Skew-Tolerant Circuit Design

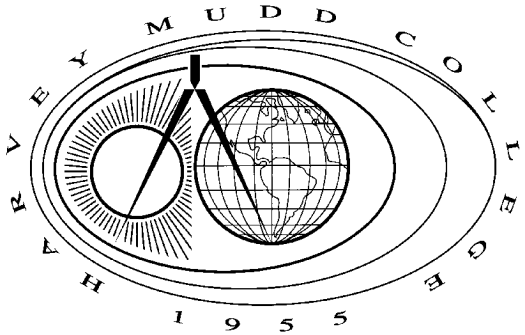
David Harris

David_Harris@hmc.edu

December, 2000

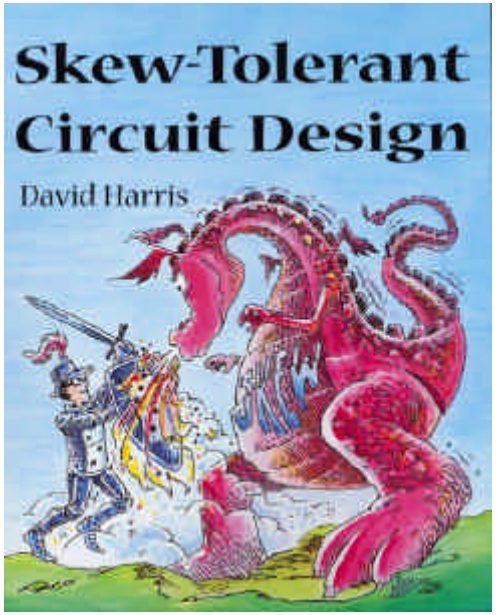
Harvey Mudd College

Claremont, CA



Outline

- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Example



Talk based on:

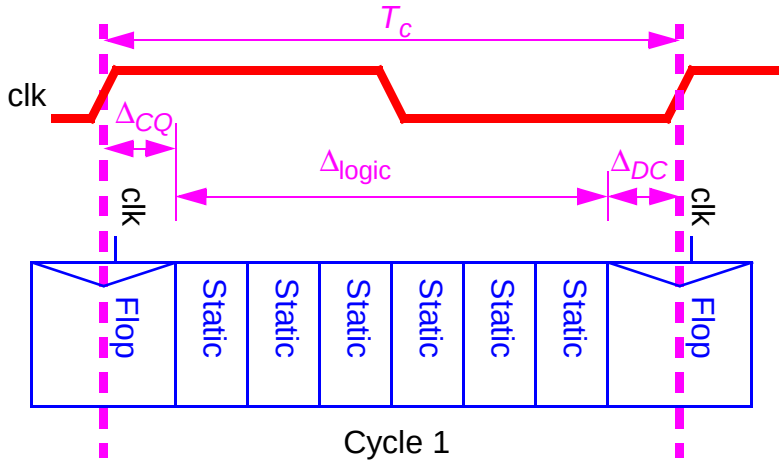
- *Skew-Tolerant Circuit Design*, Morgan Kaufmann Publishers, 2001.

Sequencing Overhead in Flip-Flop Systems

Ideally, each full clock cycle T_c is available for useful computation

But we must keep token in current pipestage from catching up with token in next

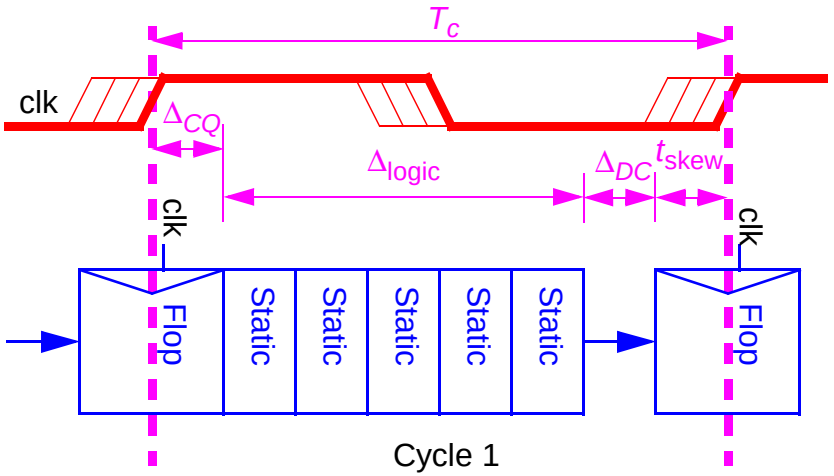
- Purpose of sequencing elements such as flip-flops is not to “remember” data but merely to retard fast paths so they don’t corrupt tokens further along in the pipeline
- This inherently retards the slower paths as well
- Introduces “sequencing overhead” inherent to keeping tokens sequenced



For flip-flops: $\Delta_{logic} = T_c - \Delta_{CQ} - \Delta_{DC}$

Clock Skew & Time Borrowing

Clock skew increases sequencing overhead



- Data may launch on the latest skewed clock edge
- Must setup before the earliest skewed clock edge
- Clock skew directly cuts into time available for computation

$$\Delta_{logic} = T_c - \Delta_{CQ} - \Delta_{DC} - t_{skew}$$

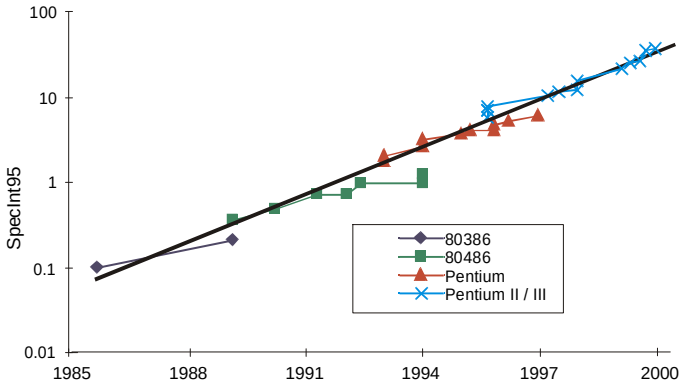
Any unused time at end of cycle is wasted

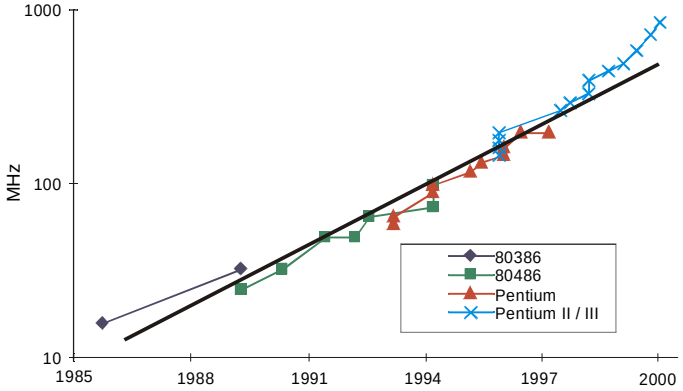
- No time borrowing possible to balance logic between stages

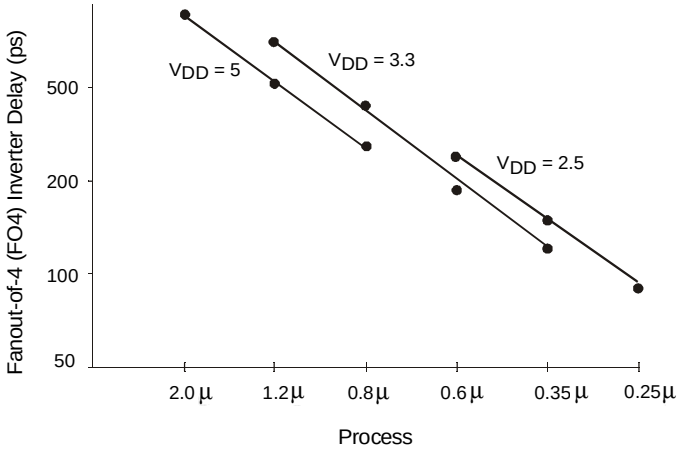
Cycle Time Trends

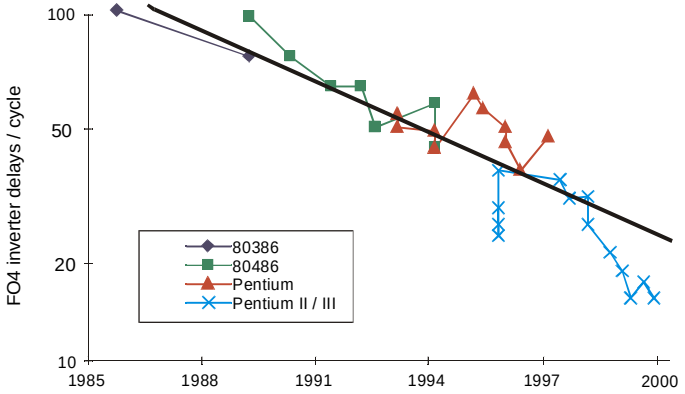
Much of CPU performance improvement comes from higher frequencies

- Clock rates increasing faster than simple process improvement allows
- Fixed sequencing overhead becomes greater portion of cycle time









SIA Frequency Roadmap

Semiconductor Industry Association has predicted clock frequencies

- Estimate cycle time in FO4 inverter delays
- Cycles are getting very short
- Minimizing sequencing overhead will be ever more important

Process (mm)	Year	Frequency (MHz)	Cycle time (FO4)
0.25	1997	750 (600)	13.3 (16.7)
0.18	1999	1250 (733)	11.1 (18.9)
0.15	2001	1500 (> 1500)	11.1 (<11.1)
0.13	2003	2100	9.2
0.10	2006	3500	7.1
0.07	2009	6000	6.0
0.05	2012	10000	5.0

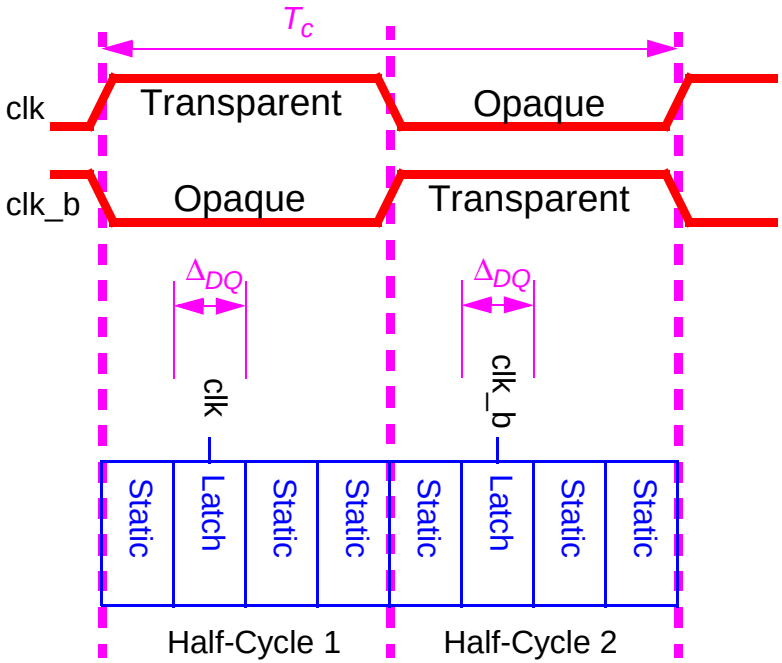
Outline

- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Example

Transparent Latches

Transparent latch-based systems use a latch in each half-cycle

- Latch is transparent when controlling clock is high, opaque when low
- Two latches required so tokens does not skip ahead to next pipeline stage



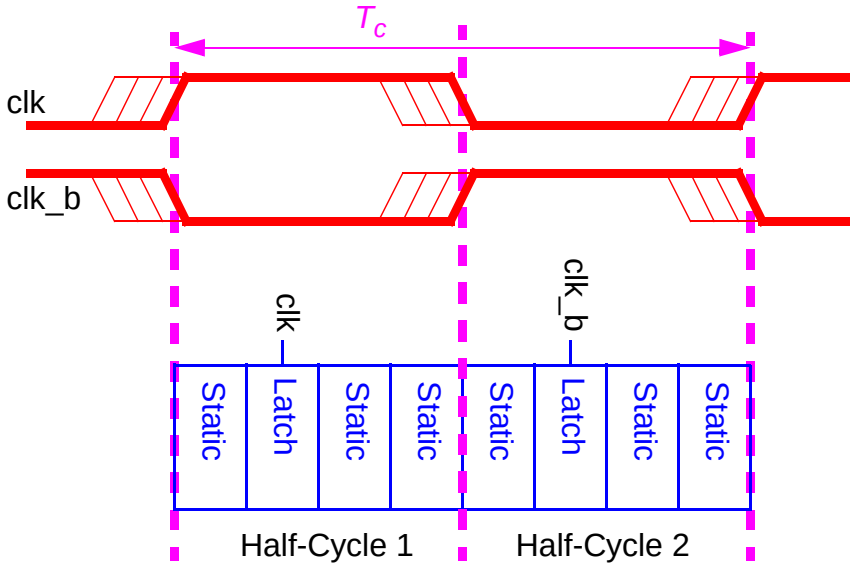
$$\Delta_{logic} = T_c - 2\Delta_{DQ}$$

Transparent Latches with Skew

Flip-flops are sensitive to skew because of the hard edges

- Data not launched until latest rising edge of clock
- Data must setup before earliest falling edge of clock
- Overhead would shrink if we could soften an edge

Latch-based systems may still use entire cycle even if there is skew

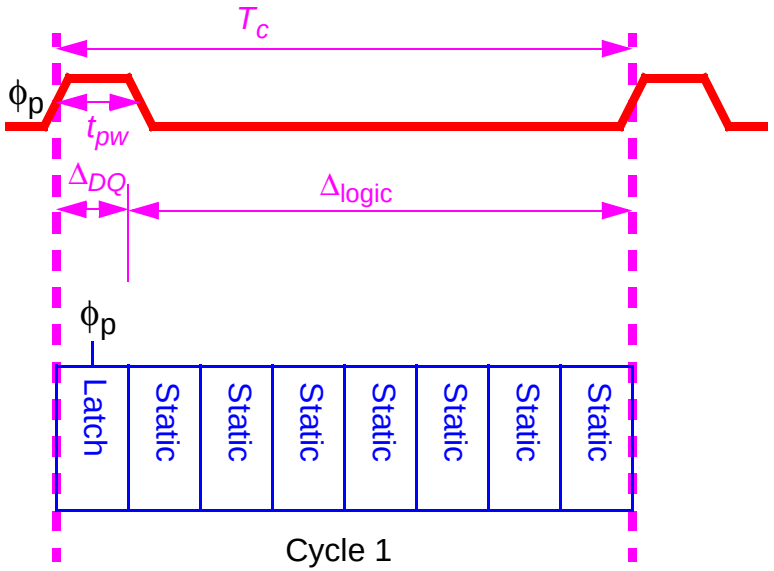


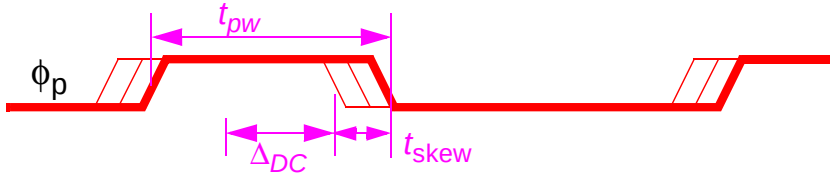
$$\Delta_{logic} = T_c - 2\Delta_{DQ}$$

Pulsed Latches

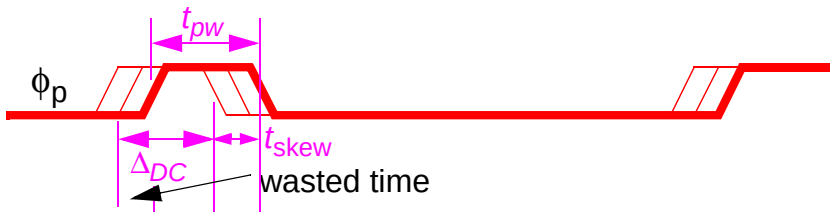
Can we keep tokens in sequence with only one latch per cycle?

- Pulse the latch transparent briefly compared to delay through pipestage





Case 1: pulse wider than setup time + clock skew
data arrives while latch is transparent, no time wasted



Case 2: pulse narrower than setup time + clock skew
data may arrive while latch is opaque, leaving wasted time

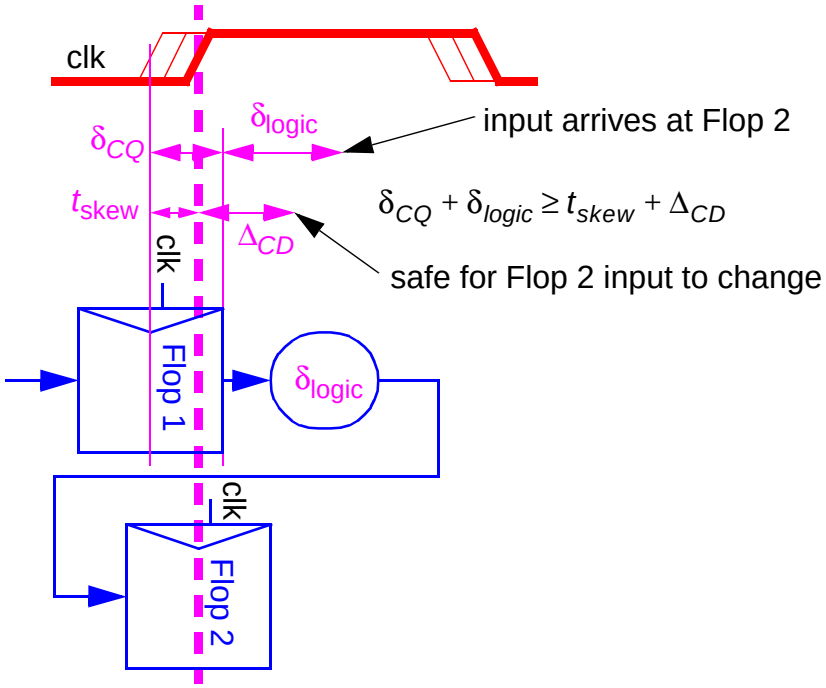
$$\Delta_{logic} = \begin{cases} T_c - \Delta_{DQ} & \text{if } t_{pw} \geq \Delta_{DC} + t_{skew} \\ T_c + t_{pw} - \Delta_{DQ} - \Delta_{DC} - t_{skew} & \text{otherwise} \end{cases}$$

Min-Delay

Min-Delay (a.k.a hold time or race problem) is especially insidious

- Setup time problems can be fixed by slowing clock
- Hold time problems require redesign of chip

Consider min-delay constraint for flip-flops with little intervening logic

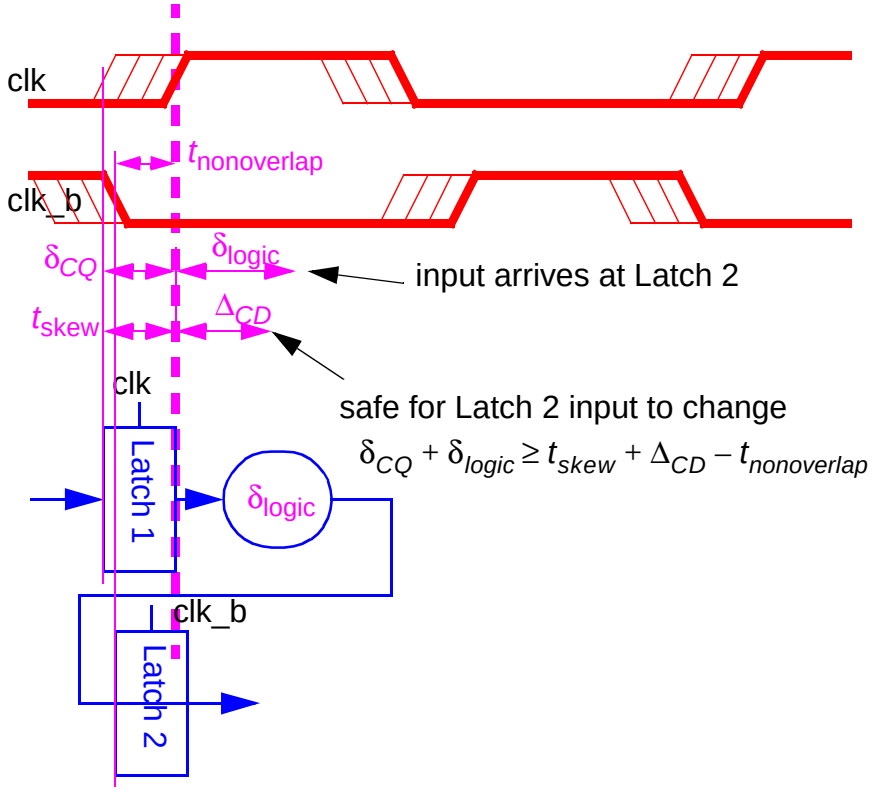


$$\delta_{logic} \geq \Delta_{CD} + t_{skew} - \delta_{CQ}$$

Min-Delay (Transparent Latches)

Latches may use nonoverlapping clocks to reduce min-delay problems

- But min-delay constraints exist between each latch (twice per cycle)



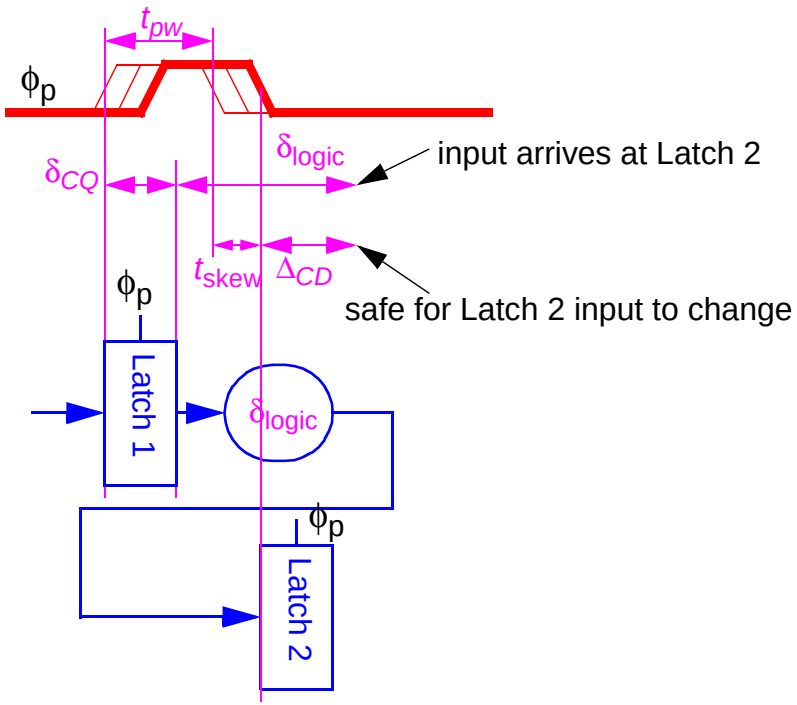
$$\delta_{logic} \geq \Delta_{CD} + t_{skew} - \delta_{CQ} - t_{nonoverlap}$$

(in each half-cycle paths)

Min-Delay (Pulsed Latches)

Pulsed latches with wide pulses are especially susceptible to min-delay

- Data launches off earliest rising edge of pulse
- Must hold until after latest falling edge of pulse

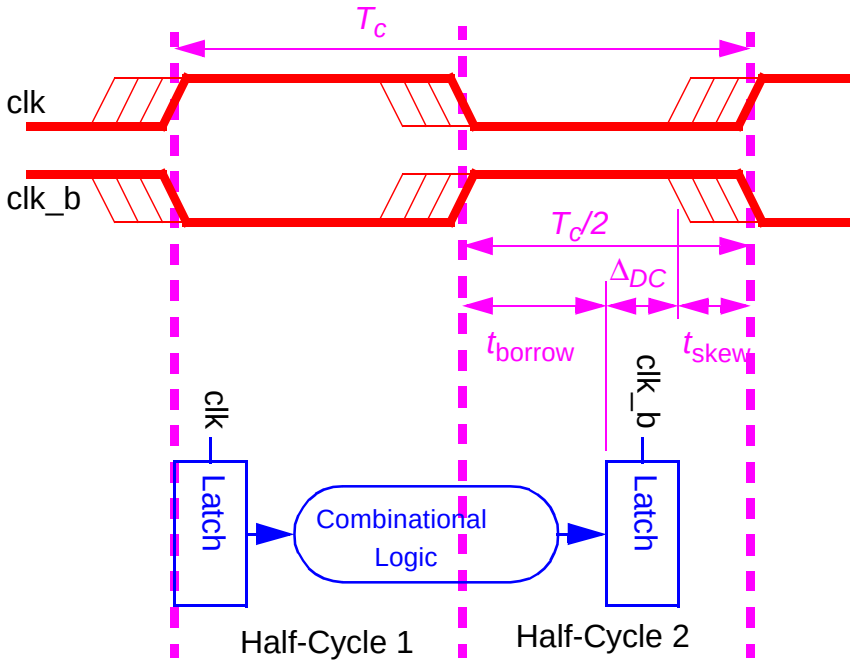


$$\delta_{logic} \geq t_{pw} + \Delta_{CD} + t_{skew} - \delta_{CQ}$$

Time Borrowing

Logic between latches may occupy more than exactly 1/2 cycle

- Borrow time so long as setup time at next latch is met
- Soft edges provide flexibility to partition and balance logic (unlike flops)

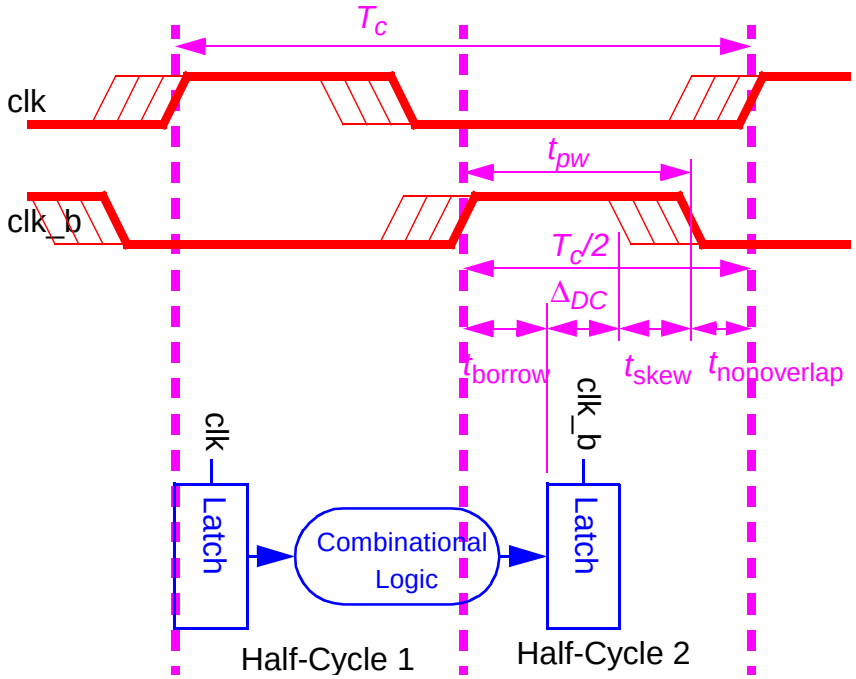


$$t_{borrow} = \frac{T_c}{2} - \Delta_{DC} - t_{skew}$$

Time Borrowing with Pulses

What if the clock duty cycle is less than 1/2 cycle (pulses or nonoverlap)?

- Still meet setup time before end of transparency pulse



$$t_{borrow} = \begin{cases} t_{pw} - \Delta_{DC} - t_{skew} & \textit{pulsed latches} \\ \frac{T_c}{2} - t_{nonoverlap} - \Delta_{DC} - t_{skew} & \textit{transparent latches} \end{cases}$$

Skew-Tolerant Static Circuits Summary

Flip-Flops:

- Popular because CAD tools and junior engineers understand them
- Worst sequencing overhead and no time borrowing

Transparent Latches:

- Better sequencing overhead
- Excellent time borrowing
- Understood by most but not all engineers and CAD tools

Pulsed Latches:

- Lowest sequencing overhead
- Can be treated like flip-flops
- Requires thorough min-delay analysis
- Minimal time borrowing

Element	Sequencing overhead	Time borrowing	Min-delay δ_{logic}
Flip-flop	$\Delta_{CQ} + \Delta_{DC} + t_{skew}$	0	$\Delta_{CD} + t_{skew} - \delta_{CQ}$
Transparent latch	$2\Delta_{DQ}$	$\frac{T_c}{2} - \Delta_{DC} - t_{skew} - t_{nonoverlap}$	$\Delta_{CD} + t_{skew} - \delta_{CQ} - t_{nonoverlap}$ (in each half-cycle)
Pulsed latch	$\Delta_{DQ} + max(0, \Delta_{DC} + t_{skew} - t_{pw})$	$t_{pw} - \Delta_{DC} - t_{skew}$	$t_{pw} + \Delta_{CD} + t_{skew} - \delta_{CQ}$

Outline

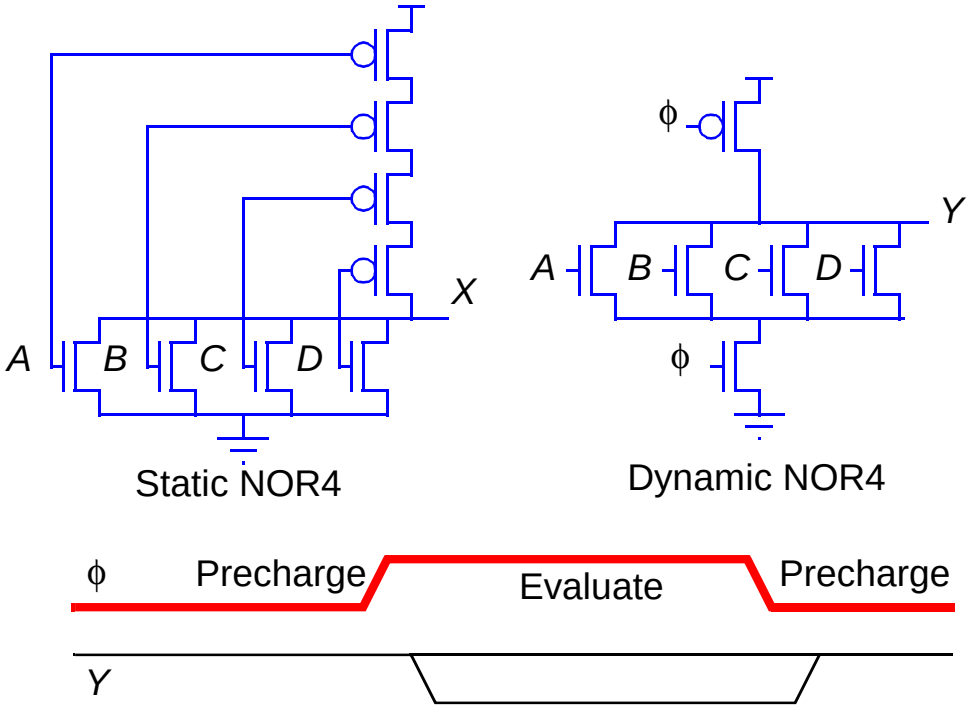
- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Example

Dynamic Circuit Review

Static CMOS circuits are slow because fat PMOS devices load input

Dynamic gates use precharged circuitry to remove PMOS from input

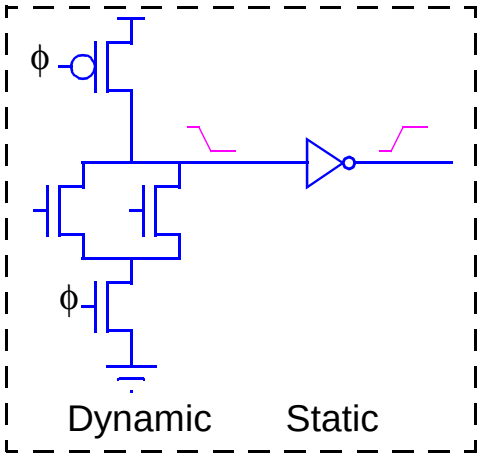
- Precharge: $\phi = 0$ output set high through clocked PMOS transistor
- Evaluate: $\phi = 1$ output possibly pulls low through logic transistors



Domino Circuits

Dynamic inputs must be monotonically rising during evaluation

- Place inverting static gate between each dynamic gate
- Dynamic / static pair is called a domino gate



Domino Gate

Domino gates can be safely cascaded

- Inputs from static CMOS logic must still be latched to avoid glitches

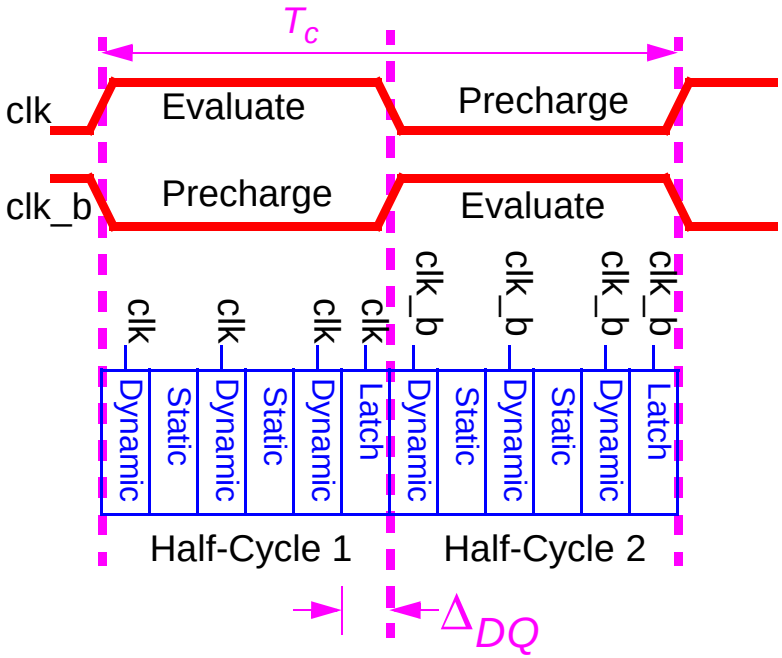
Domino evaluation delays 1.5 - 2x faster than static CMOS

- Challenge is how to prevent precharge time from impacting critical path
- Traditional domino clocking approaches introduce severe overhead
- Skew-tolerant domino circuits can hide this overhead

Traditional Domino Circuits

How do we prevent precharge time from appearing in the critical path?

Hide precharge time by ping-ponging between half-cycles

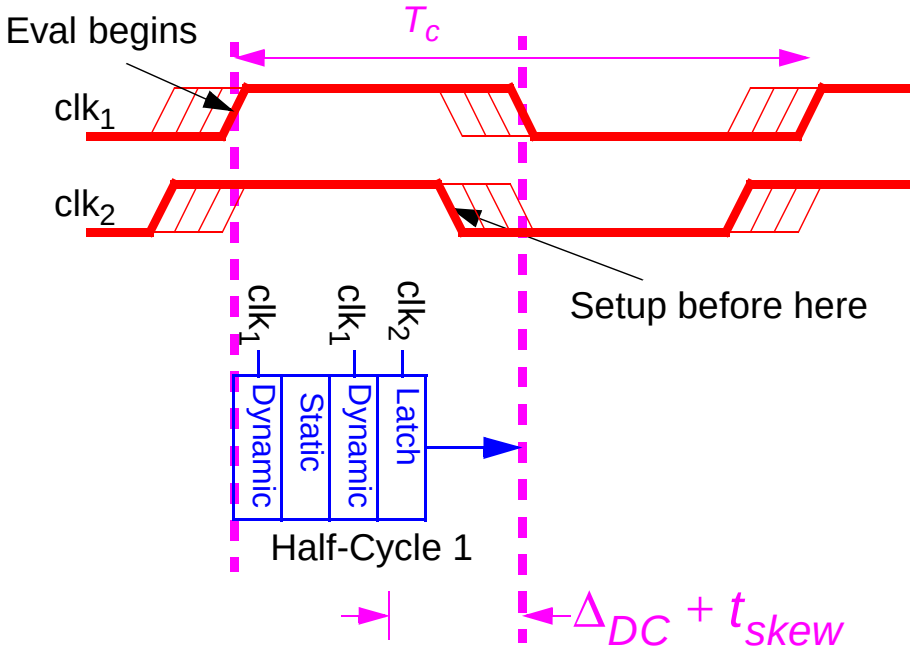


- One evaluates while other precharges
- Latches hold results during precharge

$$\Delta_{logic} = T_c - 2\Delta_{DQ}$$

Clock Skew

Clock skew increases sequencing overhead

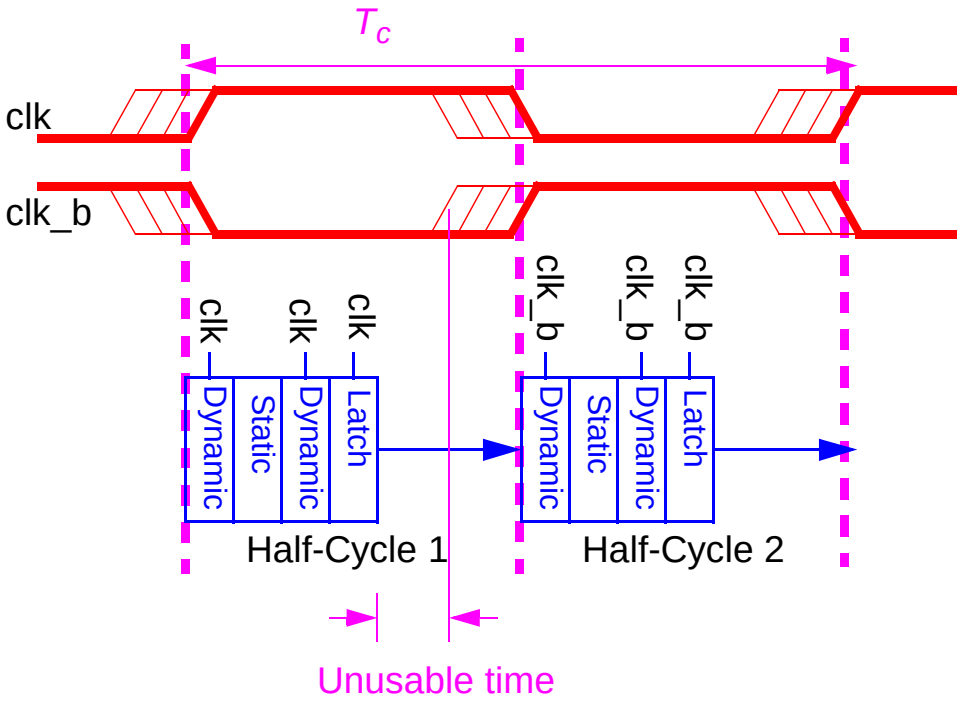


- Evaluation begins at latest rising edge
- Latch input setup before earliest falling edge
- Budget clock skew twice each cycle

$$\Delta_{logic} = T_c - 2\Delta_{DC} - 2t_{skew}$$

Balancing Logic

Logic may not exactly fit half-cycle



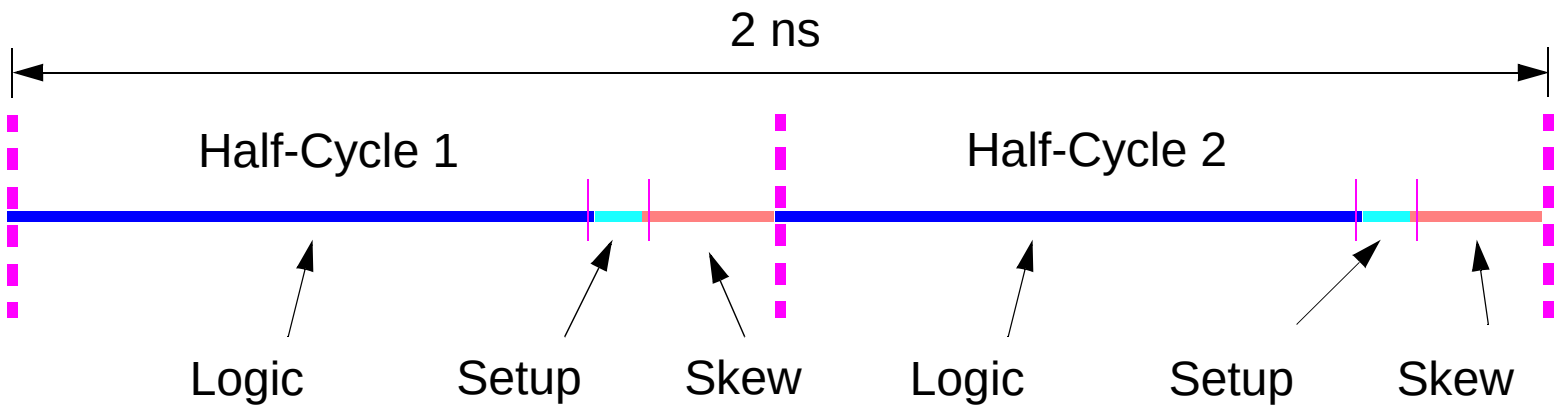
- No flexibility to borrow time to balance logic between half-cycles

$$\Delta_{logic} = T_c - 2\Delta_{DC} - 2t_{skew} - t_{imbalanced}$$

How bad is it?

Consider 500 MHz system similar to Alpha 21164

- T_c = 2ns
- t_{skew} = 200 ps (reported budget)
- Δ_{DC} = 50 ps (optimistic estimate)



25% of cycle consumed by skew & setup!

Relaxing the Timing Constraints

Sequencing overhead caused by hard edges (just as for flip-flops)

- Data not launched until latest rising edge of clock
- Data must setup before earliest falling edge of clock
- Overhead would shrink if we could soften an edge
- Let's look at the role of the latch on the falling edge

Latch function

(1) Prevent glitches on inputs of domino gates

(2) Hold results during precharge

Latches would be unnecessary if we could do these tasks in another way

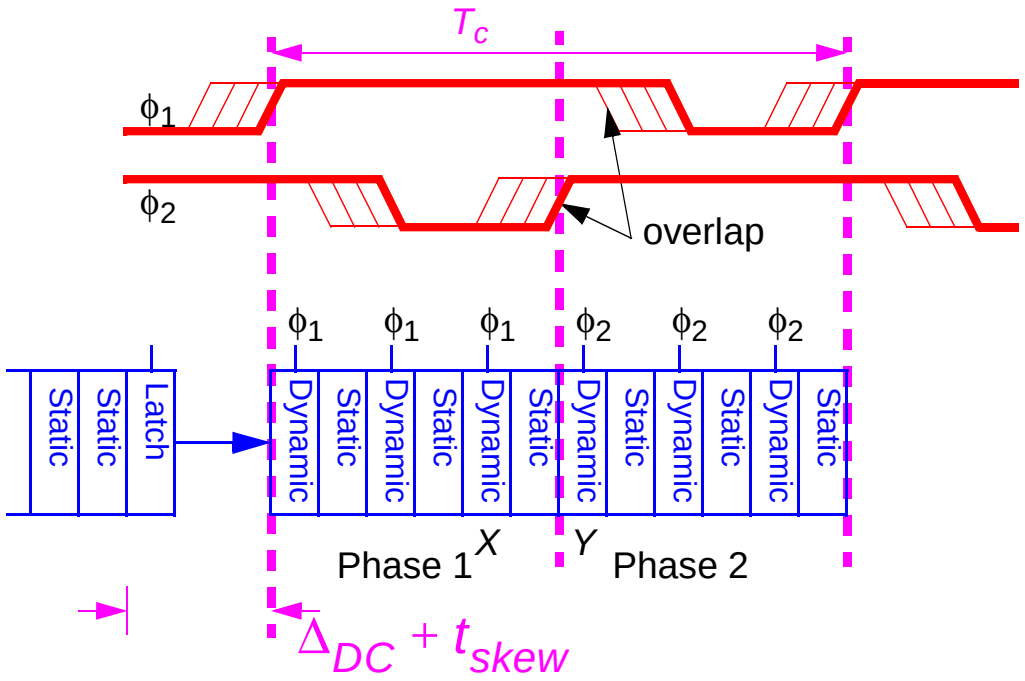
- Glitches will not occur if inputs come from domino logic
- Can we build domino circuits which consume the results before previous half-cycle precharges?

Outline

- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Example

Skew-Tolerant Domino Circuits

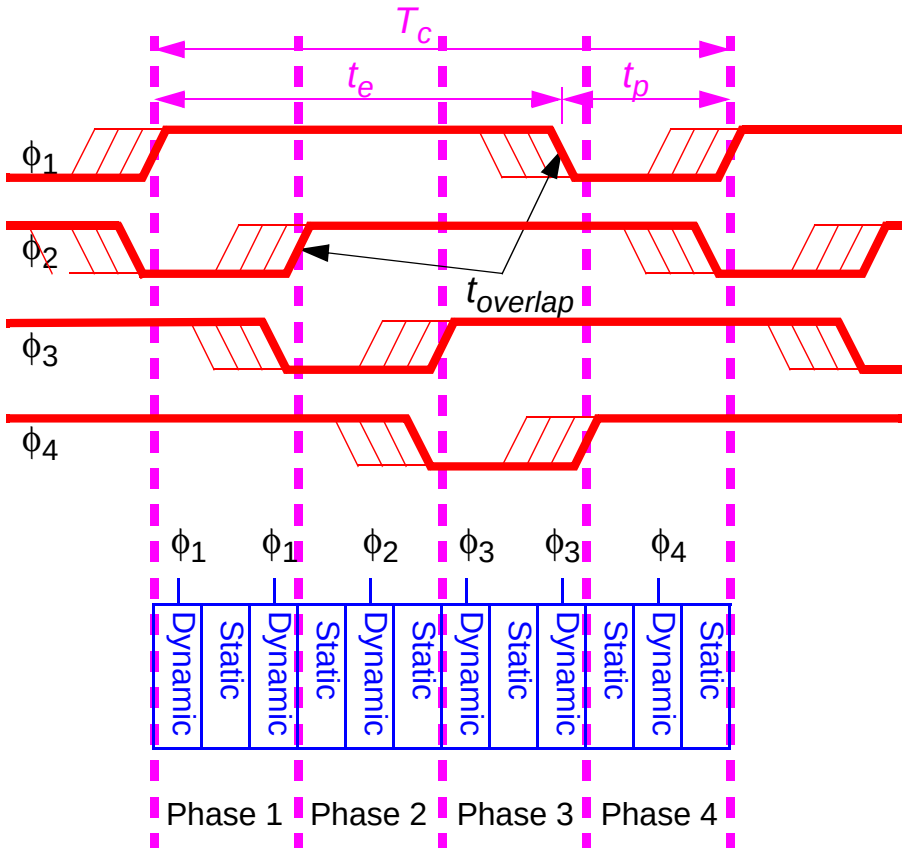
Overlap clocks so gate *Y* evaluates before *X* precharges



- We can remove latches within domino pipeline
- Eliminate all sequencing overhead within the domino pipeline
- Still budget skew and setup at static / domino interface

Skew-Tolerant Domino Waveforms

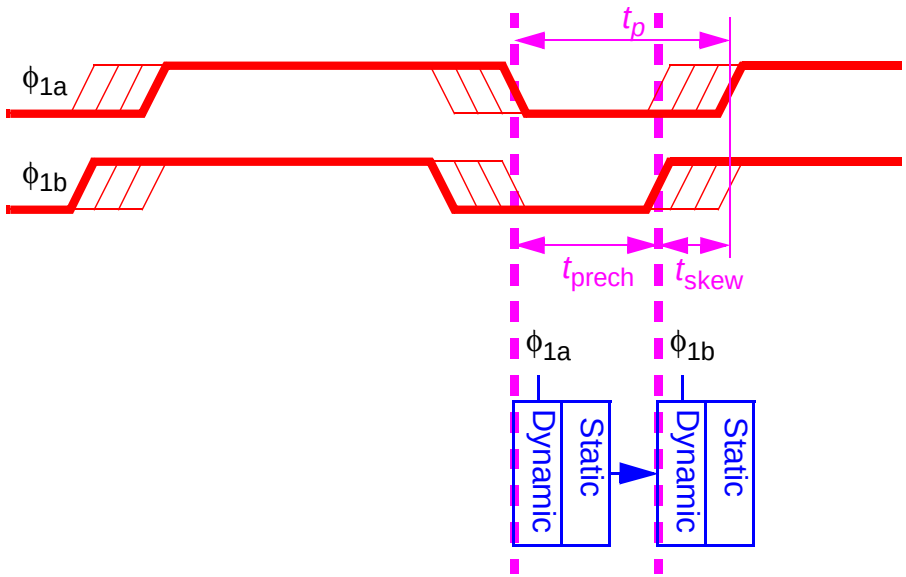
In general, we can use *N* phases



- Each phase delayed T_c/N from the previous
- What is the best duty cycle (t_e and t_p)?

Precharge Time (t_p)

Precharge time set by precharge of two gates in a phase with skewed clocks

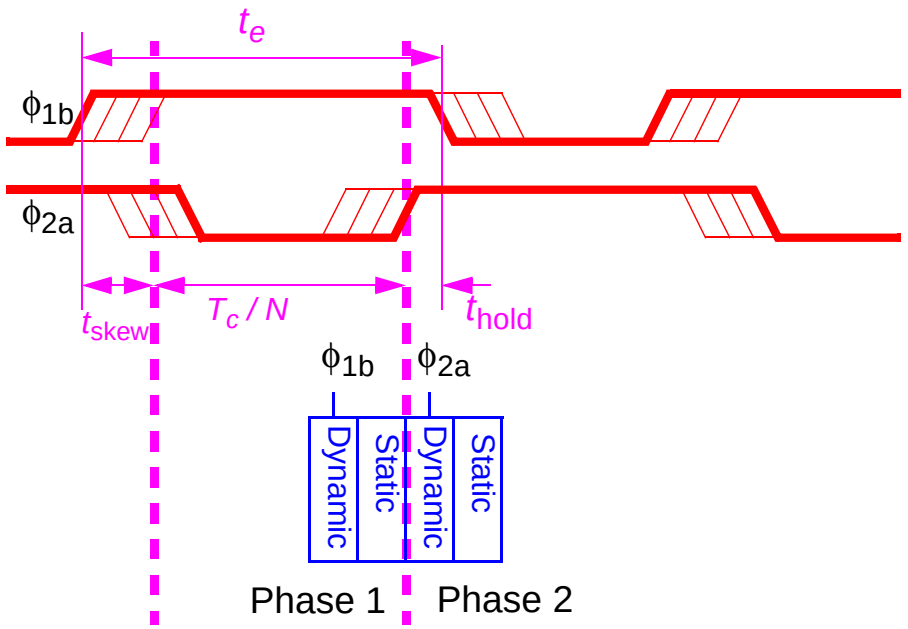


- First gate begins precharging late
- Requires t_{prech} to fully precharge
- Precharge must complete before second gate reenters evaluation

$$t_p = t_{prech} + t_{skew}$$

Evaluation Time (t_e)

Evaluation time set by next phase consuming result before first phase precharges



- First phase begins evaluation early
- Must not precharge until some hold time t_{hold} after second phase evaluates

$$t_e = \frac{T_c}{N} + t_{hold} + t_{skew}$$

Skew Tolerance

We've found the best duty cycle

- Precharge: $t_p = t_{prech} + t_{skew}$
- Evaluation: $t_e = \frac{T_c}{N} + t_{hold} + t_{skew}$

Solve for the maximum tolerable skew

$$t_{skew\text{-}max} = \frac{\frac{N-1}{N} T_c - t_{prech} - t_{hold}}{2}$$

Observations

- Skew tolerance increases with *N*
- Precharge and hold time cut into skew tolerance
- In the limit of large *N* and long cycles, skew tolerance ~ 1/2 cycle

Local Skew

Local clock domains

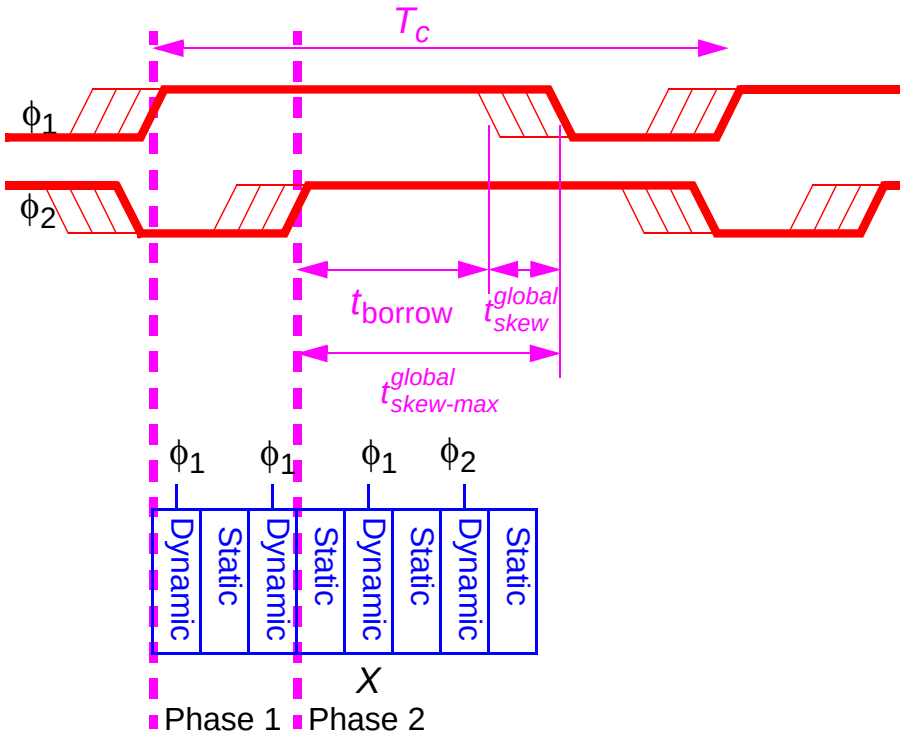
- Most circuits experience less skew in a small area than across chip
- Reduce skew within a phase of logic, thereby tolerating more skew globally

- $t_p = t_{prech} + t_{skew}^{local}$
- $t_e = \frac{T_c}{N} + t_{hold} + t_{skew}^{global}$
- $t_{skew\text{-}max}^{global} = \frac{N-1}{N} T_c - t_{prech} - t_{hold} - t_{skew}^{local}$
- This is the nominal overlap between phases

Time Borrowing

Time borrowing

- Excess overlap allows time borrowing into next phase (e.g. gate *X*)

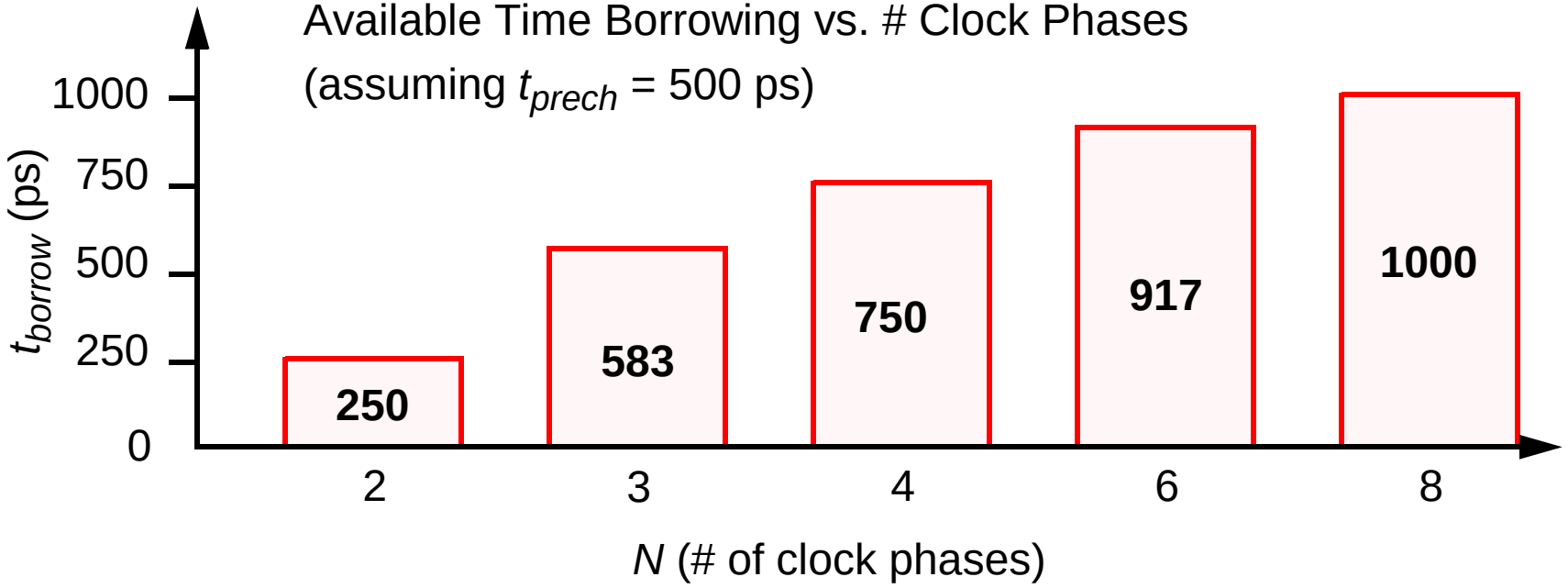


- $t_{borrow} = t_{skew-max}^{global} - t_{skew}^{global} = \frac{N-1}{N} T_c - t_{prech} - t_{hold} - t_{skew}^{local} - t_{skew}^{global}$

Example and Summary

Again consider a 500 MHz processor:

- T_c = 2ns t_{skewG} = 200 ps / t_{skewL} = 50 ps t_{hold} = 0 (conservative)



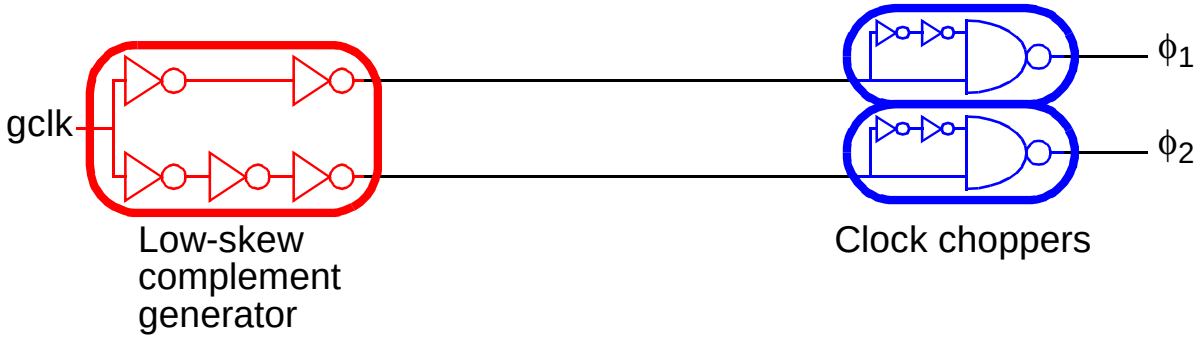
Domino Style	Sequencing overhead	Skew Tolerance	Time borrowing
Traditional	$2\Delta_{DC} - 2t_{skew} - t_{imbalanced}$	n/a	0
Skew-Tolerant	0	$\frac{N-1}{N}T_c - t_{prech} - t_{hold} - t_{skew}^{local}$	$\frac{N-1}{N}T_c - t_{prech} - t_{hold} - t_{skew}^{local} - t_{skew}^{global}$

Outline

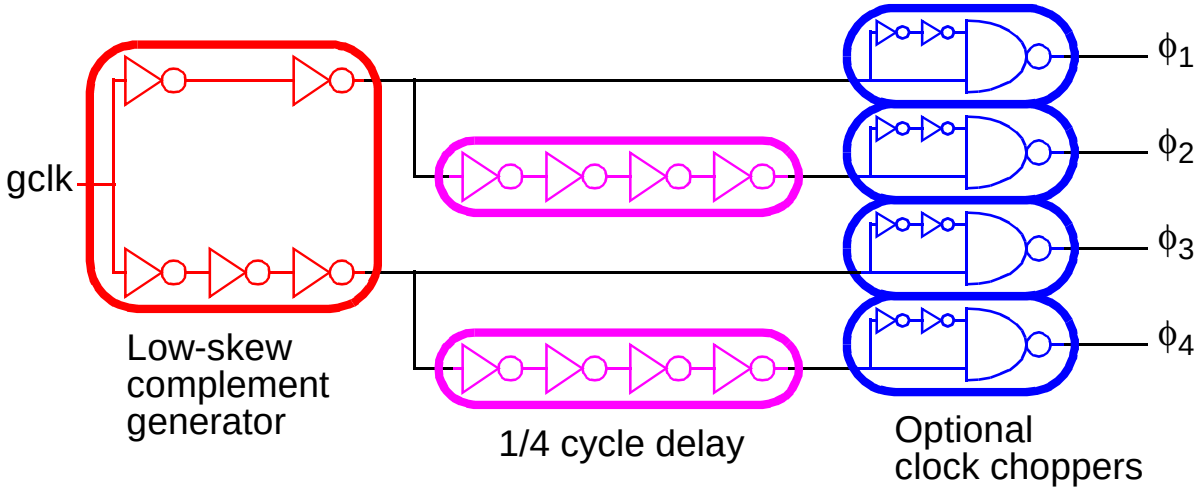
- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Results

Clock Generation

Two-Phase Clock Generator



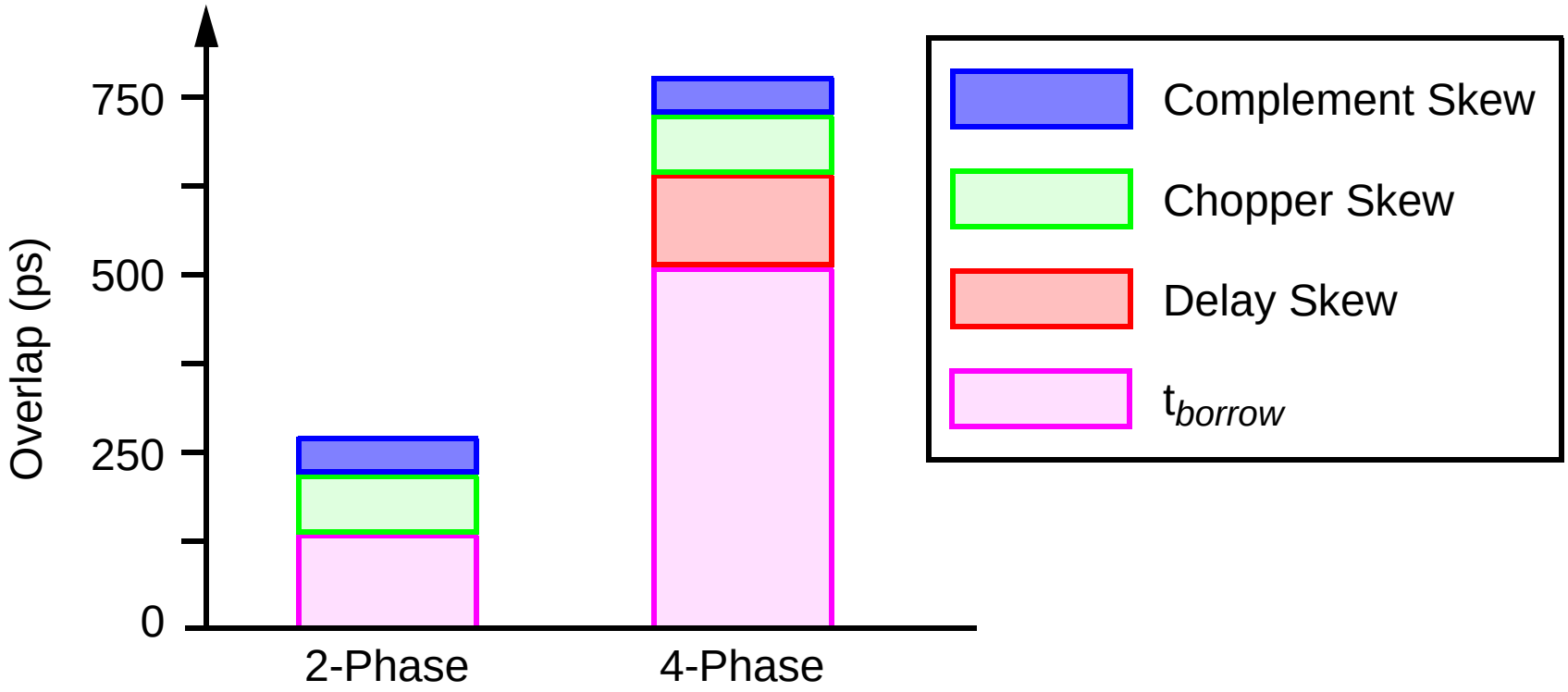
Four-Phase Clock Generator



- Inverters used to delay clock by quarter of nominal cycle time

Generator-Induced Clock Skew

Variation in clock generator delays appears as clock skew



Variation in delay 20-30% relative to other clocks

- 4-Phase is still a large net benefit

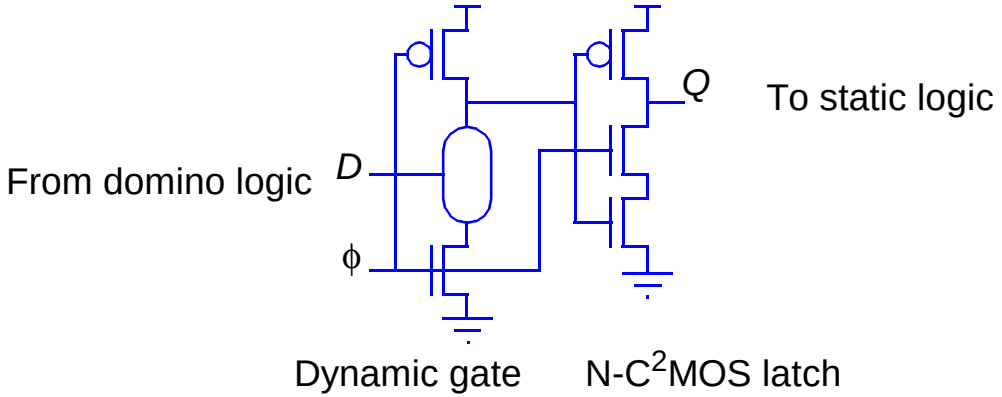
Interface with Static Logic

Real systems mix both static and domino logic

- Transparent latches use ϕ_1 and ϕ_3
- Pulsed latches use ϕ_p (see book)
- Four-phase skew-tolerant domino uses ϕ_1, ϕ_2, ϕ_3, and ϕ_4
- Which connections are legal?

Domino outputs must be latched before driving static logic

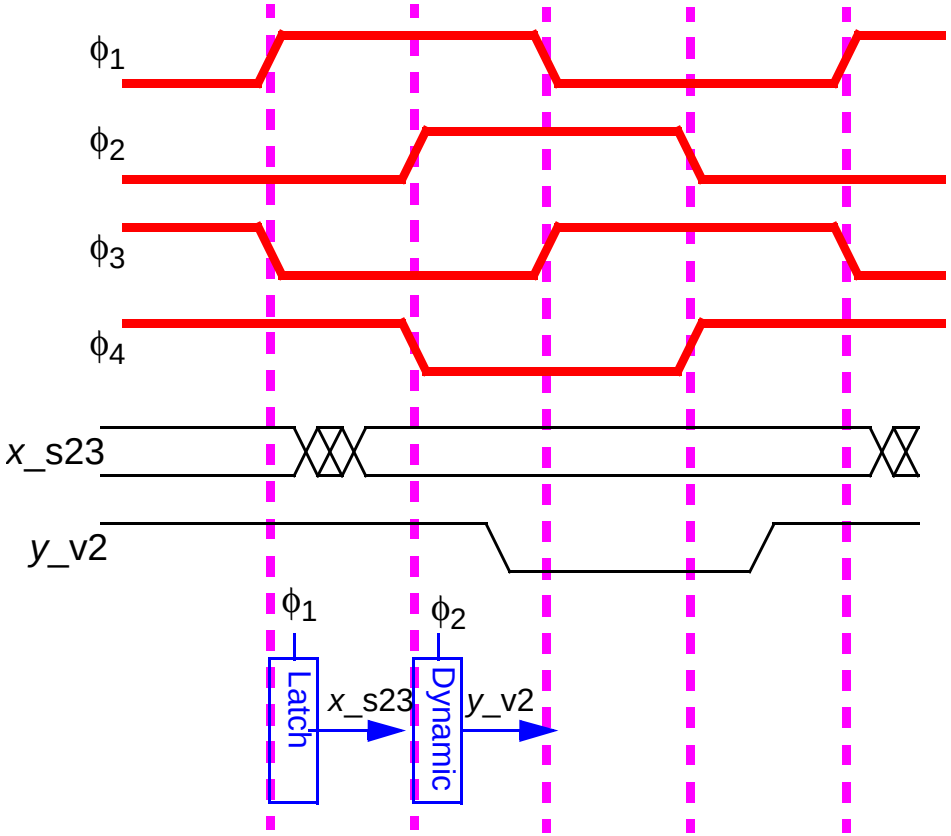
- Use N-C^{2}MOS Latch for speed and to avoid clock skew problems



Timing Types

Timing types define legal interfaces between gates

- Signal suffixes define when data is legal to use, allow static verification
- _s indicates that a signal is stable throughout that phase
- _v indicates that a signal becomes valid before the end of that phase



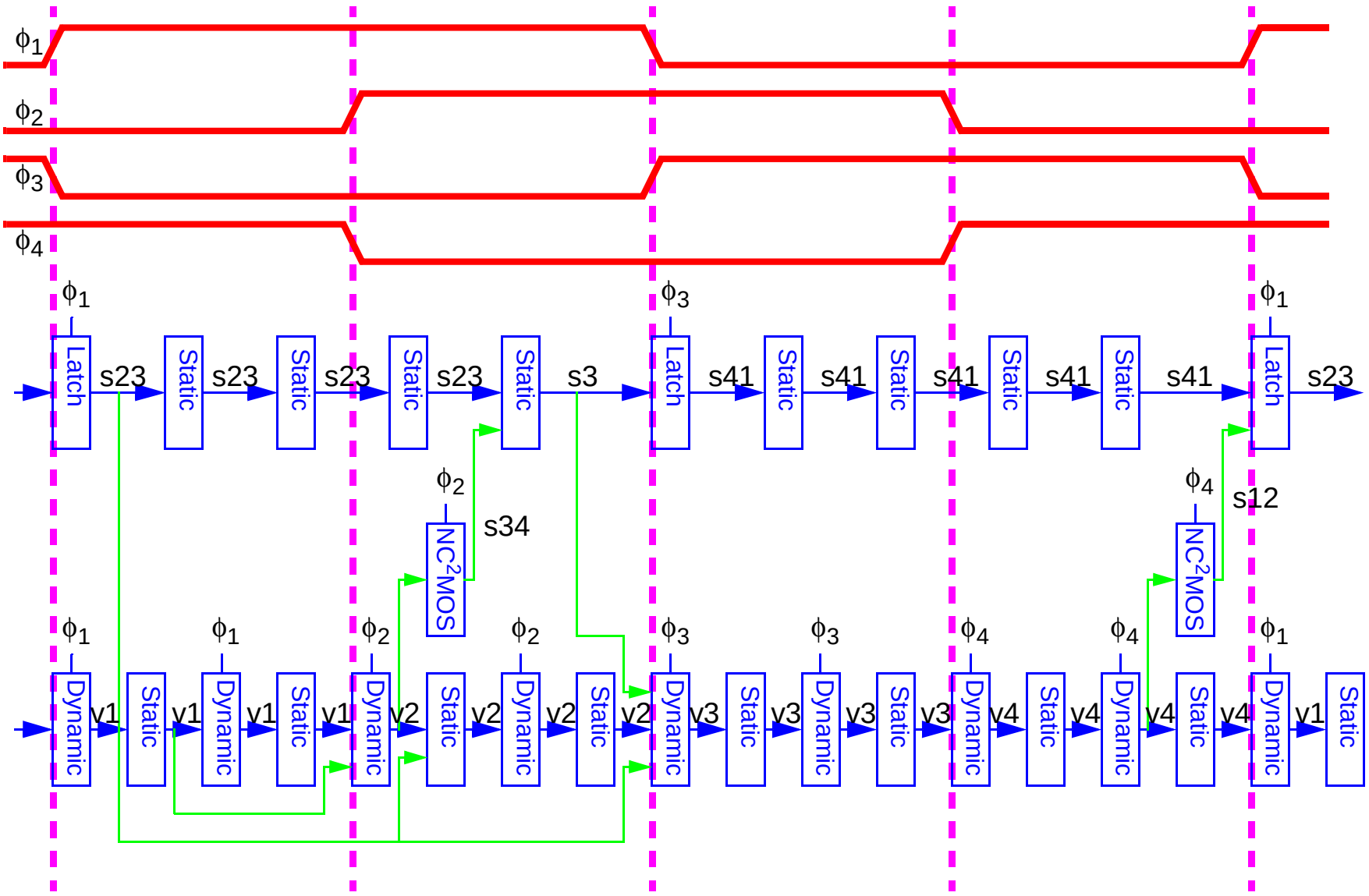
4-Phase Skew-Tolerant Domino Timing Types

Static gate output suffix is same as input suffix

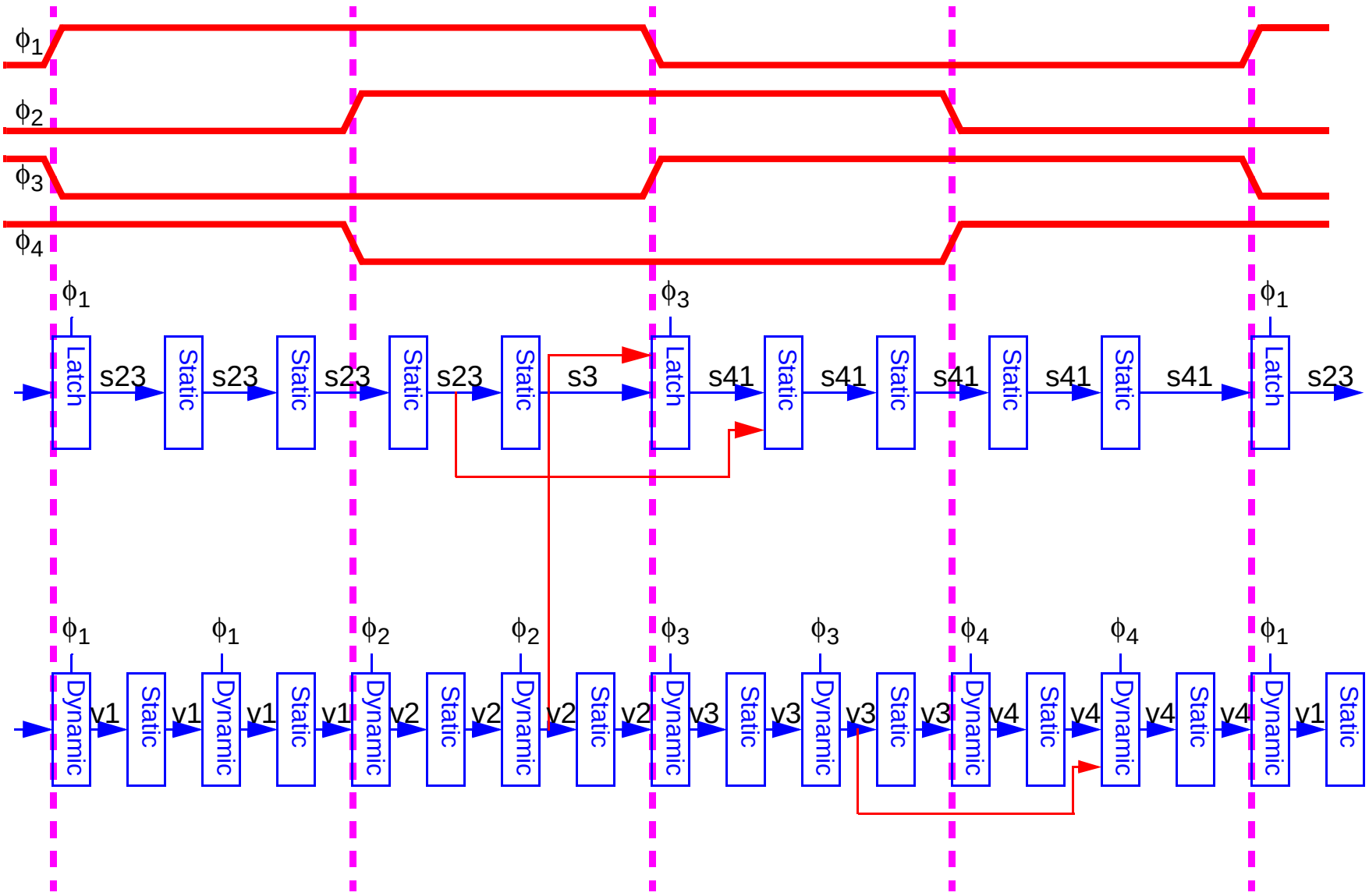
Clocked gates and latches set output suffix

Element Type	Clock	Input	Output
Domino	ϕ_1	_s1, _v4 (first) / v1 (others)	_v1
	ϕ_2	_s2, _v1 (first) / _v2 (others)	_v2
	ϕ_3	_s3, _v2 (first) / _v3 (others)	_v3
	ϕ_4	_s4, _v3 (first) / _v4 (others)	_v4
Transparent Latch	ϕ_1	_s1	_s23
	ϕ_3	_s3	_s41
N-C^{2}MOS Latch	ϕ_2	_v2	_s34
	ϕ_4	_v4	_s12

Example of Legal Connections



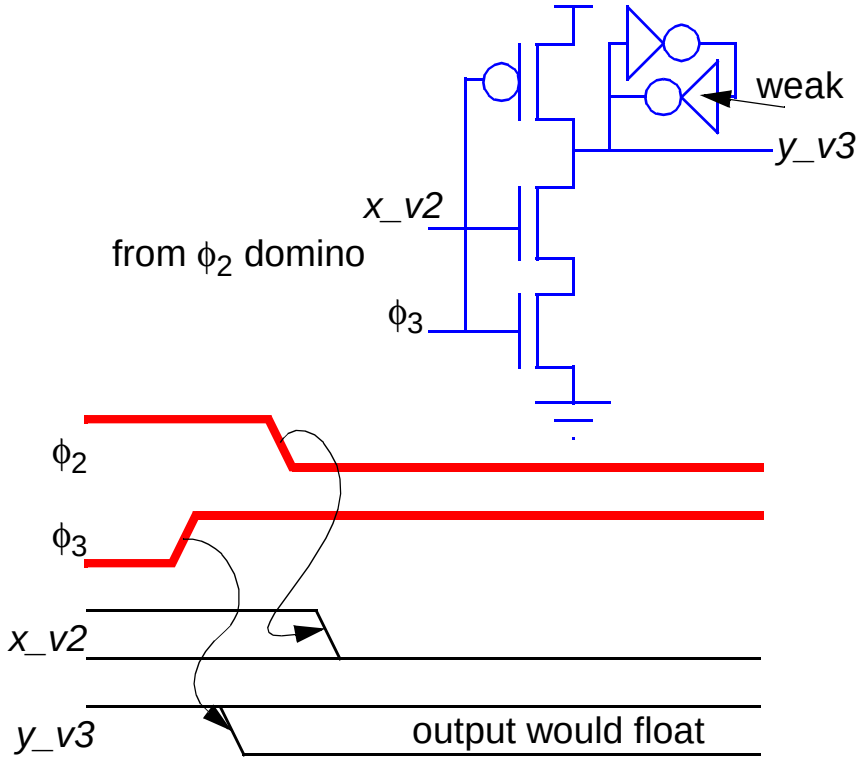
Example of Illegal Connections



Testability

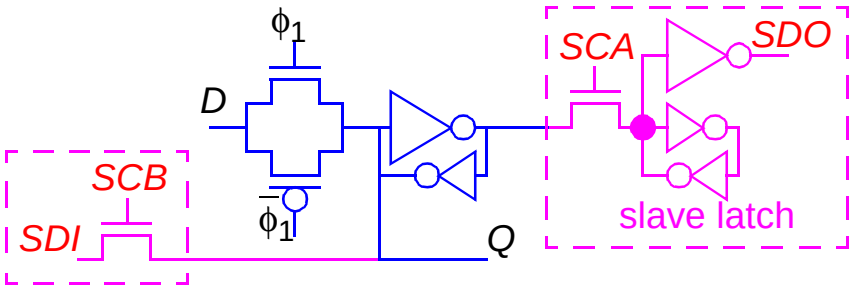
To test chips, we would like to be able to stop the clock and observe and control one element in each path through each cycle of logic.

- Stop with gclk low (ϕ_1 and ϕ_2 low, ϕ_3 and ϕ_4 high)
- Add full keeper to first ϕ_3 gate of each cycle to avoid floating high or low
- Different from half keeper of ordinary dynamic gates which only float high



Transparent Latch Scan

Add slave latch and access transistor to scan ϕ_1 static latches



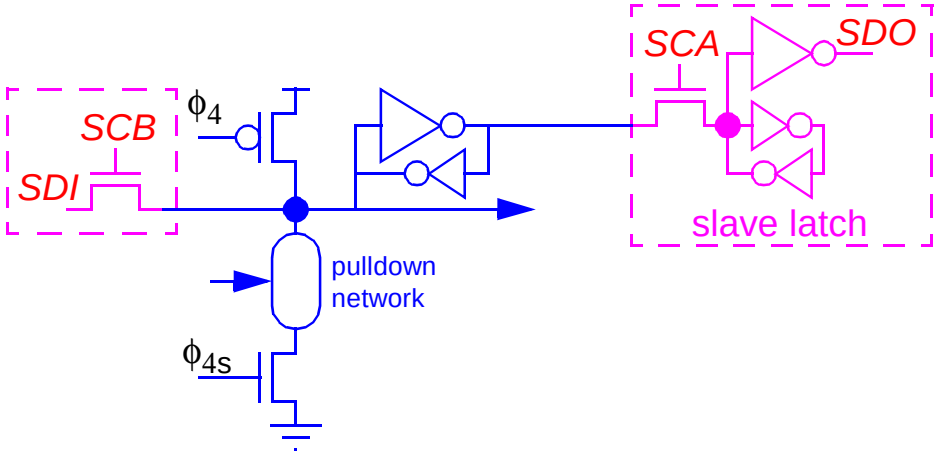
Scan Procedure:

1. Stop gclk low
2. Toggle *SCA* and *SCB* to march data through the scan chain
3. Restart gclk

How do we scan cycles of domino logic with no transparent latches?

Domino Scan

Scanning domino logic is very similar. Which gate should be scanned?



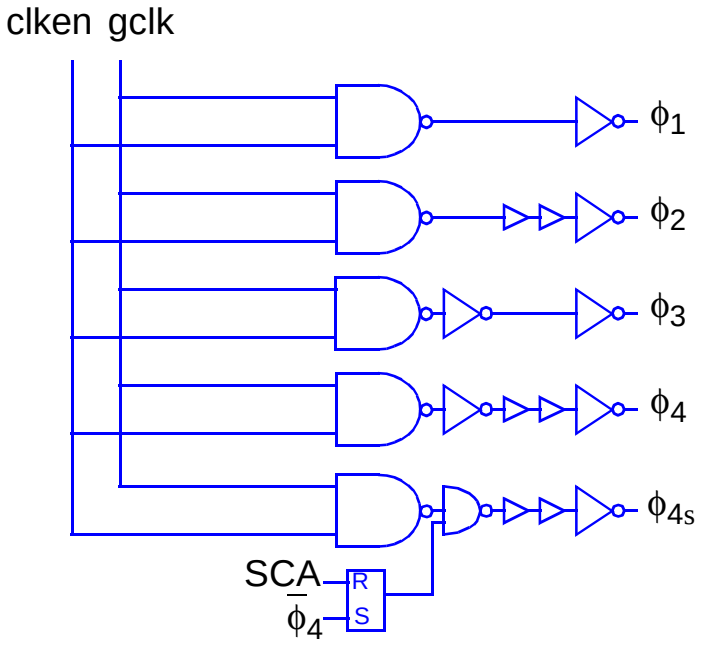
- Next gate should be precharging so glitches do not propagate
- When normal operation resumes, scan data should remain until consumed
- Thus, choose last ϕ_4 domino gate of each cycle for scan

Scan Procedure

1. Stop gclk low
2. Stop ϕ_{4s} low
3. Toggle *SCA* and *SCB* to march data through the scan chain
4. Restart gclk
5. Release ϕ_{4s} once scannable gate begins precharge (race)

Improved Clock Generator

Putting together skew-tolerant domino, stop clock, and scan:

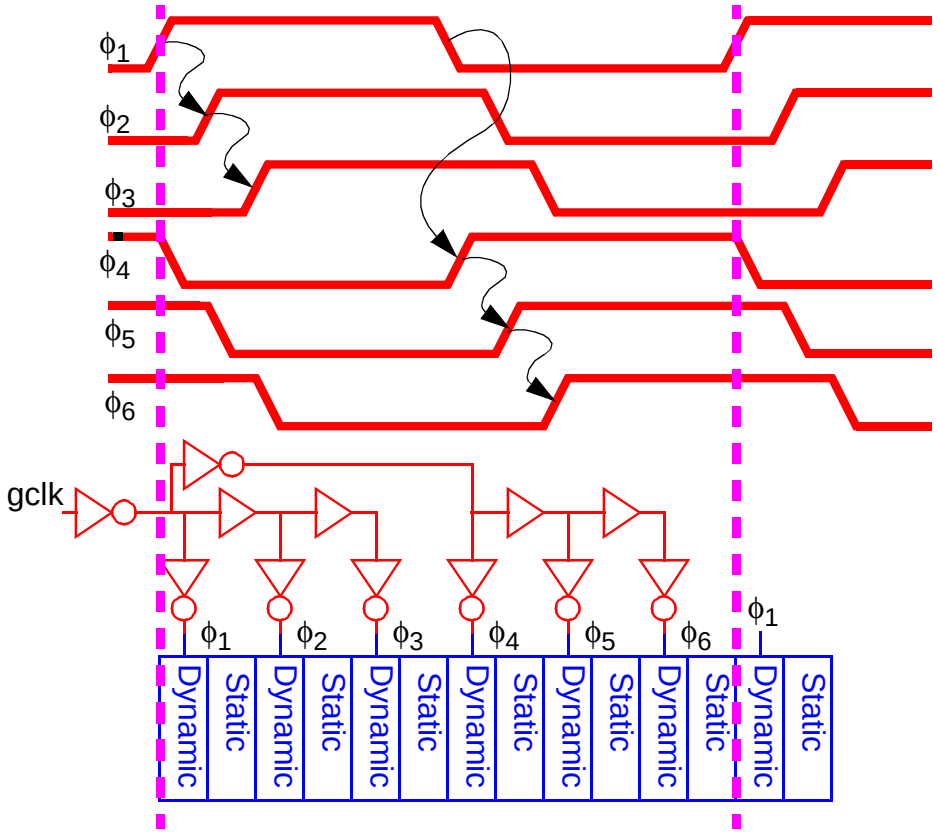


- SR latch solves the race reenabling ϕ_{4s}
  1. Stop gclk low
  2. Toggle *SCA* and *SCB* to march data through the scan chain. The first pulse of SCA will force ϕ_{4s} low.
  3. Restart gclk. The falling edge of ϕ_4 will release ϕ_{4s} to track ϕ_4.

Delayed Clocking / Delayed Reset Domino

Alternative approach uses a single clock phase per gate:

- Locally generate clocks with matched delay buffers

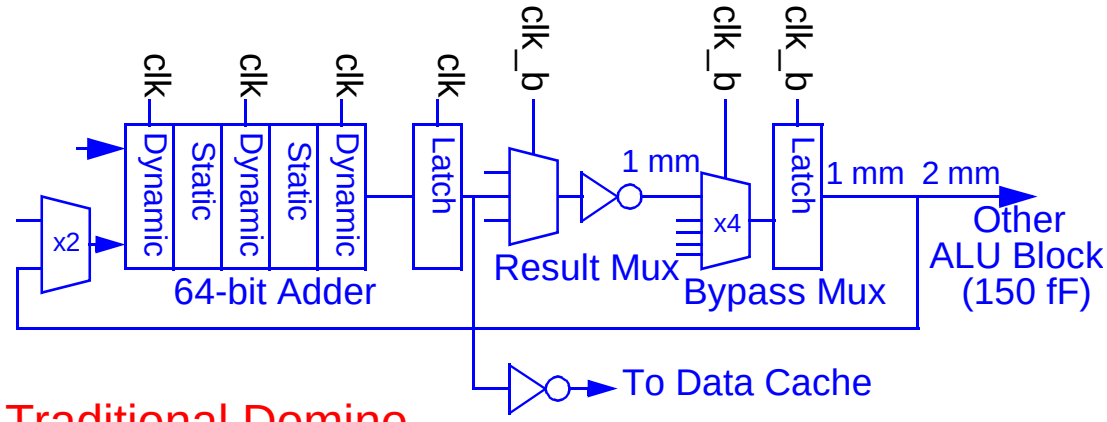


Outline

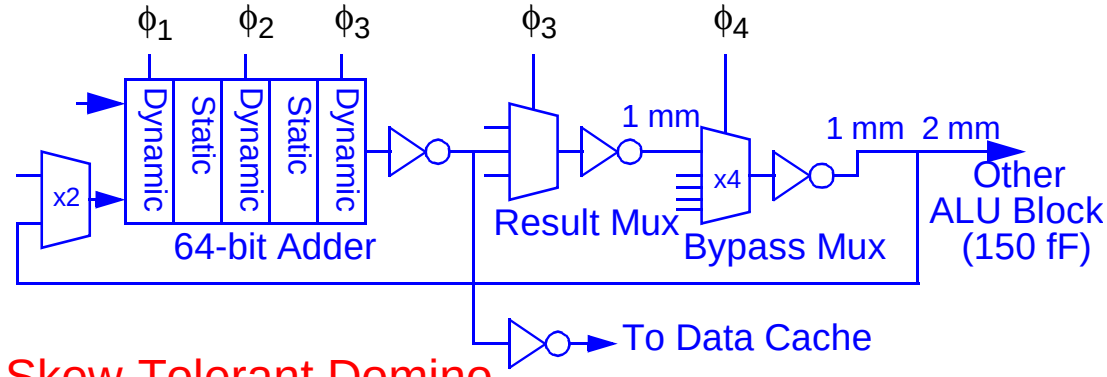
- Introduction
- Skew-Tolerant Static Circuits
- Traditional Domino Circuits
- Skew-Tolerant Domino Circuits
- Design Methodology
- Example

Example

64-bit superscalar ALU self-bypass path



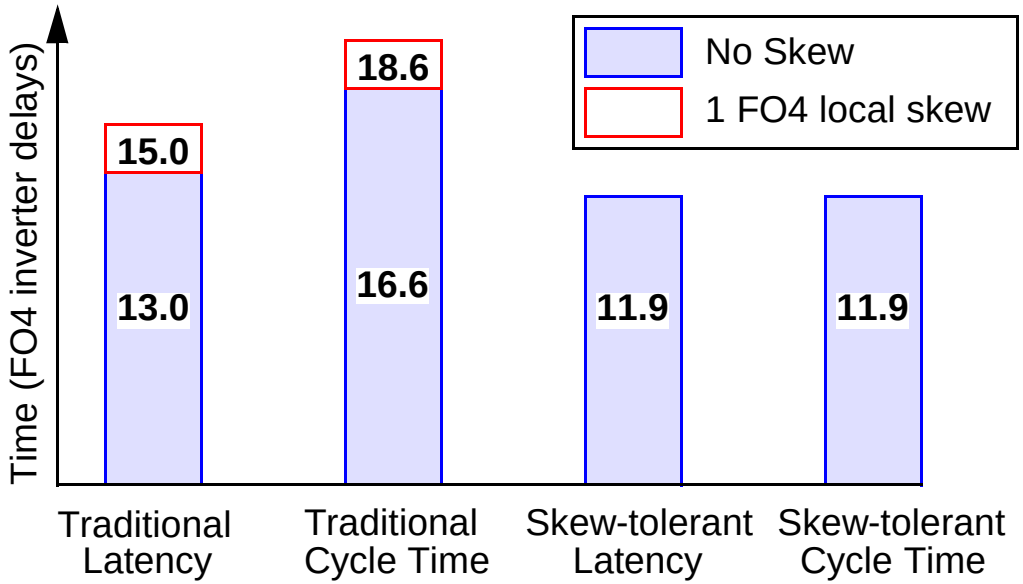
Traditional Domino



Skew-Tolerant Domino

Simulation Results

Adders were simulated in 0.6µ HP14 technology



- Skew-tolerant domino 25%$^{+}$ faster

Skew-Tolerant Domino Conclusions

Domino is very attractive for high speed ICs

Traditional domino limited by sequencing overhead

- Latch delay
- Clock skew
- Inability to balance logic through time borrowing

Skew-tolerant domino eliminates overhead

- Use overlapping clocks and remove latches
- Overhead at static / domino interface motivates building critical loops entirely from domino

Domino design issues

- Four-phase skew-tolerant domino is a reasonable choice
- Timing types specify legal connections among static and dynamic gates
- Latchless pipelines can still be fully scanned for testability
- Simple clock generators

Now in widespread use among top design teams

- DEC Alpha ALUs, Intel “OTB Domino,” IBM “Delayed-Reset Domino,” Sun “Delayed Clocking,” *etc*.