

# 24. Skew-Tolerant Design

- Previous Unit:
  - Power
    - Dynamic
    - Static
  - Design for low power
- This Unit:
  - Clock Distribution
  - Clock Skew
  - Skew-Tolerant Static Circuits
  - Skew-Tolerant Domino Circuits

# Clocking

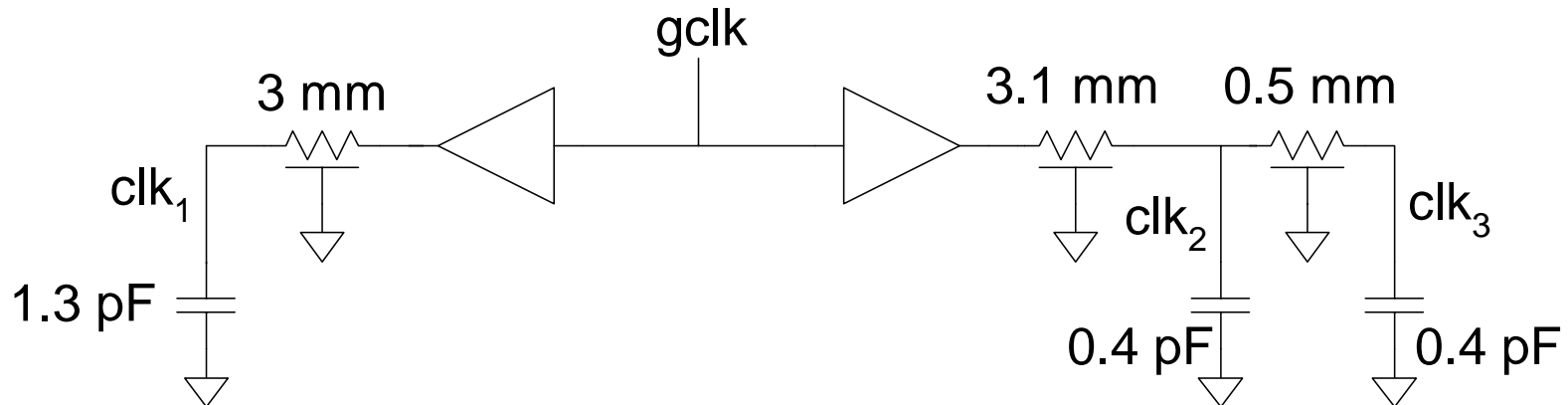
- Synchronous systems use a clock to keep operations in sequence
  - Distinguish *this* from *previous* or *next*
  - Determine speed at which machine operates
- Clock must be distributed to all the sequencing elements
  - Flip-flops and latches
- Also distribute clock to other elements
  - Domino circuits and memories

# Clock Distribution

- On a small chip, the clock distribution network is just a wire
  - And possibly an inverter for clkb
- On practical chips, the RC delay of the wire resistance and gate load is very long
  - Variations in this delay cause clock to get to different elements at different times
  - This is called *clock skew*
- Most chips use repeaters to buffer the clock and equalize the delay
  - Reduces but doesn't eliminate skew

# Example

- Skew comes from differences in gate and wire delay
  - With right buffer sizing,  $\text{clk}_1$  and  $\text{clk}_2$  could ideally arrive at the same time.
  - But power supply noise changes buffer delays
  - $\text{clk}_2$  and  $\text{clk}_3$  will always see RC skew

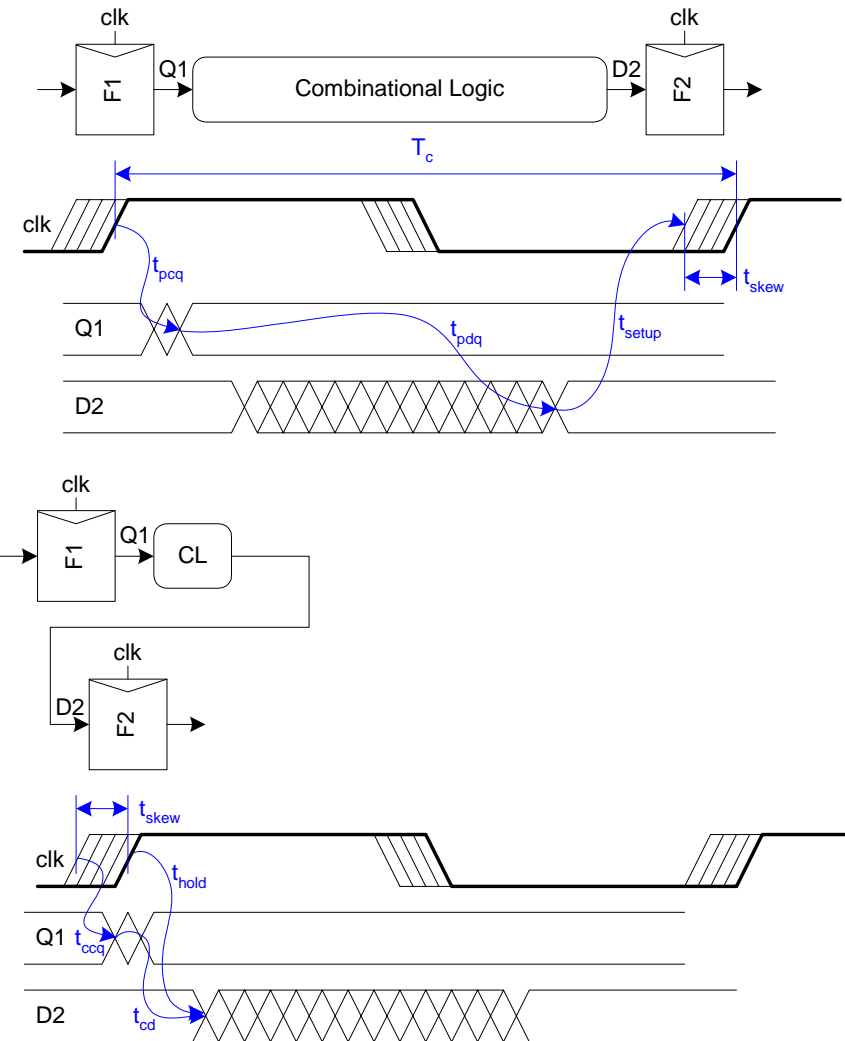


# Review: Skew Impact

- Ideally full cycle is available for work
- Skew adds sequencing overhead
- Increases hold time too

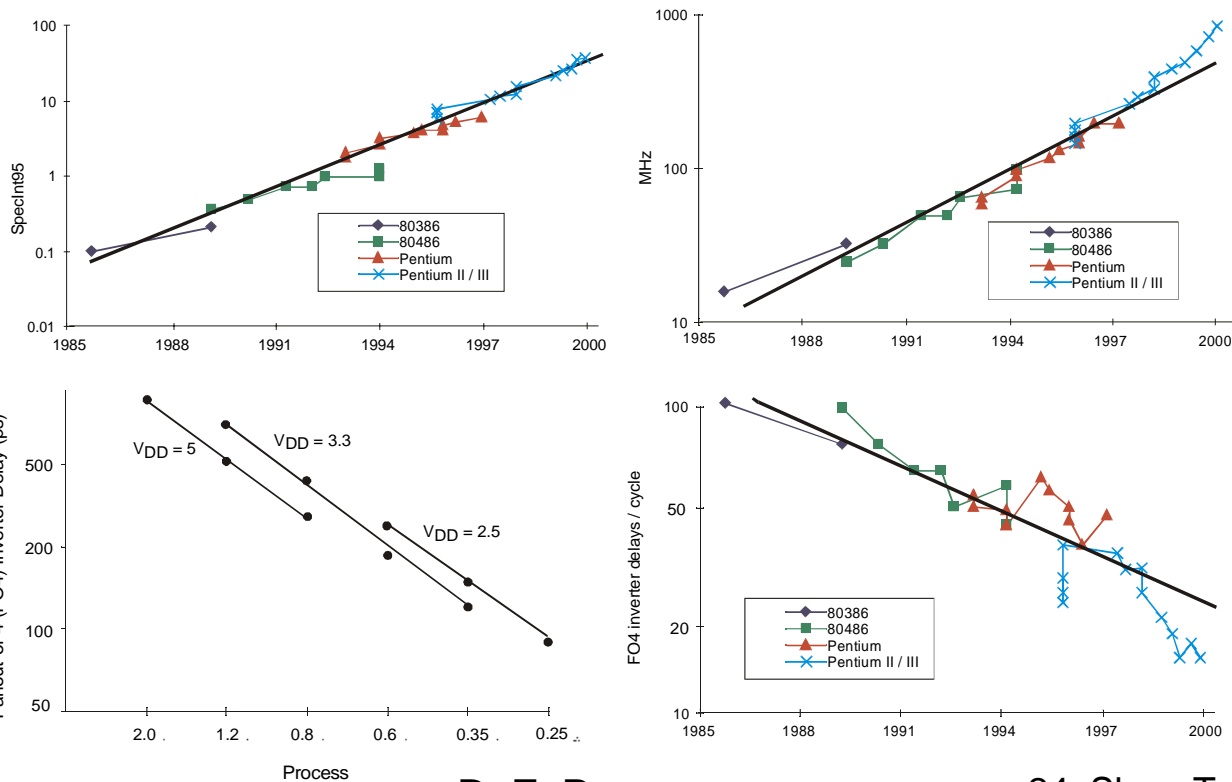
$$t_{pd} \leq T_c - \underbrace{(t_{pcq} + t_{setup} + t_{skew})}_{\text{sequencing overhead}}$$

$$t_{cd} \geq t_{hold} - t_{ccq} + t_{skew}$$



# Cycle Time Trends

- Much of CPU performance comes from higher  $f$ 
  - $f$  improving faster than simple process shrinks
  - Sequencing overhead is bigger part of cycle



D. Z. Pan

24. Skew-Tolerant Design

# Solutions

- Reduce clock skew
  - Careful clock distribution network design
  - Plenty of metal wiring resources
- Analyze clock skew
  - Only budget actual, not worst case skews
  - Local vs. global skew budgets
- Tolerate clock skew
  - Choose circuit structures insensitive to skew

# Clock Distribution Networks

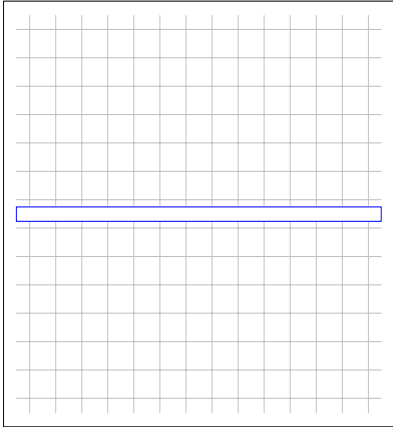
- *Ad hoc*
- Grids
- H-tree
- Hybrid

# Clock Grids

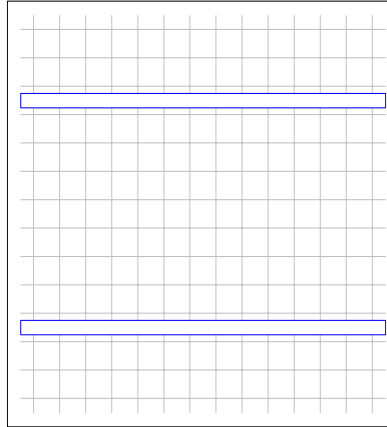
- Use grid on two or more levels to carry clock
- Make wires wide to reduce RC delay
- Ensures low skew between nearby points
- But possibly large skew across die

# Alpha Clock Grids

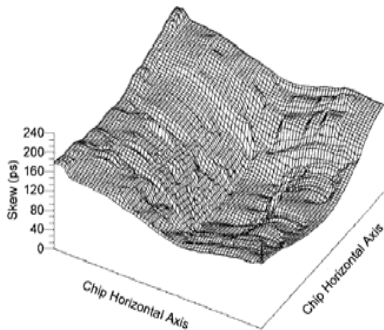
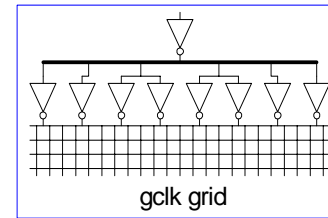
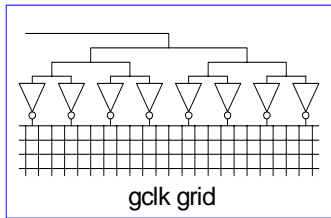
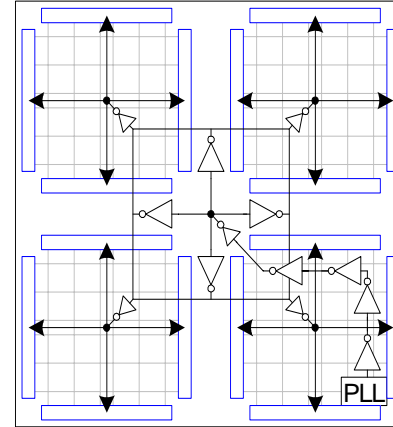
Alpha 21064



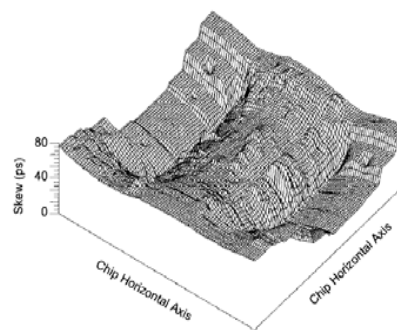
Alpha 21164



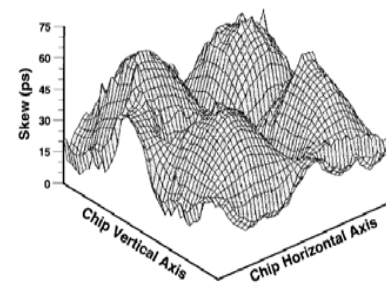
Alpha 21264



Alpha 21064



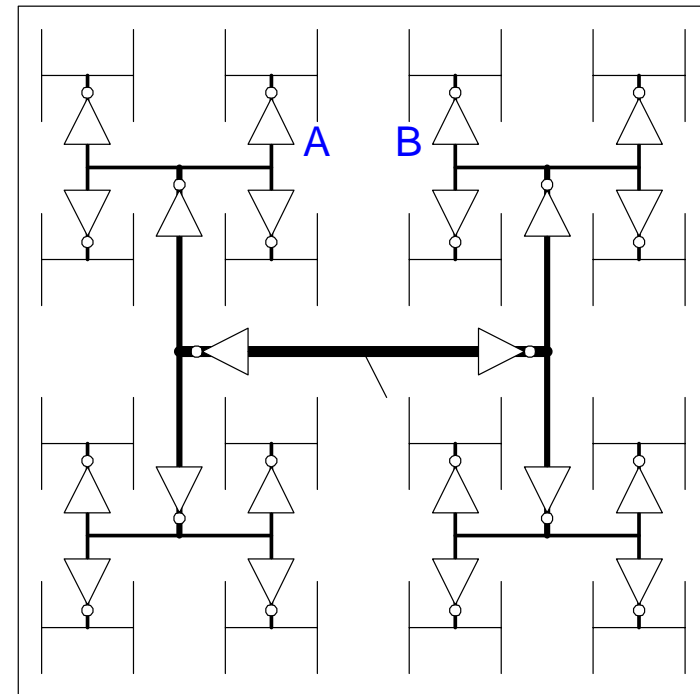
Alpha 21164



Alpha 21264

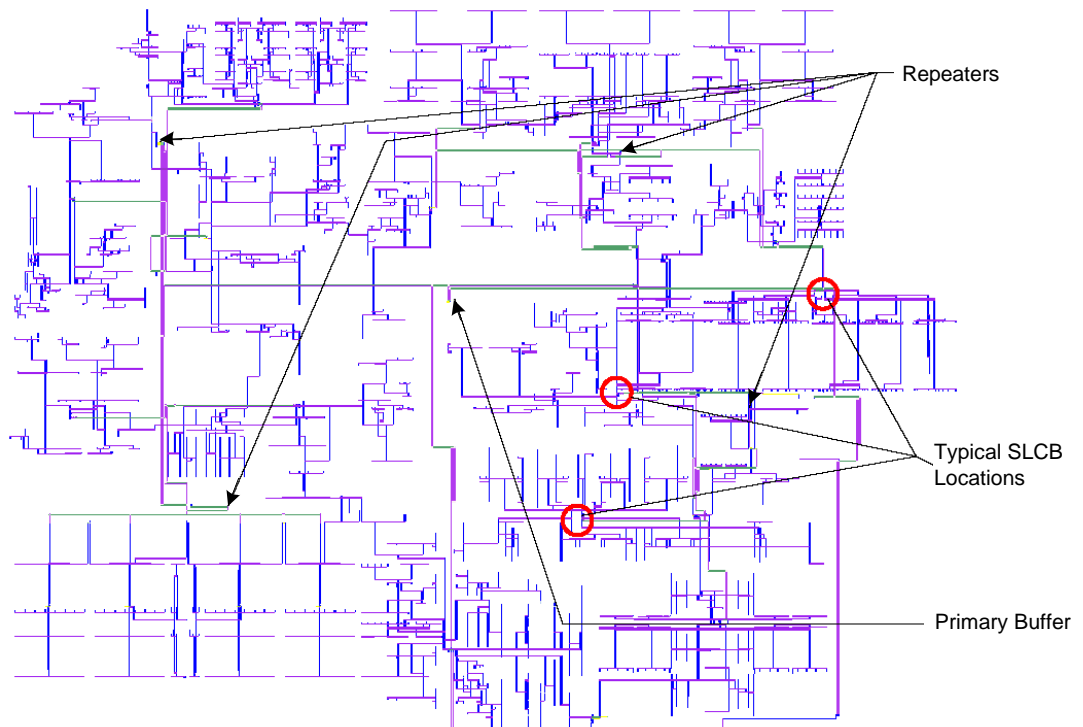
# H-Trees

- Fractal structure
  - Gets clock arbitrarily close to any point
  - Matched delay along all paths
- Delay variations cause skew
- A and B might see big skew



# Itanium 2 H-Tree

- Four levels of buffering:
  - Primary driver
  - Repeater
  - Second-level clock buffer
  - Gater
- Route around obstructions



# Hybrid Networks

- Use H-tree to distribute clock to many points
- Tie these points together with a grid
- Ex: IBM Power4, PowerPC
  - H-tree drives 16-64 sector buffers
  - Buffers drive total of 1024 points
  - All points shorted together with grid

# Skew Tolerance

- Flip-flops are sensitive to skew because of *hard edges*
  - Data launches at latest rising edge of clock
  - Must setup before earliest next rising edge of clock
  - Overhead would shrink if we can soften edge
- Latches tolerate moderate amounts of skew
  - Data can arrive anytime latch is transparent

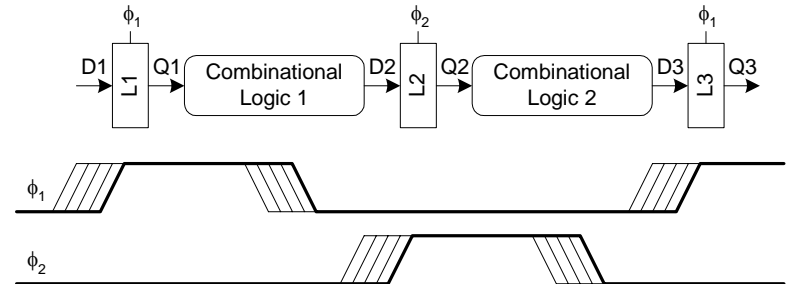
# Skew: Latches

## 2-Phase Latches

$$t_{pd} \leq T_c - \underbrace{(2t_{pdq})}_{\text{sequencing overhead}}$$

$$t_{cd1}, t_{cd2} \geq t_{\text{hold}} - t_{ccq} - t_{\text{nonoverlap}} + t_{\text{skew}}$$

$$t_{\text{borrow}} \leq \frac{T_c}{2} - (t_{\text{setup}} + t_{\text{nonoverlap}} + t_{\text{skew}})$$



## Pulsed Latches

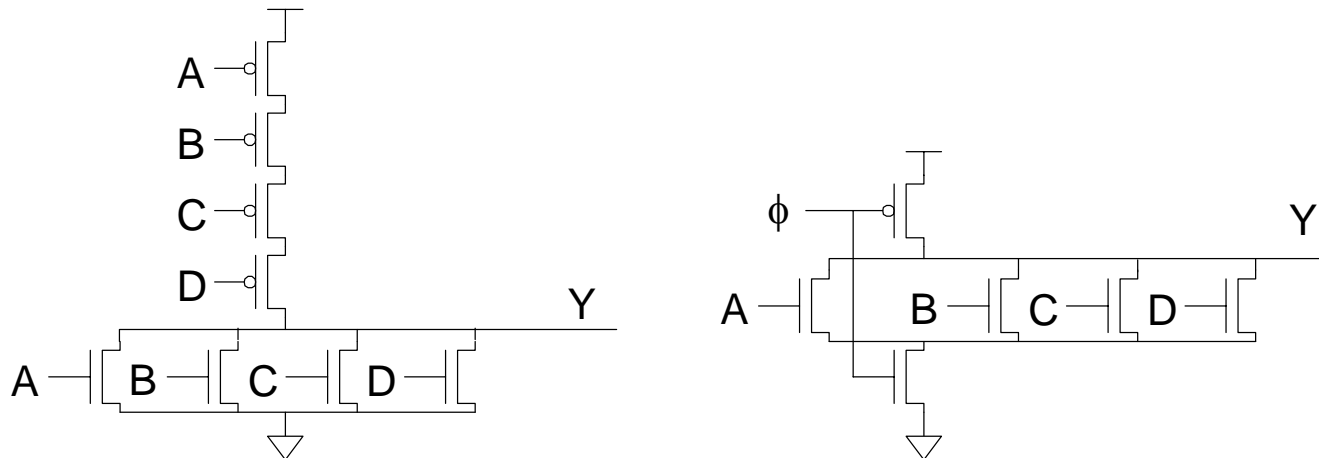
$$t_{pd} \leq T_c - \underbrace{\max(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw} + t_{\text{skew}})}_{\text{sequencing overhead}}$$

$$t_{cd} \geq t_{\text{hold}} + t_{pw} - t_{ccq} + t_{\text{skew}}$$

$$t_{\text{borrow}} \leq t_{pw} - (t_{\text{setup}} + t_{\text{skew}})$$

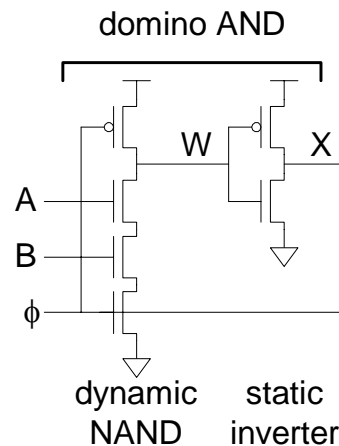
# Dynamic Circuit Review

- Static circuits are slow because fat pMOS load input
- Dynamic gates use precharge to remove pMOS transistors from the inputs
  - Precharge:  $\phi = 0$  output forced high
  - Evaluate:  $\phi = 1$  output may pull low



# Domino Circuits

- Dynamic inputs must monotonically rise during evaluation
  - Place inverting stage between each dynamic gate
  - Dynamic / static pair called domino gate
- Domino gates can be safely cascaded



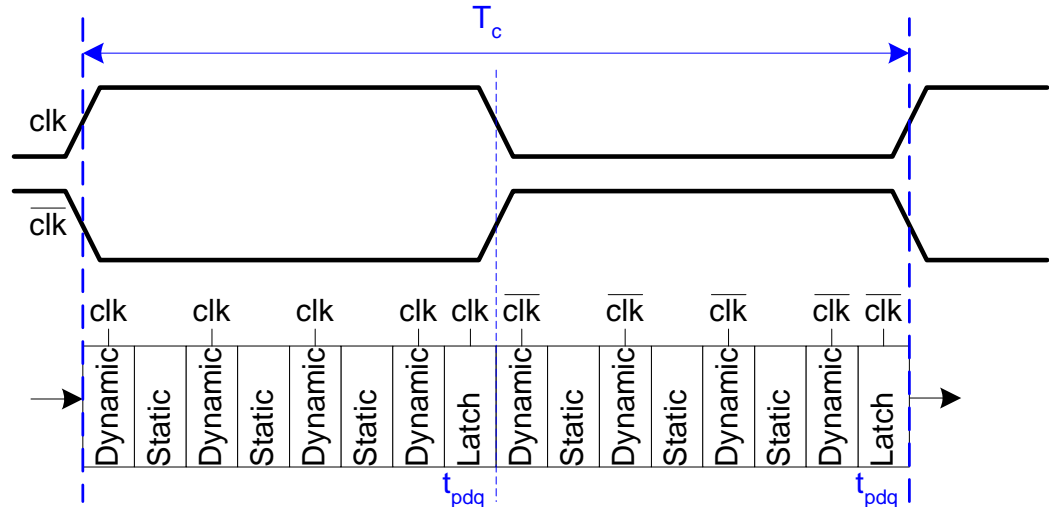
# Domino Timing

- Domino gates are 1.5 – 2x faster than static CMOS
  - Lower logical effort because of reduced  $C_{in}$
- Challenge is to keep precharge off critical path
- Look at clocking schemes for precharge and eval
  - Traditional schemes have severe overhead
  - Skew-tolerant domino hides this overhead

# Traditional Domino Ckts

- Hide precharge time by ping-ponging between half-cycles
  - One evaluates while other precharges
  - Latches hold results during precharge

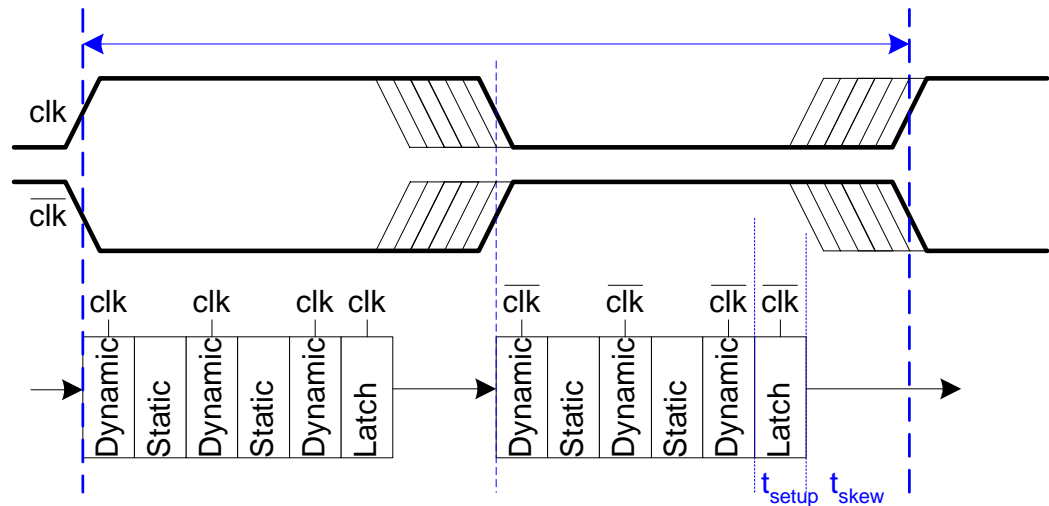
$$t_{pd} = T_c - 2t_{pdq}$$



# Clock Skew

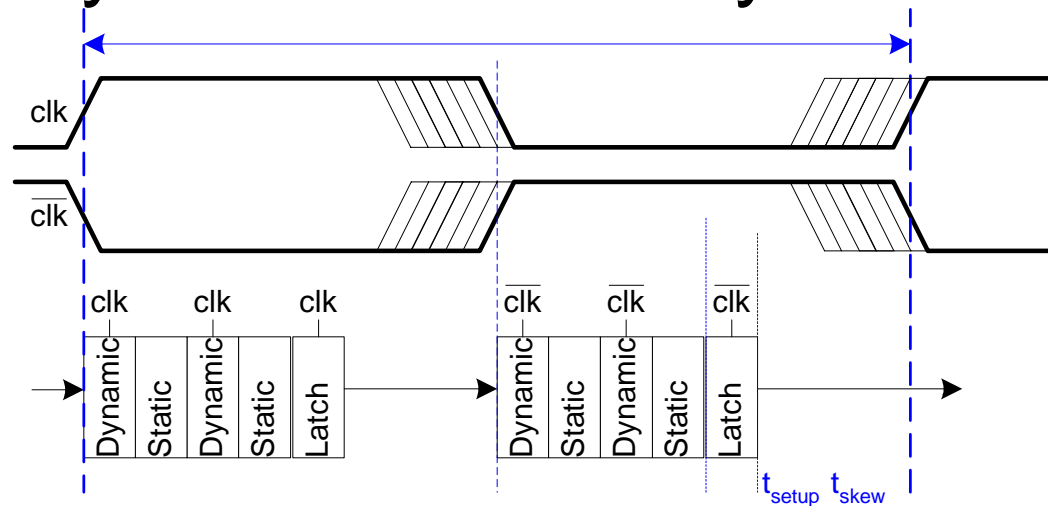
- Skew increases sequencing overhead
  - Traditional domino has hard edges
  - Evaluate at latest rising edge
  - Setup at latch by earliest falling edge

$$t_{pd} = T_c - 2t_{\text{setup}} - 2t_{\text{skew}}$$



# Time Borrowing

- Logic may not exactly fit half-cycle
  - No flexibility to borrow time to balance logic between half cycles
- Traditional domino sequencing overhead is about 25% of cycle time in fast systems!

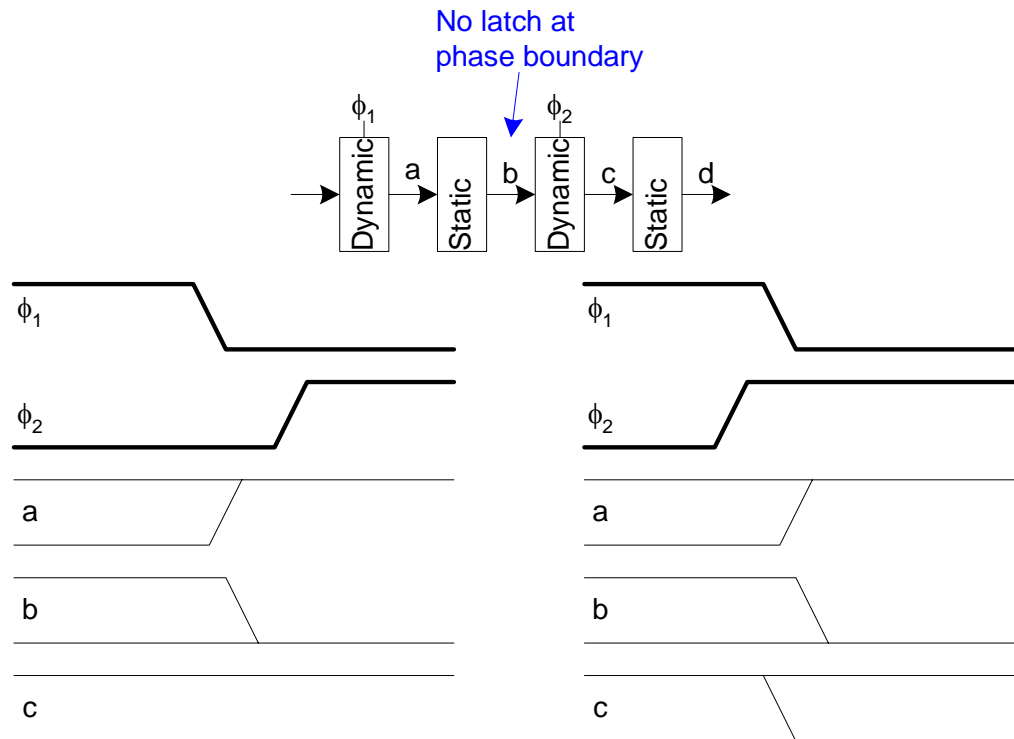


# Relaxing the Timing

- Sequencing overhead caused by hard edges
  - Data departs dynamic gate on late rising edge
  - Must setup at latch on early falling edge
- Latch functions
  - Prevent glitches on inputs of domino gates
  - Holds results during precharge
- Is the latch really necessary?
  - No glitches if inputs come from other domino
  - Can we hold the results in another way?

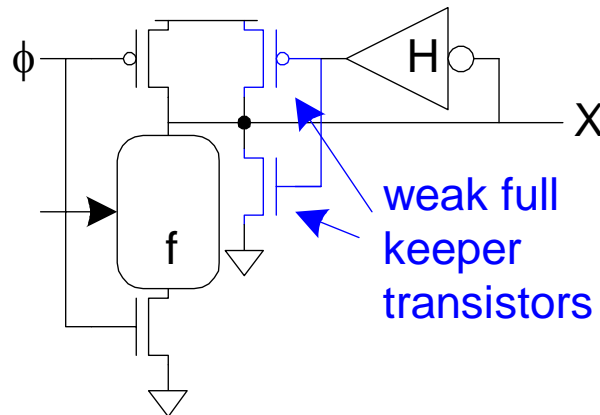
# Skew-Tolerant Domino

- Use overlapping clocks to eliminate latches at phase boundaries.
  - Second phase evaluates using results of first



# Full Keeper

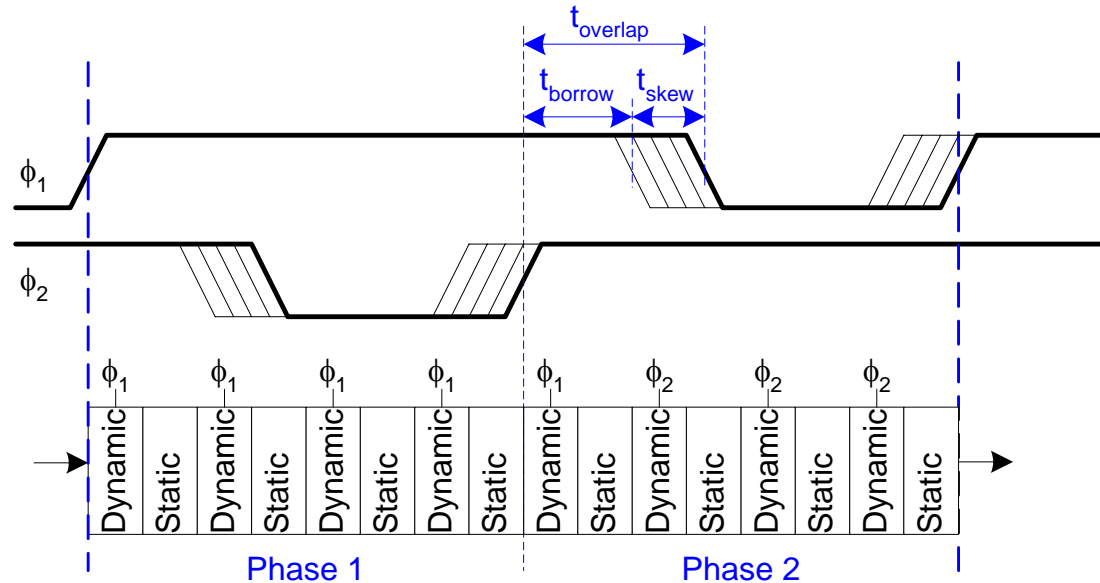
- After second phase evaluates, first phase precharges
- Input to second phase falls
  - Violates monotonicity?
- But we no longer need the value
- Now the second gate has a floating output
  - Need full keeper to hold it either high or low



# Time Borrowing

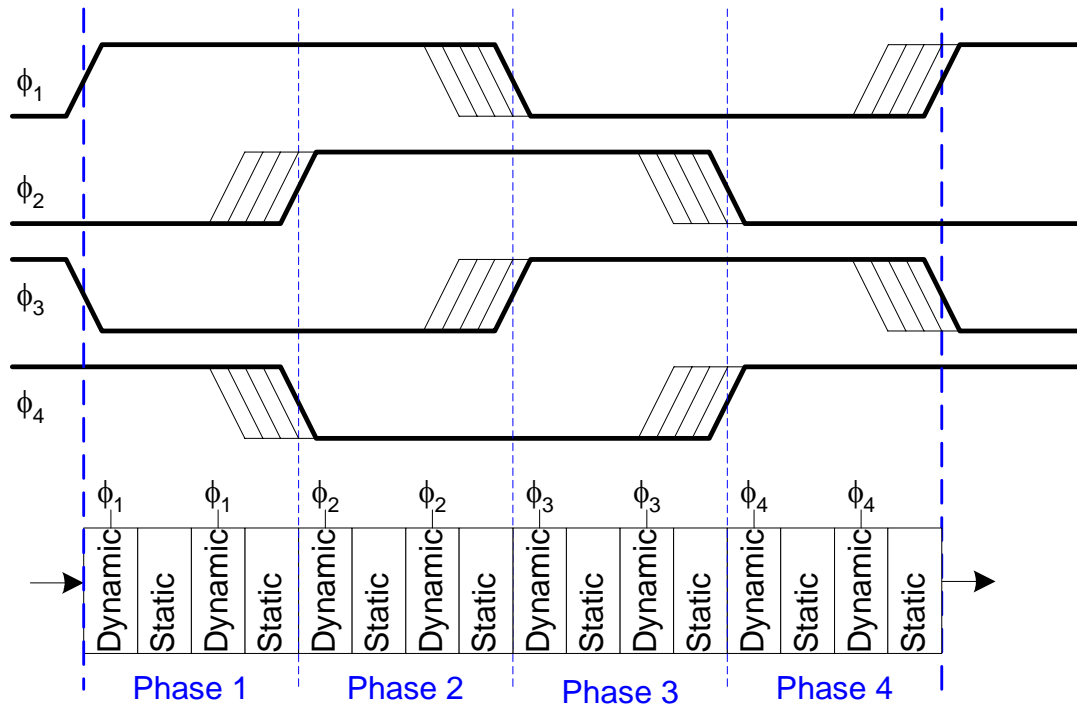
- Overlap can be used to
  - Tolerate clock skew
  - Permit time borrowing
- No sequencing overhead

$$t_{pd} = T_c$$

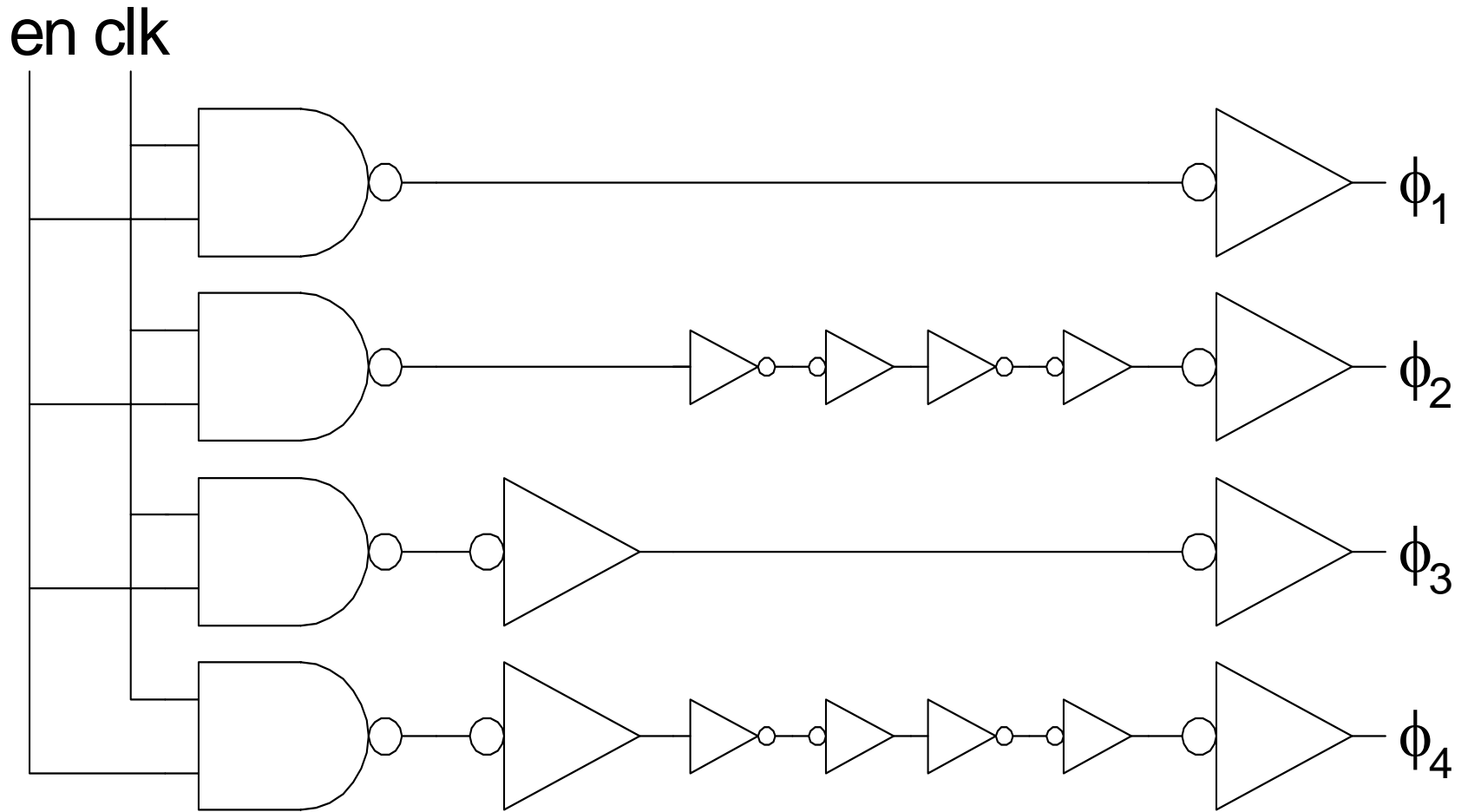


# Multiple Phases

- With more clock phases, each phase overlaps more
  - Permits more skew tolerance and time borrowing



# Clock Generation



# Summary

- Clock skew effectively increases setup and hold times in systems with hard edges
- Managing skew
  - Reduce: good clock distribution network
  - Analyze: local vs. global skew
  - Tolerate: use systems with soft edges
- Flip-flops and traditional domino are costly
- Latches and skew-tolerant domino perform at full speed even with moderate clock skews.