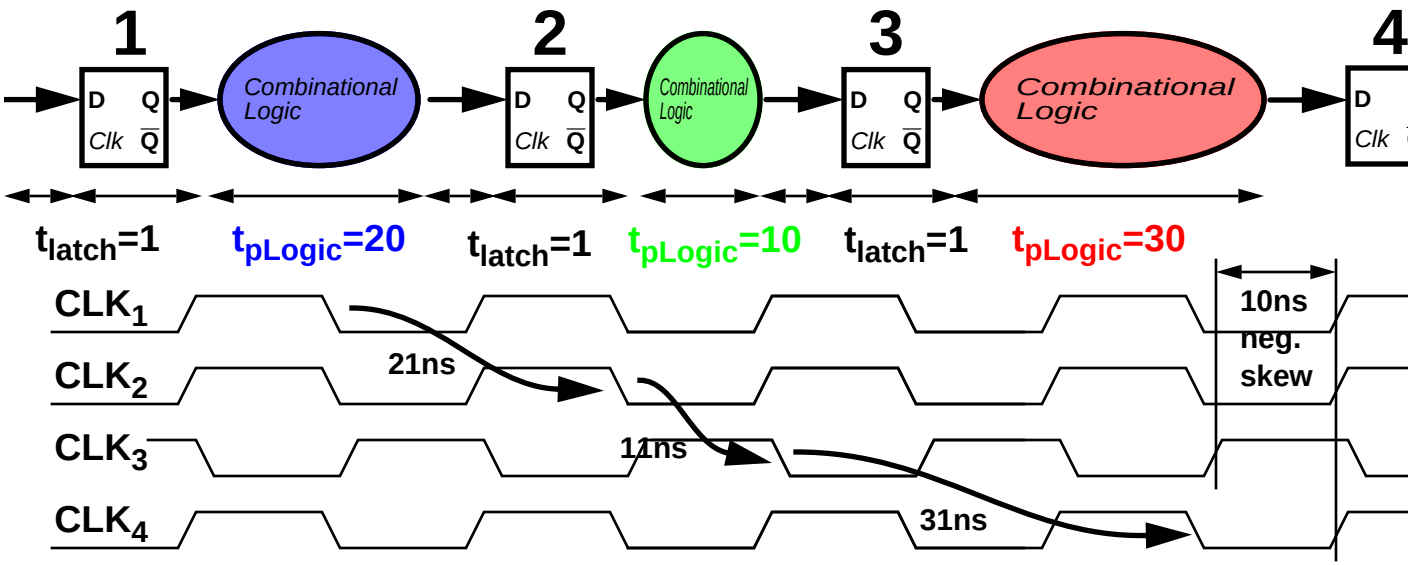


Clock period = 31ns

By design, graduate one result every 31ns, for an effective speed of 32MHz

Delay through pipe for single item = 93ns

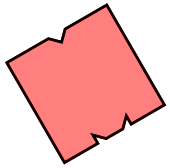
Synchronous Pipeline II



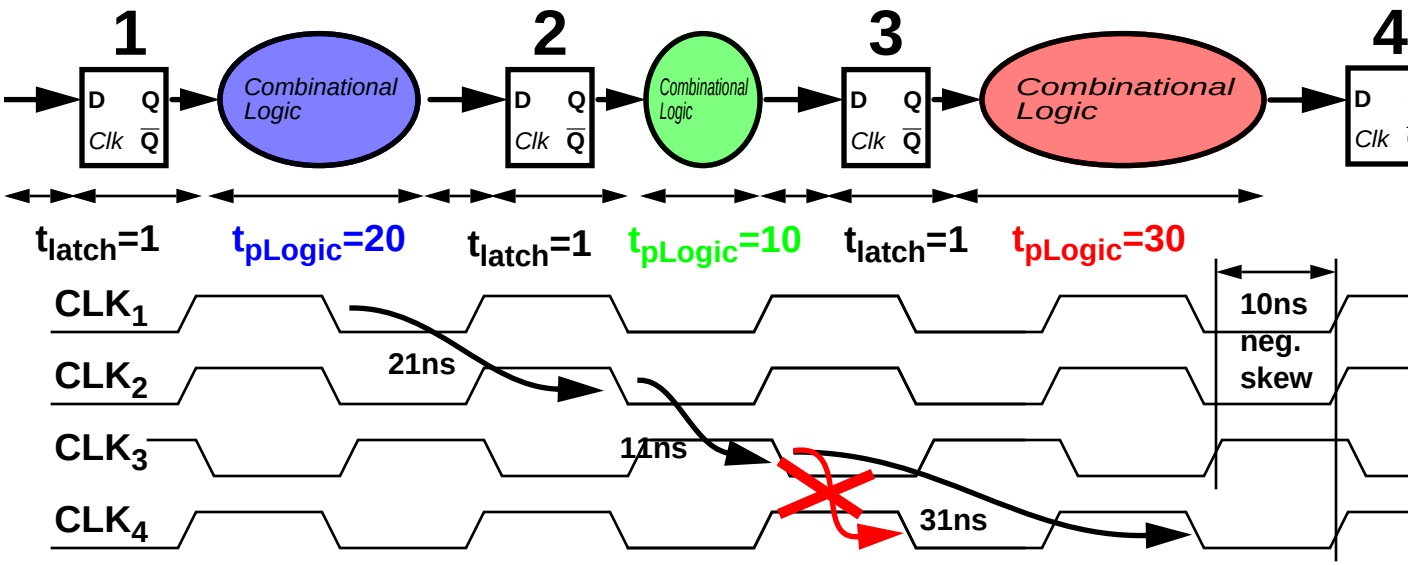
Clock period = 21ns; intentionally skew clock to register #3 to arrive 10ns *early*

By design, graduate one result every 21ns, for an effective speed of 48MHz

Delay through pipe for single item = 63ns



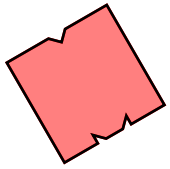
Synchronous Pipeline II



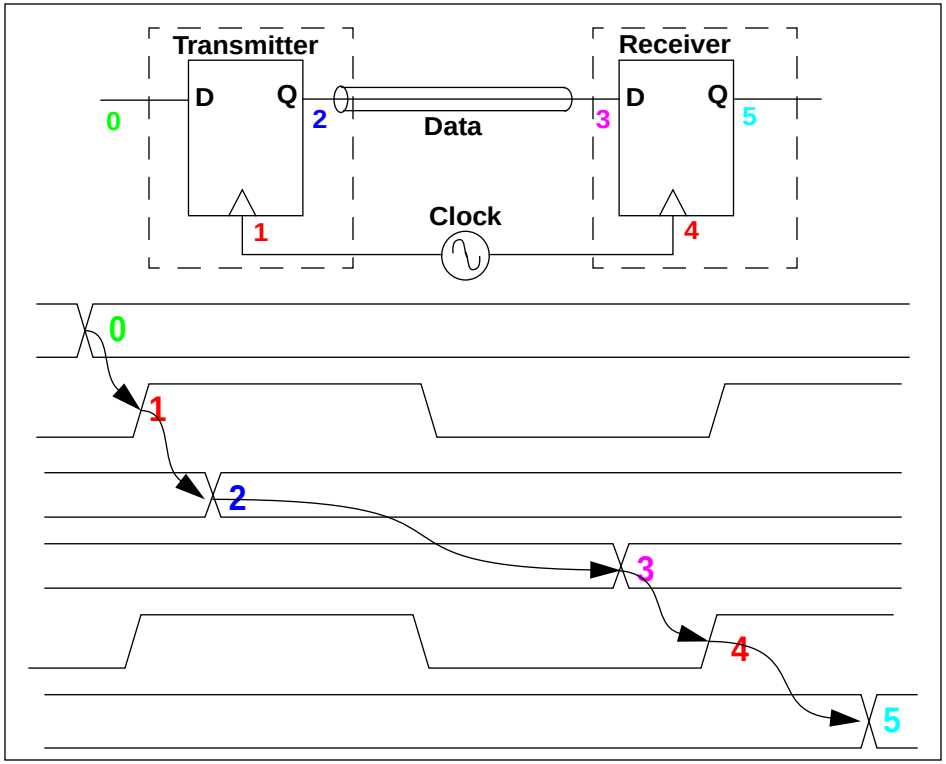
Clock period = 21ns; intentionally skew clock to register #3 to arrive 10ns *early*

By design, graduate one result every 21ns, for an effective speed of 48MHz

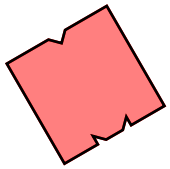
Note: This requires use of *wave-pipelining* design techniques on last combinational logic block—very hard to do (probably much easier to do asynchronous design)



Global Clock I



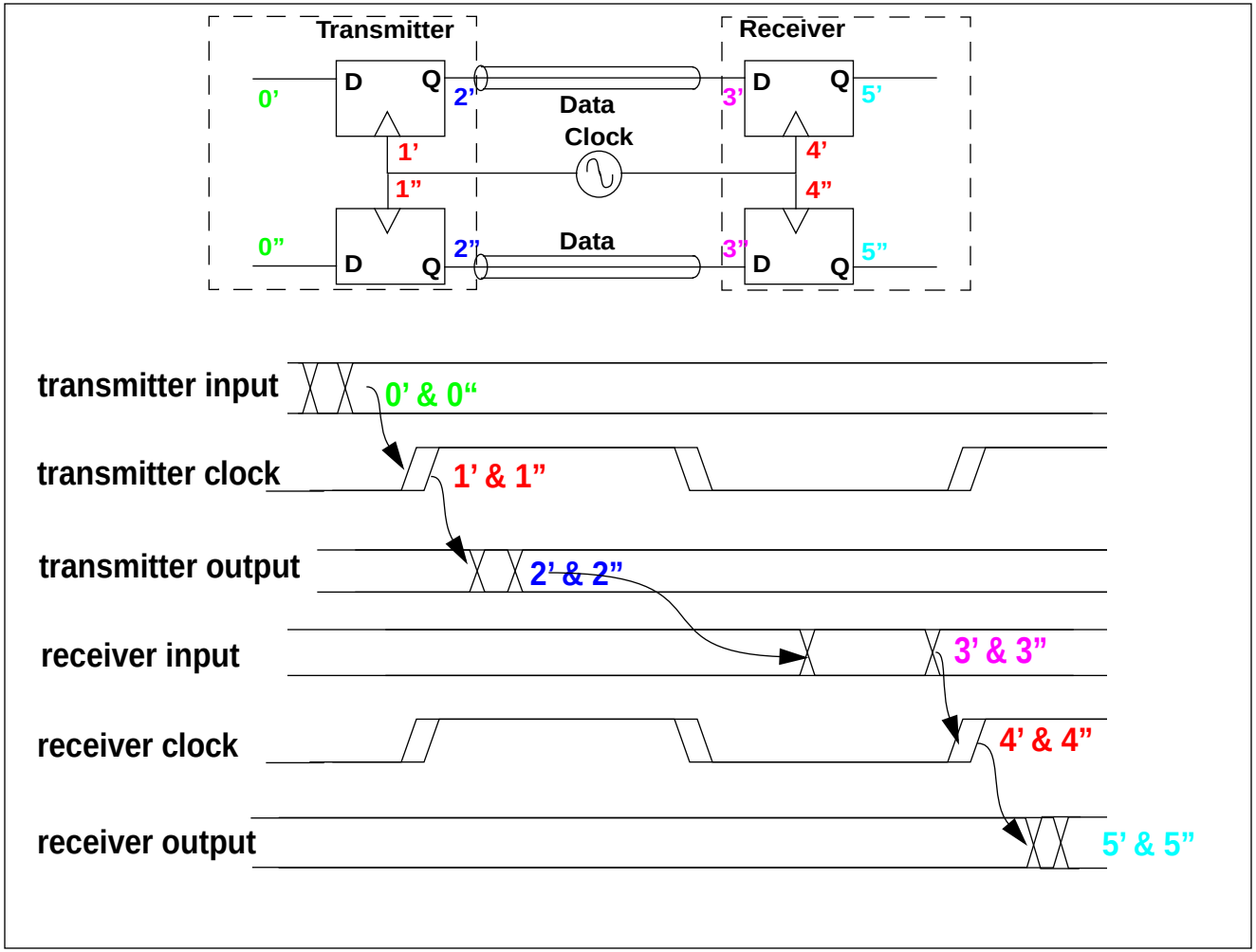
0: Assume data is stable for setup time before clock edge
1: Rising edge of transmitter clock
2: Transmitter begins to drive data (perhaps through logic)
3: Signal reaches input of receiver.
4: Rising edge of receiver clock
5: Receiver latches data and drives internal signal lines



Bruce Jacob

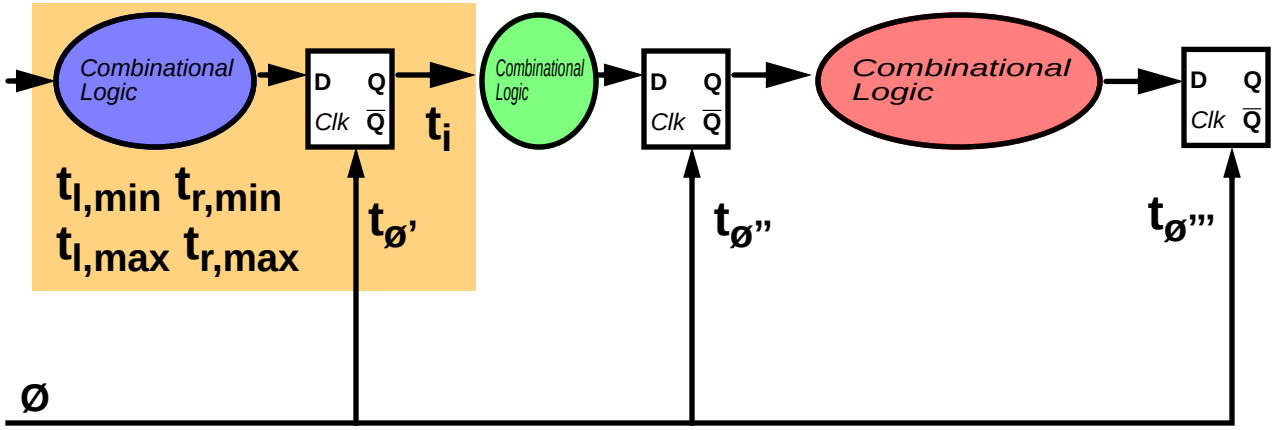
University of
Maryland
ECE Dept.

Global Clock II: Parallel Data



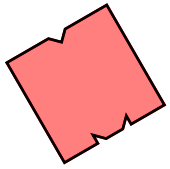
- Skew and jitter eats into timing budget
- Luckily, uncertainty does not accumulate beyond latches

Clock Skew & Pipelines

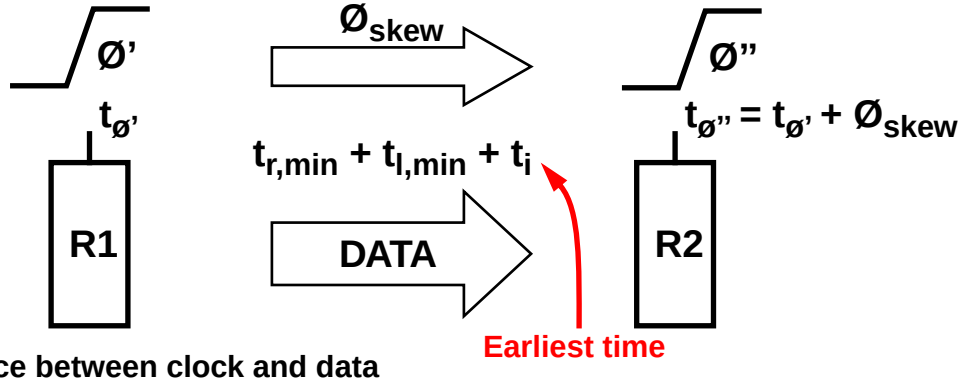


Clock edge timing depends upon position

- **A clock line behaves as a distributed RC line**
- **Each register sees a local clock time depending on their distance from the clock source -> *clock skew***
- **$Ø_{skew} = t_{ø''} - t_{ø'}$ (> 0 or <0)**
- **Clock skew can severely affect the performance**
- **Note: we assumed here t_{setup}=0**



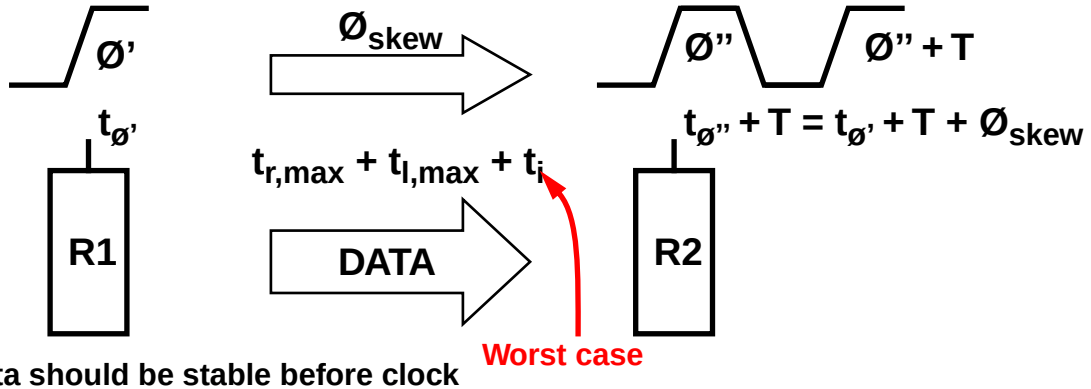
Constraints on Skew



(a) Race between clock and data

If the local clock of R2 is delayed w.r.t. R1, it might happen that the inputs of R2 change before the previous data is latched -> *race*

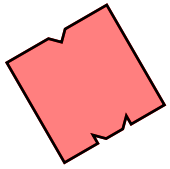
$$Ø_{skew} \leq t_{r,min} + t_i + t_{l,min}$$



(b) Data should be stable before clock

The correct input data is stable at R2 after the worst-case propagation delay. The clock period must be large enough for the computations to settle.

$$T \geq t_{r,max} + t_i + t_{l,max} - Ø_{skew}$$

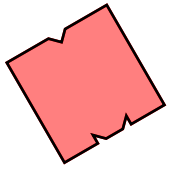


Clock Constraints in Edge-Triggered Logic

$$(1)\ Ø_{skew} \leq t_{r,min} + t_i + t_{l,min}$$

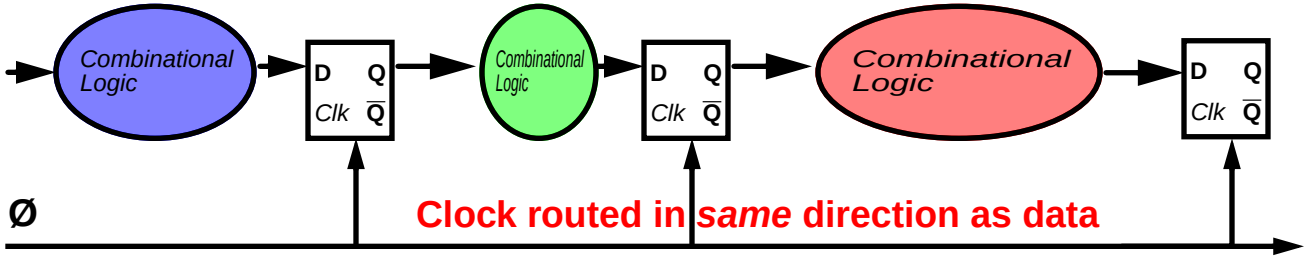
$$(2)\ T \geq t_{r,max} + t_i + t_{l,max} - Ø_{skew}$$

- **Maximum Clock Skew Determined by Minimum Delay between Latches (condition 1)**
- **Minimum Clock Period Determined by Maximum Delay between Latches (condition 2)**



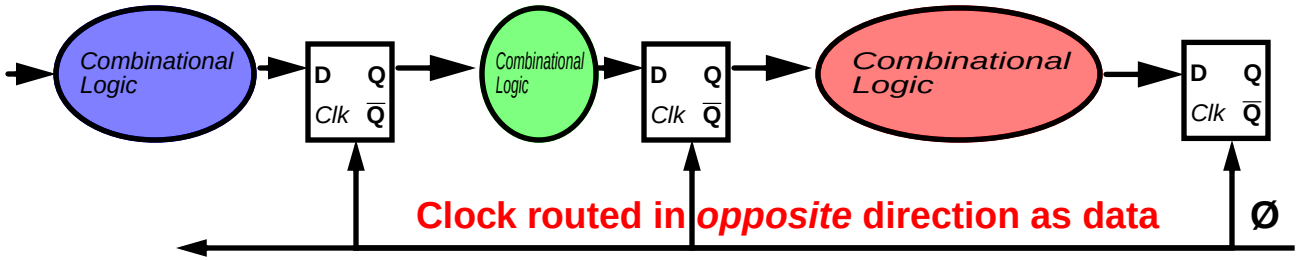
Positive and negative Skew

POSITIVE SKEW:

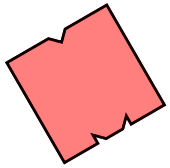


The skew has to satisfy (1)
If it violates (1), then the circuit malfunction independently of the clock period
Clock period decreases!!!

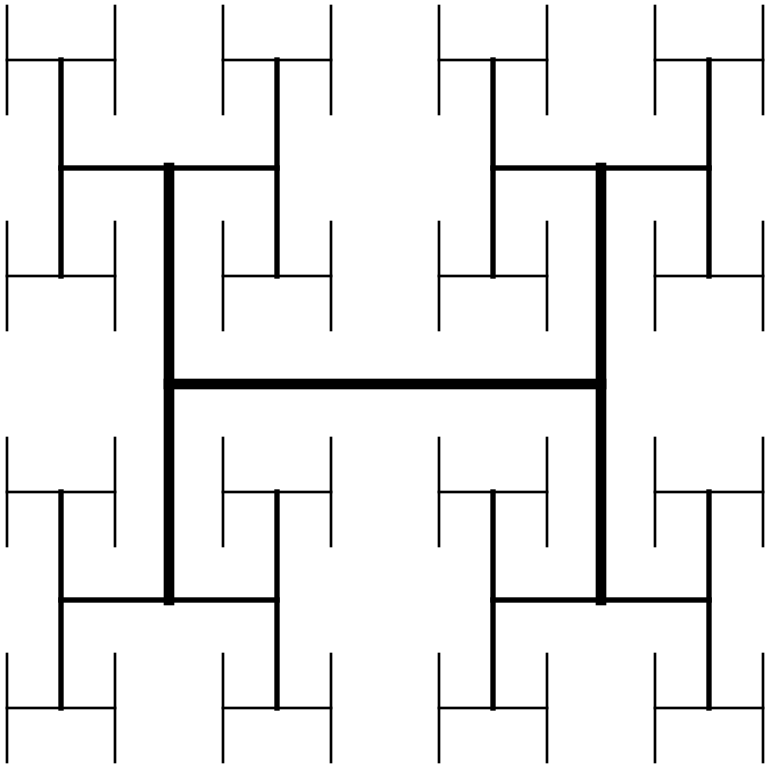
NEGATIVE SKEW:



(1) is satisfied implicitly.
The circuit operates correctly independently of the skew
Clock period increases by | $Ø_{skew}$ |

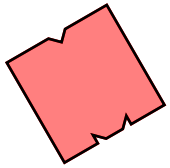


Clock Tree I



Clock distribution is a *major* design problem!

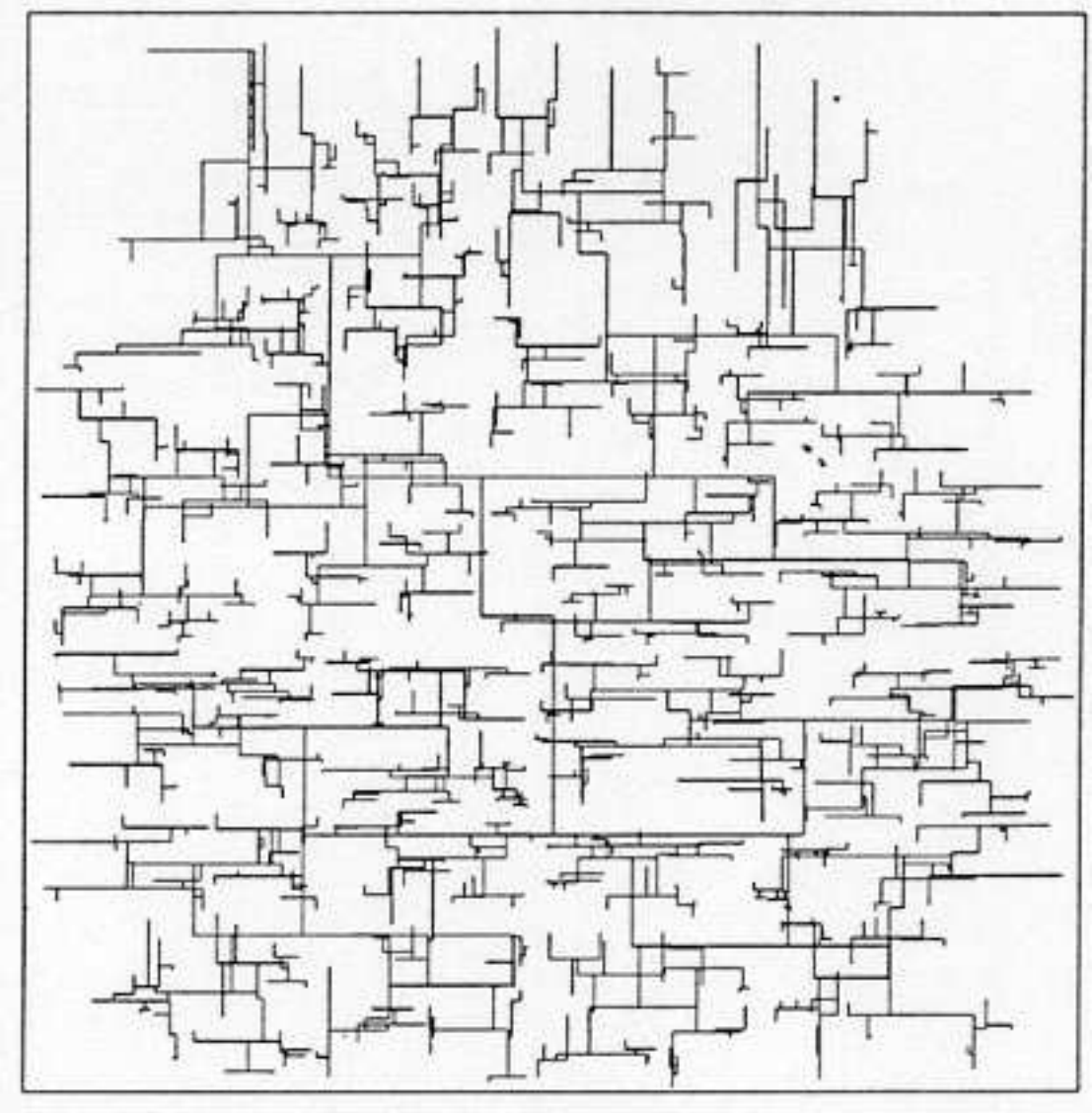
- **Every branch sees same wire length and capacitance**
- **The clock skew is theoretically zero**
- **The sub-blocks should be small enough s.t. the skew within the block is tolerable**
- **Essential to consider clock distribution *early* in the design process**



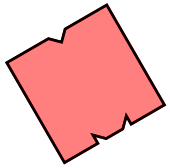
Bruce Jacob

University of
Maryland
ECE Dept.

Clock Tree I



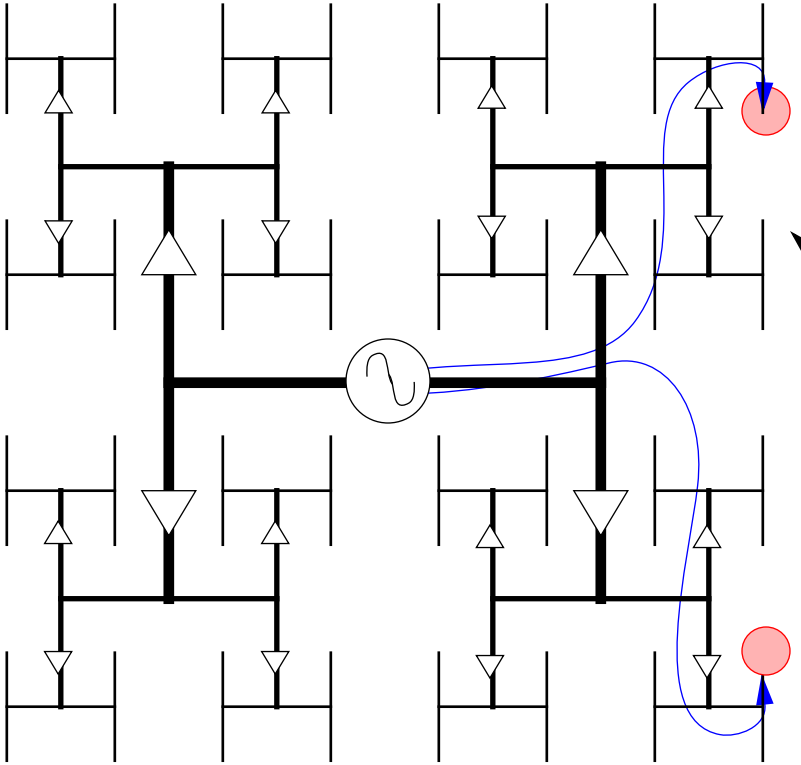
An on-chip clock tree



Bruce Jacob

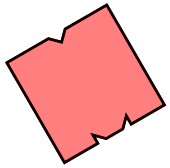
University of
Maryland
ECE Dept.

Clock Tree II (I with buffers)

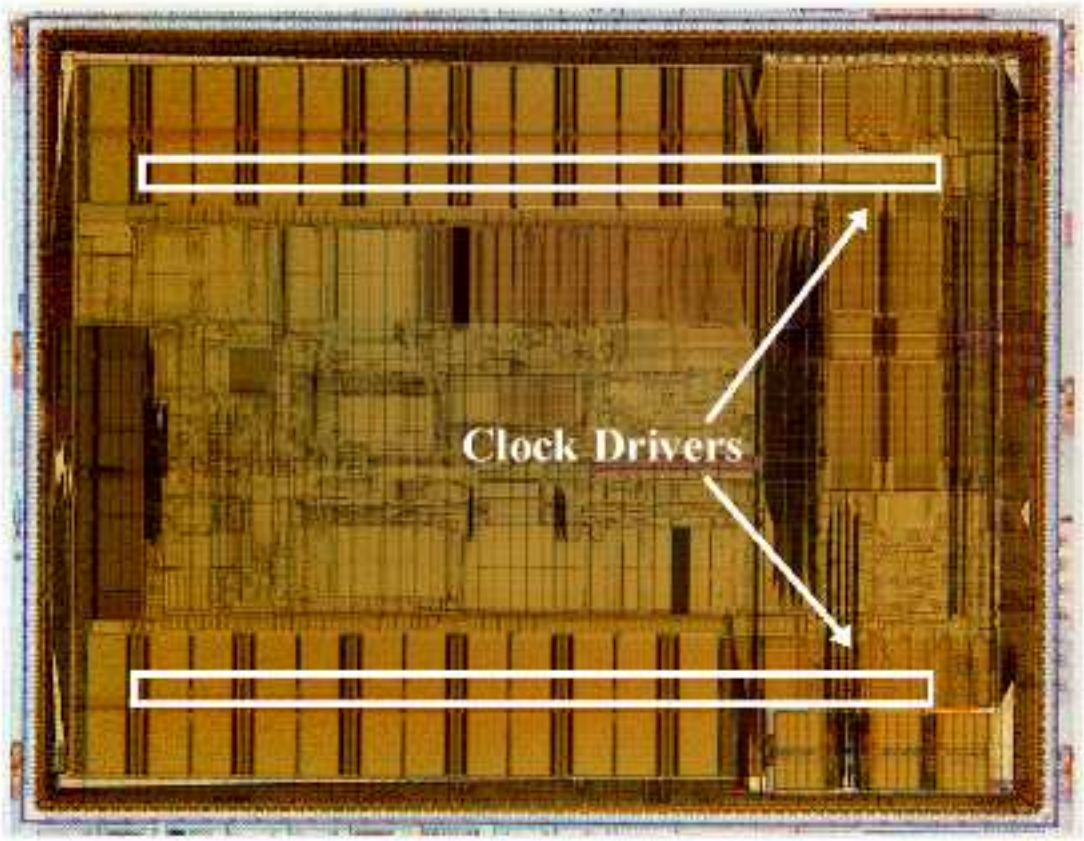


buffer chips could have +/- xx% tolerance

- Large synchronous systems require all components (chips or registers) to be driven by clock signal.
- Clock signal paths and buffers could introduce both skew and jitter at each stage
- Jitter and skew are additive with larger systems. More buffering, more skew and jitter.



DEC Alpha 21164

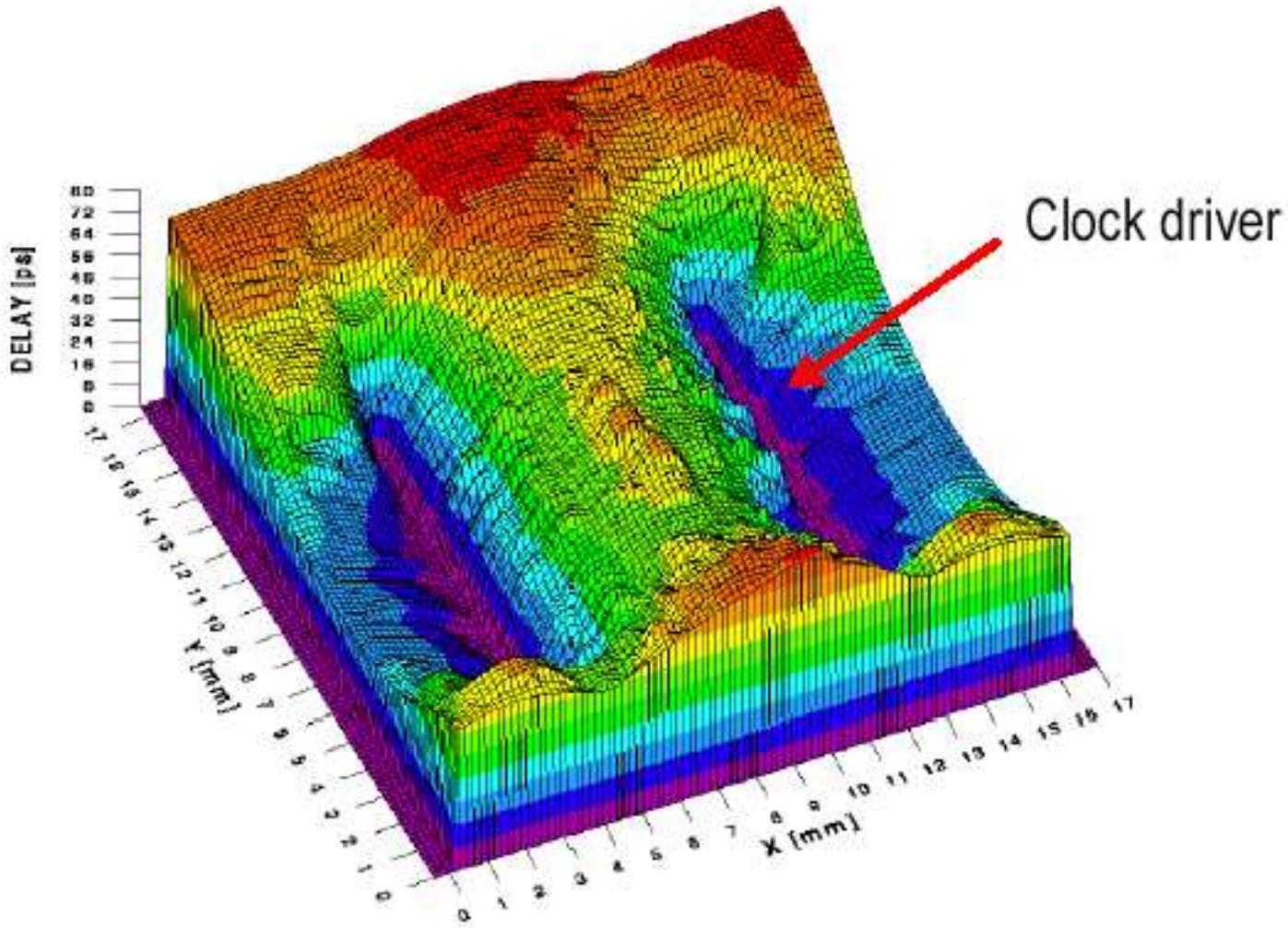


9.3 M Transistors, 4 metal layers, 0.55µm
Clock Freq: 300 MHz
Clock Load: 3.75 nF
Power in Clock = 20W (out of 50W)
Two Level Clock Distribution:
• Single 6-stage driver at center
• Secondary buffers drive left and right side
Max clock skew less than 100psec
• Routing the clock in the opposite direction
• Proper timing

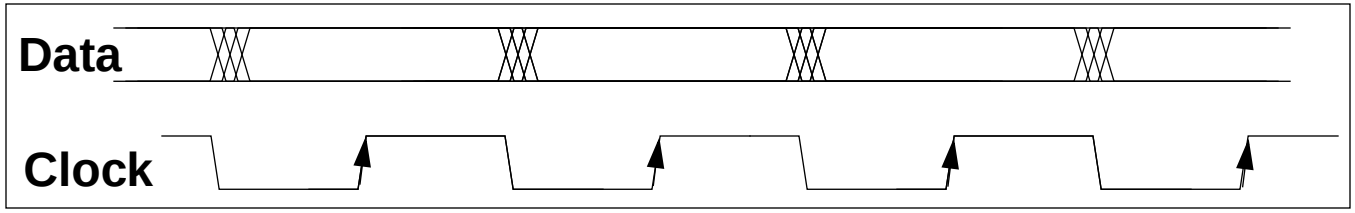
Bruce Jacob

University of
Maryland
ECE Dept.

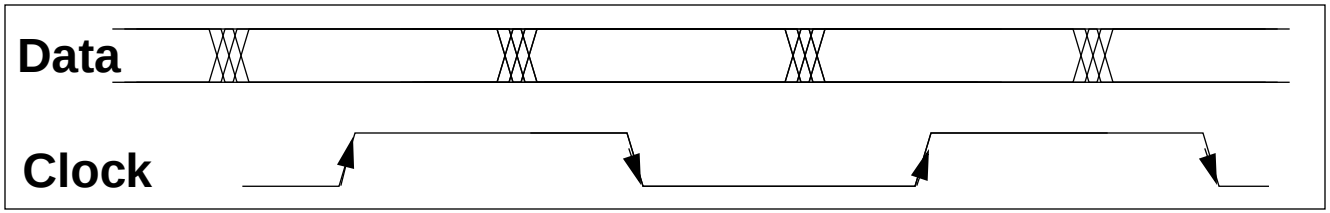
Clock Skew in Alpha



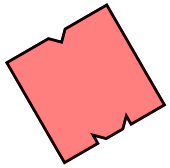
Dual Edge Clocking



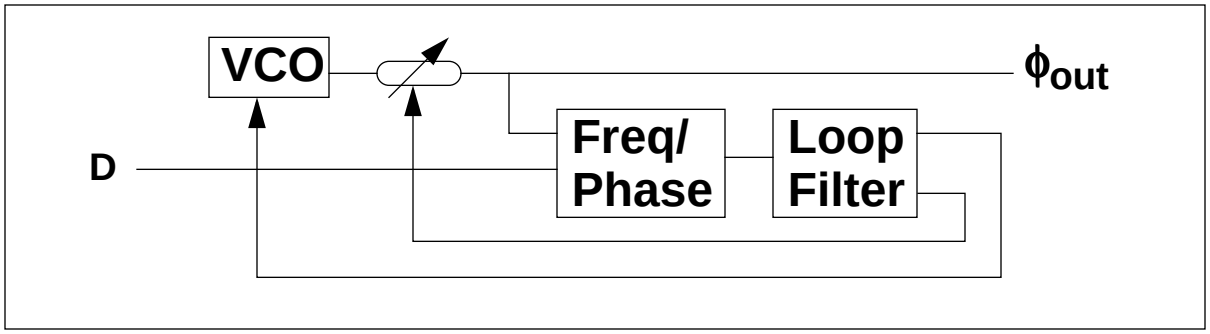
- Only one edge of clock latches data
- Duty cycle of clock signal is not relevent
- Clock signal operating at 2X switching rate of data
- Always a clock edge where you need one



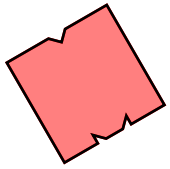
- Both edges of clock used to latch in data
- Duty cycle & rise/fall times of clock must be even
- Clock signal must be phase shifted by 90 degrees relative to phase of data signal
- How do you get 90 degrees ??



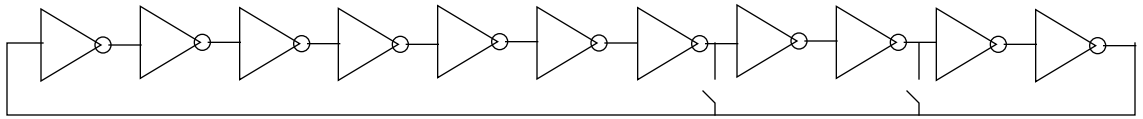
Phase Locked Loop



- **Given a data signal, recover the frequency and phase of the data signal, generate local reference clock ϕ_{out}**
- **Local reference clock may be frequency multiple of input clock**
- **PLL depends on data input to provide "enough" signal transitions to lock onto, else PLL could lose coherency.**
- **Modern processors utilize PLL's for frequency multiplication**

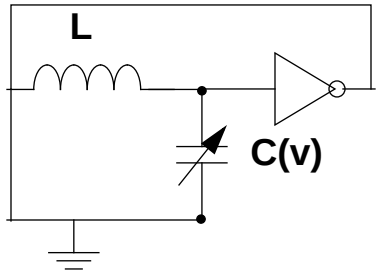


Voltage Controlled Oscillator



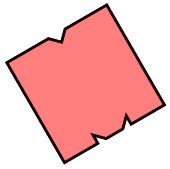
Ring Oscillator

- VCO may be designed from ring oscillator where voltage controls the number of (odd) stages of inverters in the feedback ring

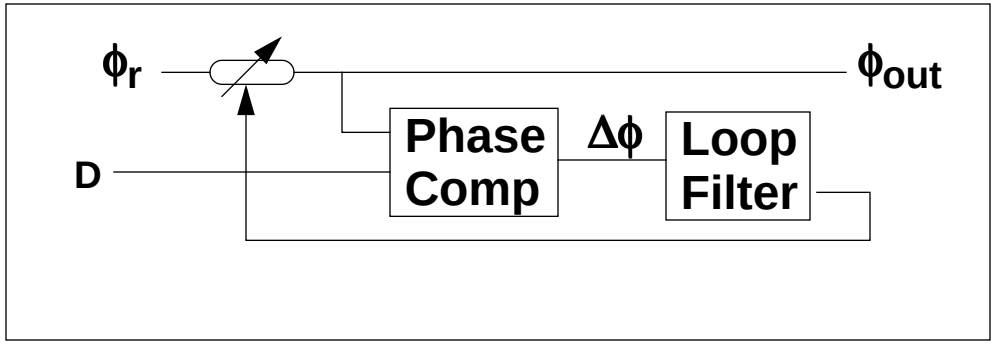


LC Oscillator (conceptual illustration)

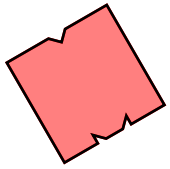
- VCO may be designed from resonant oscillator where voltage controls capacitance in LC circuit.



Delay Locked Loop



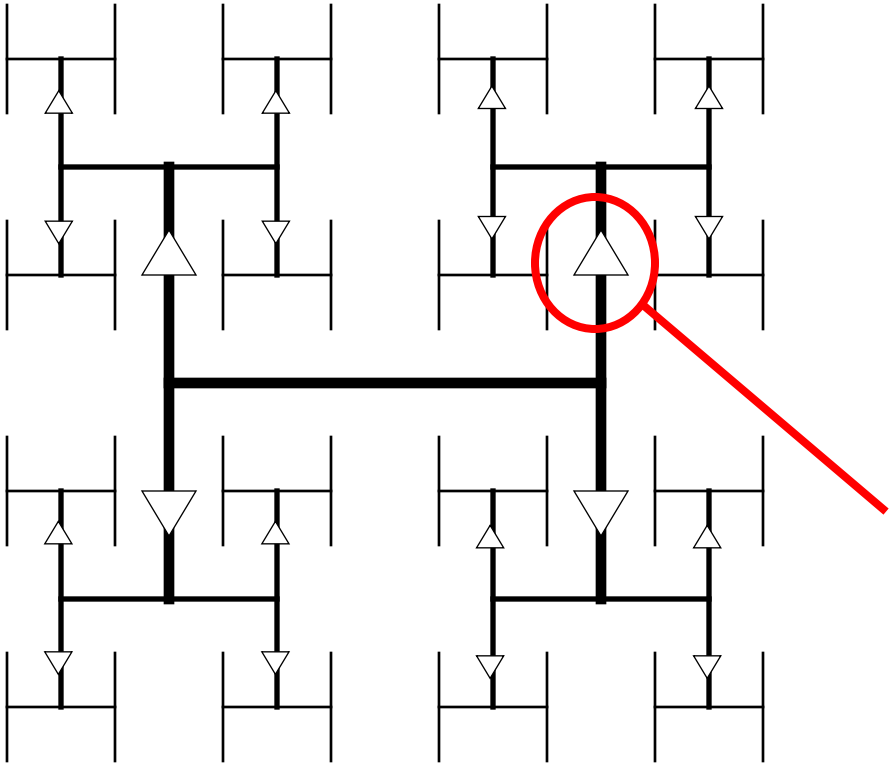
- Given a data signal and reference clock, compare and adjust phase of local clock signal by $\Delta\phi$
- Unlike PLL, requires reference clock
- Hence, no need to "recover" clock signal with VCO
- Modern DRAM with dual edged clocking utilizes DLL's for phase compensation. (gets you 90 degrees)

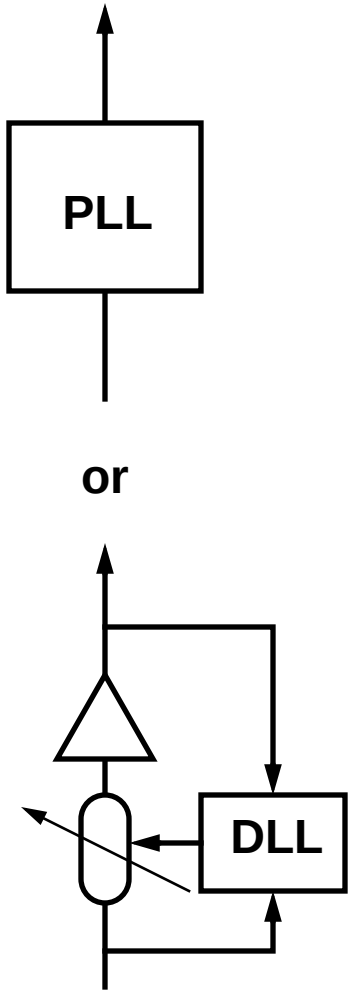


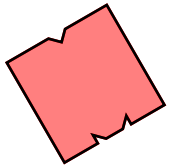
Bruce Jacob

University of
Maryland
ECE Dept.

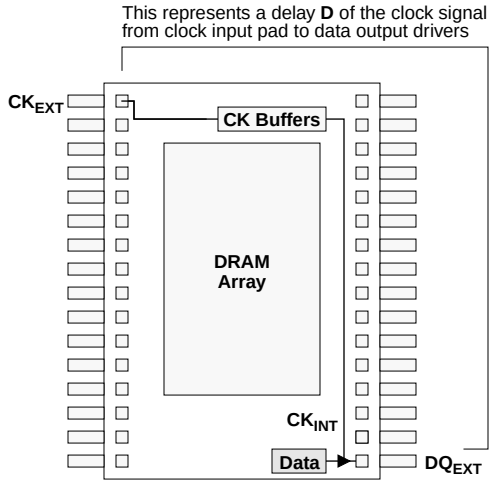
Zero-Skew Clock Distribution

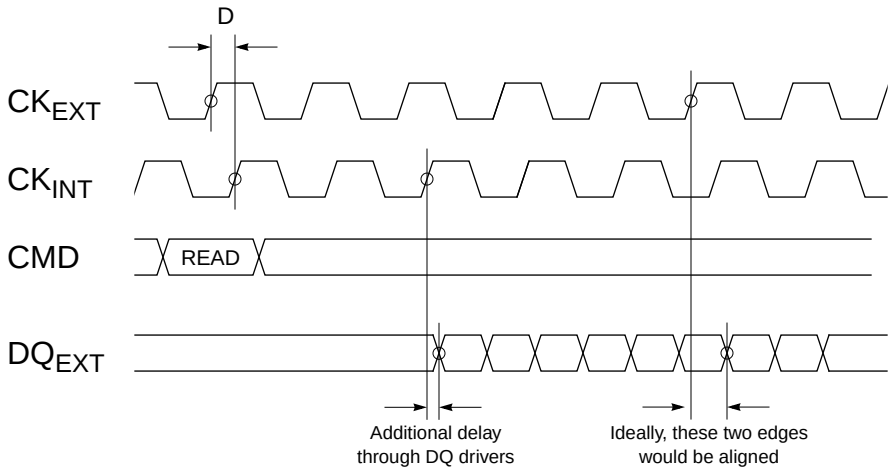


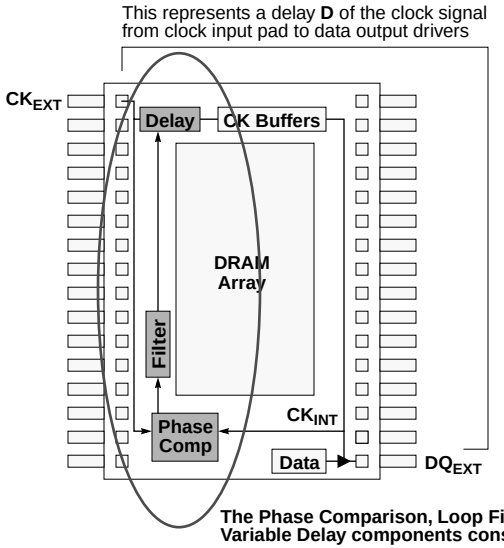




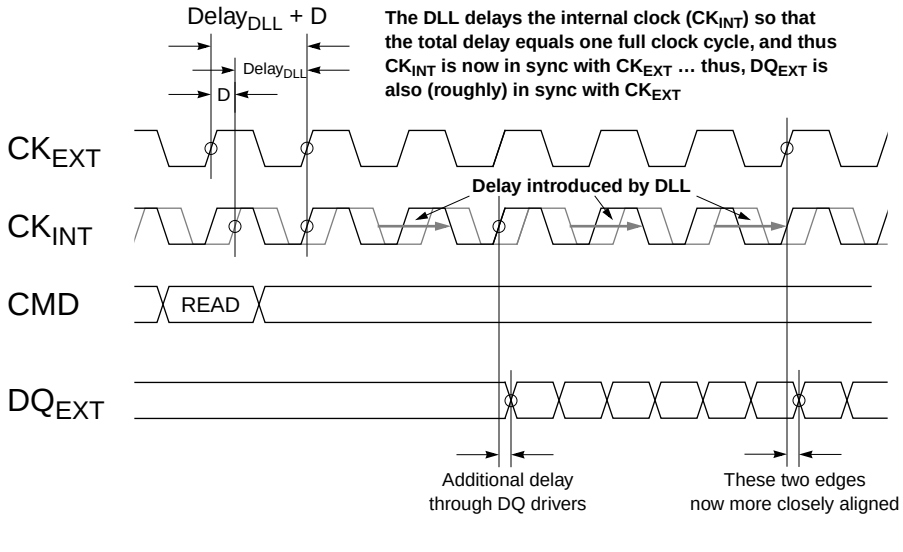
DLL in DDR SDRAM







The Phase Comparison, Loop Filter, and Variable Delay components constitute a DLL



The DLL delays the internal clock (CK_{INT}) so that the total delay equals one full clock cycle, and thus CK_{INT} is now in sync with CK_{EXT} … thus, DQ_{EXT} is also (roughly) in sync with CK_{EXT}