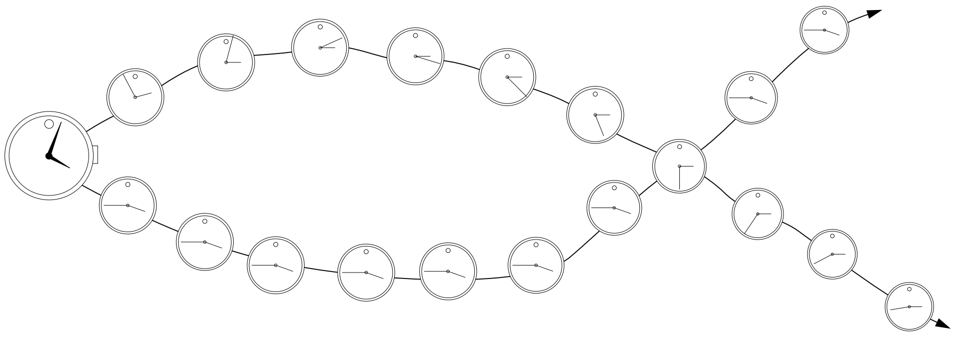
Keeping The Clock Pure

or alternately

Making The Impurities Digestible



"*Timing is everything*."

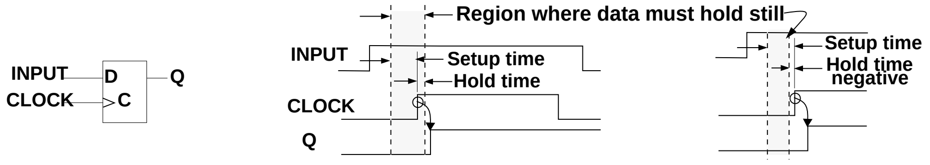
Review of Timing Properties of Flip-Flops

Setup Time and Hold Time

FIG. 4-2 .

Every flip-flop has time regions around the active clock edge in which the input should not change

If the input changes in these restricted regions, the output may be derived from either: the old input, the new input, or even half-way in between.



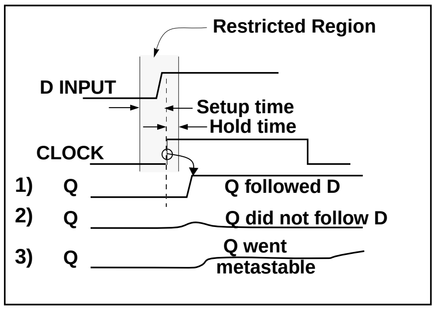
The *setup time* is the interval before the clock where the data must be held stable.
The *hold time* is the interval after the clock where the data must be held stable.

Most modern flip-flops have a zero or a negative hold time.
A negative hold time means the data can change slightly before the clock edge and still be properly captured.

Summary of the Restricted Region

The *restricted region:*
Time interval near the active clock edge where the D input signal should not change.

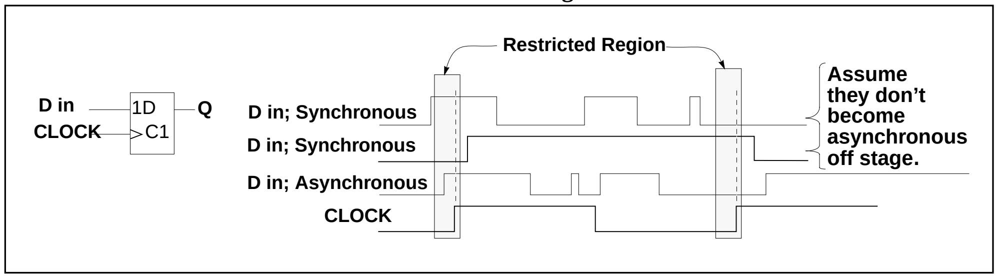
Otherwise the flip-flop's output, after the clock edge, may:
1) follow the change in D.
2) not follow D.
3) follow it halfway (go metastable).



Synchronous and Asynchronous Signals

A *synchronous* signal
One which is constrained so it cannot change in the restricted region.
An asynchronous signal
can and will change anywhere.

FIG. 4-3 **Three different D inputs.**
The upper two are synchronous;
they do not change in the restricted region.
The lower one is asynchronous;
it has a transition inside the restricted region.

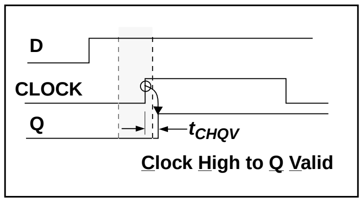


The Clock-to-Output Propagation Delay, t_{CHQV}

The time from the active clock edge until the Q output changes.

Another name is t_{CHQV}
(time from Clock going High to Q becoming Valid).

Any reasonable flip-flop will have $t_{CHQV} > t_{HOLD}$.
This is essential in shift-registers.

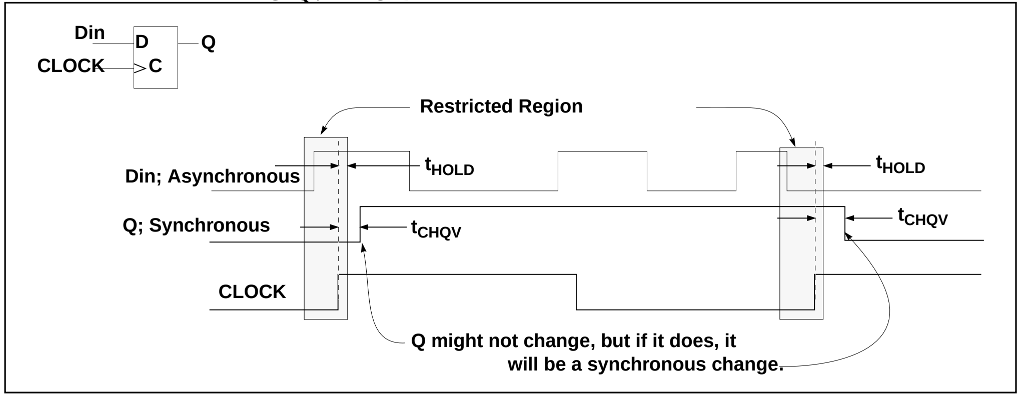


The Output Signal From a Clocked Flip-Flop is Always Synchronous

Any signal which passes through a flip-flop is synchronous.

The delay $t_{CHQV,}$ is enough to move Q changes out of the restricted region.
The *Q* signal in below is synchronous.
It results from passing signal *Din* through a D flip-flop.

FIG. 4-4 **The Q output is always synchronous, even if the input D signal is not.**
The internal propagation delay moves any change out of the restricted region.
This assumes $t_{CHQV} > t_{HOLD}$



Clock-to-Clock Logic Propagation Delays.

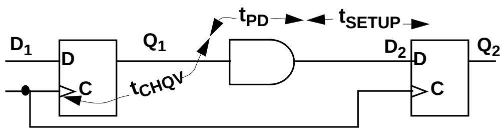
Maximum and Minimum Delays With a Perfect Clock

Maximum Logic Propagation Delays

Consider a synchronous circuit made of flip-flops with logic in between them.

FIG. 4-5 **One flip-flop feeding through logic into another flip-flop.**

The Q_1 signal takes t_{CHQV} to get out of the left flip-flop.
The propagation delay through the gate(s) is $t_{PD.}$
The D_2 signal must arrive at the right flip-flop at least t_{SETUP} before the second clock edge.



$t_{CLOCK} \geq t_{CHQV} + t_{PD} + t_{SETUP}$

The signal must get from the first flip-flop to the next, in one clock cycle, thus -

$t_{CLOCK} \geq t_{CHQV} + t_{PD} + t_{SETUP}$

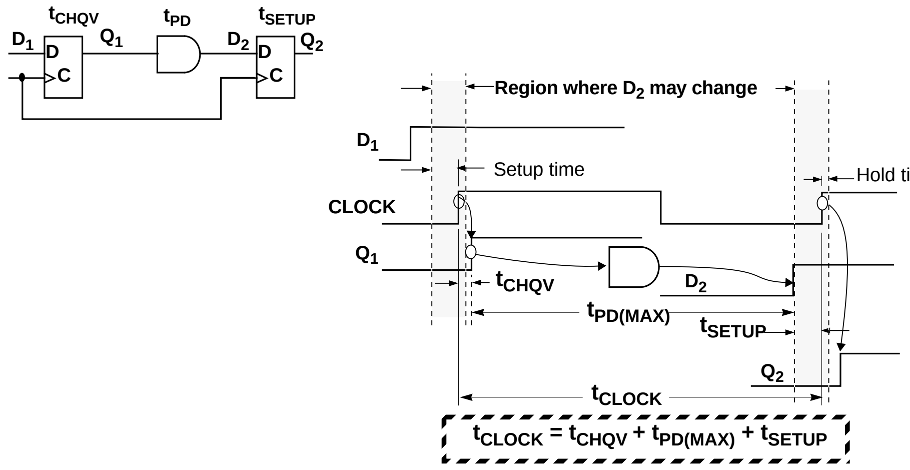
or

$\boxed{t_{PD} \leq t_{CLOCK} - t_{SETUP} - t_{CHQV}}$

FIG. 4-6 The maximum logic delay in a synchronous circuit

The Q_1 signal takes t_{CHQV} to get out of the left flip-flop.
The D_2 signal must arrive at the right flip-flop at least t_{SETUP} before the second clock edge.
The gate delay, $t_{PD,}$ could take up the rest of the clock cycle.

Call the longest allowable gate delay $t_{PD(MAX)}t$



$$t_{CLOCK} = t_{CHQV} + t_{PD(MAX)} + t_{SETUP}$$

Minimum Logic Propagation Delays.

One can have a valid minimum gate propagation delay.

This is when $t_{CHQV} < t_{HOLD}$.

Think of two flip-flops clocked on the same edge.
For a long *hold* time,
one flip-flop can flip
within the long hold time,
and send its new output to the next flip-flop
fast enough to flip it on the same clock edge.

To avoid double flips:

$t_{HOLD} \le t_{CHQV} + t_{PD}$

or

$t_{PD} \ge t_{HOLD} - t_{CHQV}$

The propagation delay of the gate(s)
must be above the minimum allowable,

$t_{PD(MIN)} = t_{HOLD} - t_{CHQV}$

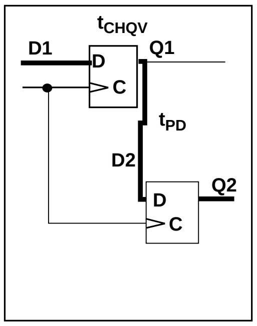
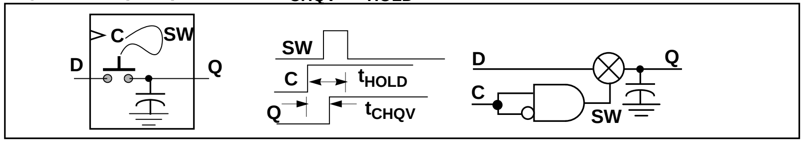


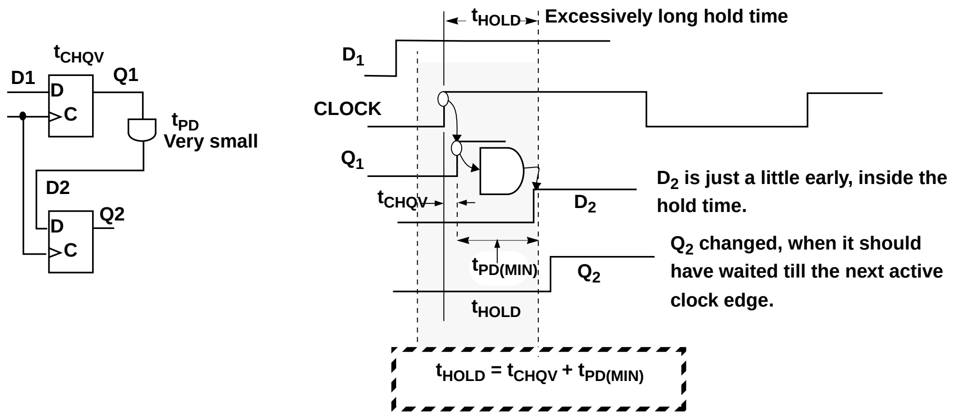
FIG. 4-7 A dynamic flip-flop in which $t_{CHQV} < t_{HOLD}$.



Minimum Propagation Delay Repeated; Different Pictures.

The minimum delay appears when the flip-flop has a *hold time* longer than t_{CHQV}.
Then D_1 can flip Q_1 of the upper flip-flop,
travel through the gate and reach the lower flip-flop
inside its hold time.

In that case the lower flip-flop may change too on the same clock edge!



tCHQV
D1
Q1
D
C
tPD
Very small
D2
Q2
D
C
tHOLD
Excessively long hold time
D1
CLOCK
Q1
tCHQV
D2
D2 is just a little early, inside the hold time.
tPD(MIN)
Q2
Q2 changed, when it should have waited till the next active clock edge.
tHOLD

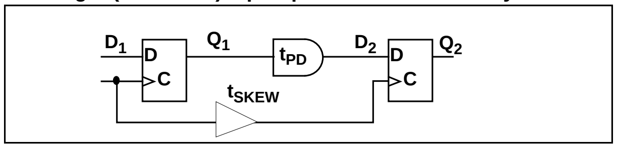
$$\boxed{t_{HOLD} = t_{CHQV} + t_{PD(MIN)}}$$

Minimum and Maximum Logic Delays With Clock Skew.

Clock Skew

Clock skew
is when the clock edge does not reach all the flip-flops at the same time.

Positive skew
Define skew as positive when
the data and clock are delayed in the same direction.
In most schematics
the right (or bottom) flip-flop will receive the delayed clock.



Maximum Logic Delays With Clock Skew

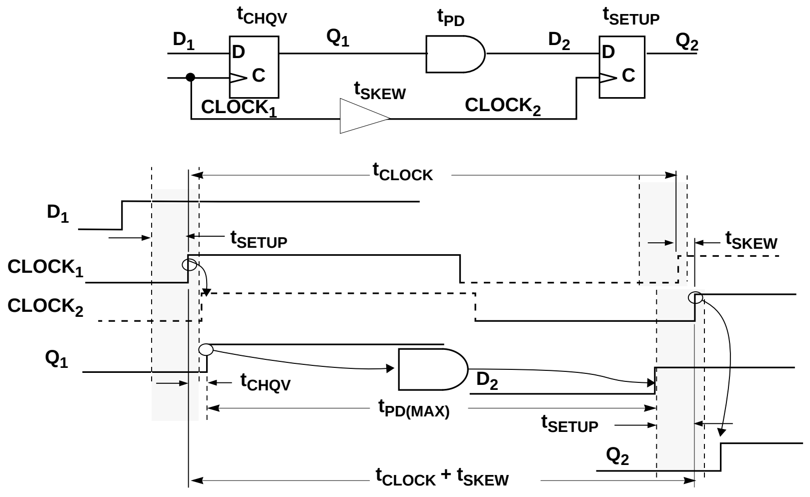
Positive skew increases the time available to get to the right-hand flip-flop.

From FIG. 4-8

$$\boxed{t_{CLOCK} + t_{SKEW} = t_{CHQV} + t_{PD(MAX)} + t_{SETUP}}$$

FIG. 4-8 **Positive Clock Skew Increases tPD(MAX)**

If there is a positive skew in the clock, there is more time to get to the right hand flip-flop, and $t_{PD(MAX)}$ is increased by the amount of the skew.



$$t_{CLOCK} + t_{SKEW} = t_{CHQV} + t_{PD(MAX)} + t_{SETUP}$$

Minimum Logic Delays With Clock Skew

See FIG. 4-9

The minimum logic delay $t_{PD(MIN)}$ is made worse (increased) by positive skew.

Modern flip-flops have $t_{HOLD} < t_{CHQV}$,
the minimum allowed prop delay is zero ($t_{PD(MIN)} = 0$).

With skew one may require $t_{PD(MIN)} > 0$.

But shift registers have $t_{PD} \cong 0$.
Thus with clock skew, shift registers may fail.

From FIG. 4-9, the limit on minimum delay is -

$$t_{SKEW} + t_{HOLD} = t_{CHQV} + t_{PD(MIN)}$$

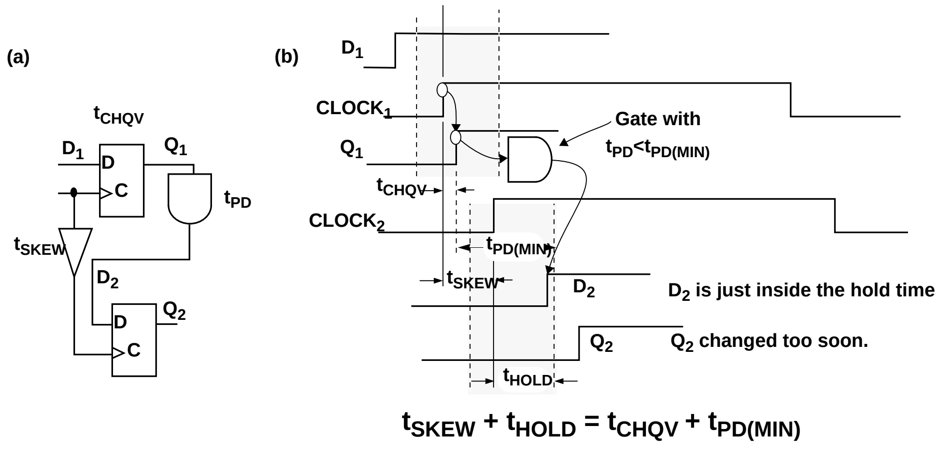
Thus

$$\boxed{t_{PD} \geq t_{HOLD} + t_{SKEW} - t_{CHQV}}$$

FIG. 4-9 **Minimum propagation delay limit with skew.**

(a) With positive clock skew, the clock to the lower flip-flop is delayed.
The chance of a D_1 change going through both flip-flops in one cycle increases.
The t_{SKEW} acts like an increase in the hold time of the lower flip-flop.

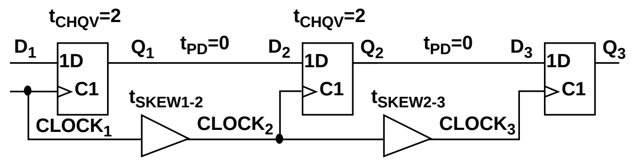
(b) Waveforms, when skew makes the actual $t_{PD} < t_{PD(MIN)}$.



$$t_{SKEW} + t_{HOLD} = t_{CHQV} + t_{PD(MIN)}$$

PROB 4.1 **Maximum Skew For A Shift Register**

Find the maximum delay in the clock buffers for the shift register shown.



t_{PD} (Q to D) = 0 ns
t_{CHQV} = 2 ns max
t_{HOLD} = -1 ns min

Solution

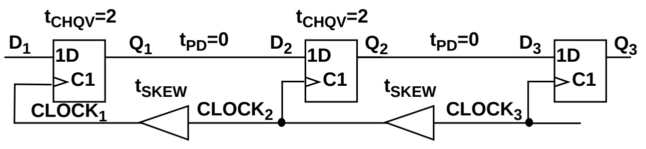
For Q_1 to D_2	For Q_2 to D_3
$t_{PD} \geq t_{HOLD} + t_{SKEW} - t_{CHQV}$	$t_{PD} \geq t_{HOLD} + t_{SKEW} - t_{CHQV}$
$0 \geq -1 + t_{SKEW1\text{-}2} - 2$	$0 \geq -1 + t_{SKEW2\text{-}3} - 2$
$t_{SKEW1\text{-}2} \leq +3$	$t_{SKEW2\text{-}3} \leq +3$

$$t_{SKEW1\text{-}3} = t_{SKEW1\text{-}2} + t_{SKEW2\text{-}3} \leq +6$$

Thus the delay from CLOCK1 to CLOCK2 may be up to 3 ns.
The delay from CLOCK1 to CLOCK3 may be up to 6 ns

PROB 4.2 **Maximum Negative Skew For A Shift Register**

Find the maximum delay in the clock buffers for the shift-register shown below. The clock delays now go opposite the data, i.e. t_{SKEW} is negative.



t_{PD} (Q to D) = 0 ns
t_{CHQV} = 2 ns max
t_{HOLD} = -1 ns min
t_{SKEW} is negative that is
$t_{SKEW} = -|t_{SKEW}|$

This is equivalent to PROB 4.1 with t_{SKEW} negative.
Thus

$-|t_{SKEW1\text{-}2}| \le +3$ and $-|t_{SKEW2\text{-}3}| \le +3$

Any negative skews is less than 3 ns,

Thus any negative skew is acceptable.

However watch out if the negative skew approaches a clock period.

In shift registers, route the clock against the shift.

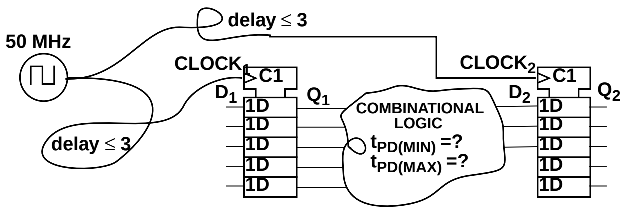
PROB 4.3 **Maximum and Minimum Delay With Bounded Skew**

Two registers of D flip-flops have a clock skew which is between -3 and 3 ns. Since we do not know the sign, we must always assume the worst case,
i.e. positive when calculating $t_{PD(MIN)}$
and negative when calculating $t_{PD(MAX)}$.

Q_1 is the collective name for any or all outputs of the right-hand register.
D_2 is the same for the inputs of the right-hand register.

Find $t_{PD\ (MIN)}$ and $t_{PD\ (MAX)}$.

$t_{SKEW} = t_{EDGE\text{-}CLOCK2} - t_{EDGE\text{-}CLOCK2}$



$|t_{SKEW}| \le 3$ ns
t_{CLOCK} = 20 ns
t_{HOLD} = 0 ns min
t_{CHQV} = 2 ns max
t_{SETUP} = 4 ns max

Solution:

For $t_{PD(MIN)}$

$t_{PD} \ge t_{HOLD} + t_{SKEW} - t_{CHQV}$
$t_{PD(MIN)} = 0 + 3 - 2$
$= 1$ ns

For $t_{PD(MAX)}$

$t_{PD} \le t_{CLOCK} + t_{SKEW} - t_{SETUP} - t_{CHQV}$
$t_{PD(MAX)} = 20 + (-3) - 4 - 2$
$= 11$ ns

Summary of Simple Propagation Delay Bounds

Define positive skew as-
Clock delay in the same direction as data-flow delay, i.e.

$t_{SKEW} = t_{DESTINATION-CLOCK-EDGE} - t_{SOURCE-CLOCK-EDGE}$

then

$$\boxed{t_{PD(MAX)} = t_{CLOCK} + t_{SKEW} - t_{CHQV} - t_{SETUP}} \qquad \text{(EQ 1)}$$

$$\boxed{t_{PD(MIN)} = t_{SKEW} + t_{HOLD} - t_{CHQV}} \qquad \text{(EQ 2)}$$

Positive skew:

- **allows longer logic delays**
- **forces the minimum delay to be longer.**

Negative skew:

- **allows a shorter minimum logic delay**
- **forces the maximum logic delays to be shorter.**

<u>Rule of thumb for maximum clock skew</u>

For a modern clock distribution system.
They do not know, or do not have time to examine the logic details.

Assume there may be very fast paths between flip-flops($t_{PD(MIN)} = 0$).
Assume modern flip-flops $t_{HOLD} \le 0$.
Approximate bound on skew, from (EQ 2) is -

$$\boxed{t_{SKEW} \le t_{CHQV}} \qquad \text{(EQ 3)}$$

Clock Skew Related to Signal Delay

Most circuits do not have a simple structure where one flip-flop or register is the source and another is the destination.

Here we relate clock skew in circuits where the data travels in more complex paths.

<u>Finding Minimum and Maximum Propagation Delays Given Clock Skew</u>

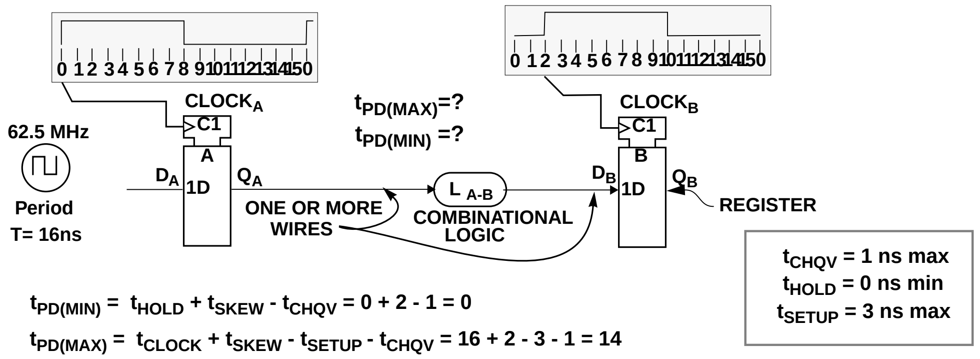
- **A register here is a set of edge-triggered simultaneously clocked D flip-flops.**
- **Registers are connected to other registers by combinational logic.**
- **Let the clock skew between registers be known.**
- **We will find the fast bound $t_{PD(MIN)}$ and the slow bound $t_{PD(MAX)}$ on the logic paths between each register pair.**

FIG. 4-10 **Figure to establish notation for skew calculations**

The registers shown have a common edge-triggered clock input.
The register's bottom block represents one or more flip-flops.
A single line entering or leaving a register, represents 1 or more wires.
The oval L_{A-B} represents combinational logic with a source register A and destination register B.

The clock delay for each register is shown by a waveform above the register.
Thus if register-A has the clock edge applied at 0 ns as shown,
then register-B will be clocked at +2 ns.

From the waveforms $t_{SKEW\ A-B}$ = +2 ns.



$t_{PD(MIN)} = t_{HOLD} + t_{SKEW} - t_{CHQV} = 0 + 2 - 1 = 0$

$t_{PD(MAX)} = t_{CLOCK} + t_{SKEW} - t_{SETUP} - t_{CHQV} = 16 + 2 - 3 - 1 = 14$

t_{CHQV} = 1 ns max
t_{HOLD} = 0 ns min
t_{SETUP} = 3 ns max

FIG. 4-11 **More complex interconnections with clock skew.**

For no clock skew, the propagation delay bounds are
$t_{PD(MIN)} = t_{HOLD} - t_{CHQV} = 0 - 1 = -1$ **(Effectively 0; logic cannot have a negative t_{PD})**
$t_{PD(MAX)} = t_{CLOCK} - t_{SETUP} - t_{CHQV} = 16 - 3 - 1 = 12$
To add skew, calculate a table of the skew for each logic block, and add it to these base delays.

t_{CHQV} = 1 ns max
t_{HOLD} = 0 ns min
t_{SETUP} = 3 ns max

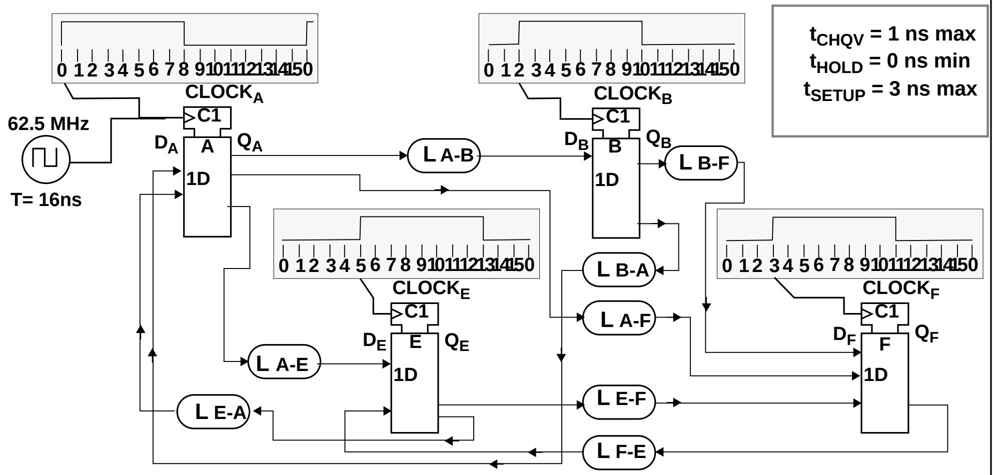


FIG. 4-12 **Clock skew, and propagation delay limits assuming no skew.**

Table of t_{SKEW} between:
source register (listed on the left), and
destination register (listed on the top).

Skew = $t_{destination} - t_{source}$

Source Reg;	Destination Reg; Clock edge delay			
delay	A; 0 ns	B; 2 ns	E; 5 ns	F; 3 ns
A; 0	0	2	5	3
B; 2	-2	0	3	1
E; 5	-5	-3	0	-2
F; 3	-3	-1	2	0

Minimum/maximum t_{PD} limits for logic connected between registers in FIG. 4-11. No skew.

Source Reg;	Destination Reg; Clock edge delay			
delay	A; 0 ns	B; 2 ns	E; 5 ns	F; 3 ns
A; 0	-1 12	-1 12	-1 12	-1 12
B; 2	-1 12	-1 12	-1 12	-1 12
E; 5	-1 12	-1 12	-1 12	-1 12
F; 3	-1 12	-1 12	-1 12	-1 12

No skew

$$t_{PD(MIN)} = t_{HOLD} - t_{CHQV} = -1$$
$$t_{PD(MAX)} = t_{CLOCK} - t_{SETUP} - t_{CHQV} = 12$$

FIG. 4-13 **Propagation delay limits with skew.**

$t_{PD(MIN)}$ / $t_{PD(MIN)}$ limits with skews.

This table is the box-by-box sum of the tables in FIG. 4-12.

Source Reg;	Destination Reg; Clock edge delay			
delay	A; 0 ns	B; 2 ns	E; 5 ns	F; 3 ns
A; 0	-1 12	1 14	4 17	2 14
B; 2	-3 10	-1 12	2 15	0 13
E; 5	-6 7	-4 9	-1 12	-3 10
F; 3	-4 9	-2 11	1 13	-1 12

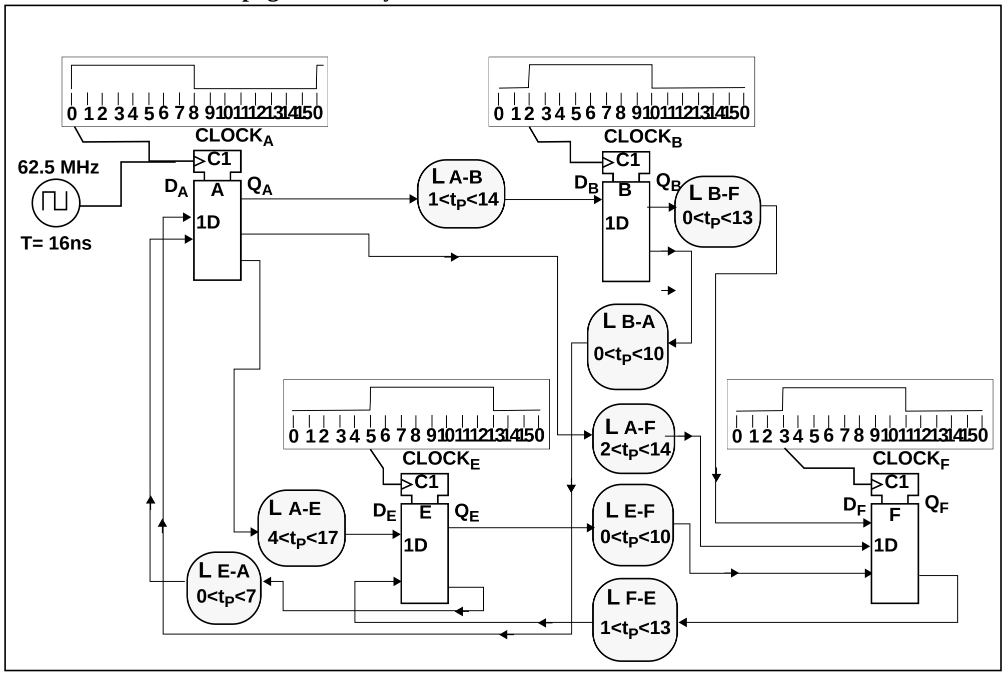
Minimum/maximum prop delays for connections actually made in FIG. 4-11.

Example: the oval shows $L_{B\text{-}A}$ has
$t_{PD(MIN)}$ = -3 ns
$t_{PD(MIN)}$ = 10 ns.

Source Reg;	Destination Reg; Clock edge delay			
delay	A; 0 ns	B; 2 ns	E; 5 ns	F; 3 ns
A; 0		1 14	4 17	2 14
B; 2	-3 10			0 13
E; 5	-6 7			-3 10
F; 3			1 13	

0 $L_{B\text{-}A}$
$-3 < t_{PD} < 10$

FIG. 4-14 Min/Max Propagation Delays Shown on Schematic



Example: Using a Ripple Counter in a Supposedly Synchronous Design

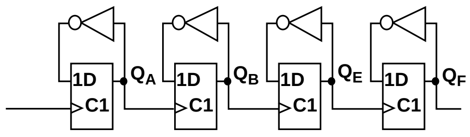
Ripple counters are a very simple binary counter.

Unfortunately they have impure clocking.
The clock does not run directly to each flip-flop's clock input.
Thus they are not strictly synchronous.

One must check for timing and hazards in ways not necessary for a truly synchronous design.

FIG. 4-15 A 4-bit Binary Ripple Counter.

The clocking is not pure.
One master clock does NOT go directly to all flip-flop clock inputs.



Consider a circuit using the ripple counter output, decoded by a blob of combinational logic, feeding a register.

Calculate $t_{PD(MAX)}$ and $t_{PD(MIN)}$ for the logic.

Each flip-flop output in the counter is delayed by t_{CHQV} from the previous stage, so t_{CHQV} is the skew between the counter flip-flops.

Also check the counter outputs for hazards which might trigger the next clock input.
Fortunately signals directly from flip-flops do not glitch.

FIG. 4-16 **A circuit using a ripple counter.**

The ripple counter flip-flops have a large clock skew.

Calculate the bounds on t_{PD} in the combinational logic.

With no clock skew

$t_{PD(MAX)} = t_{CLOCK} - t_{SETUP} - t_{CHQV} = 16 - 2 - 3 = 11$

$t_{PD(MIN)} = t_{HOLD} - t_{CHQV} = 0 - 1 = -1$

$t_{CHQV} = 1$ ns max

$t_{SETUP} = 2$ ns max

$t_{HOLD} = 0$ ns min

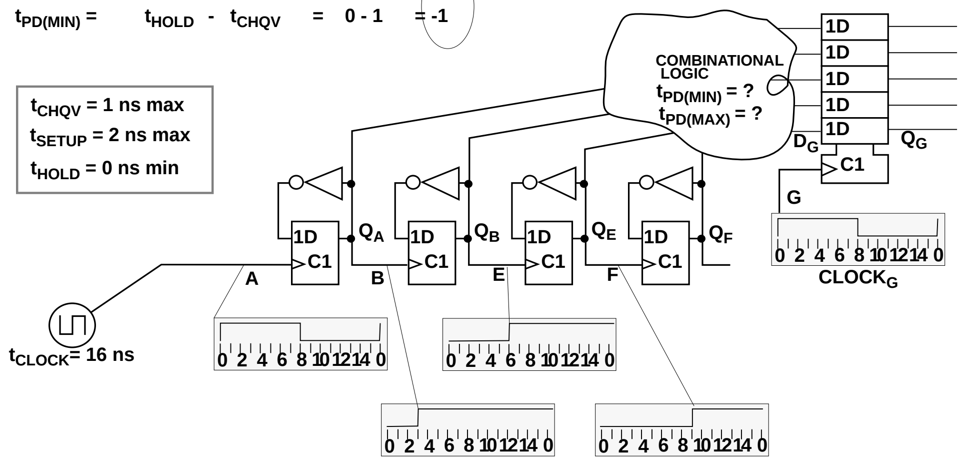


FIG. 4-17 **Clock skew, and propagation delay limits for the circuit.**

Table of t_{SKEW} between any register as source (listed on the left), and any register as destination (listed on the top).

Source Reg; delay	Destination Reg; Clock edge delay				
	A; 0 ns	B; 3 ns	E; 6 ns	F; 9 ns	G: 0 ns
A; 0	0	3	6	9	0
B; 3	-3	0	3	6	-3
E; 6	-6	-3	0	3	-6
F; 9	-9	-6	-3	0	-9
G: 0	0	3	6	9	0

Minimum/maximum prop. delay limits.
Limits for t_{SKEW}=0 are [-1, 11] as on the diagonal.
The off diagonal limits are the sum of [-1, 11] and the skew from the table on the right.

Source Reg; delay	Destination Reg; Clock edge delay				
	A; 0 ns	B; 2 ns	E; 5 ns	F; 3 ns	G: 0 ns
A; 0	-1 11	2 14	5 17	8 20	-1 11
B; 2	-4 8	-1 11	2 14	5 17	-4 8
E; 5	-7 5	-4 8	-1 11	2 10	-7 5
F; 3	-10 2	-7 5	-4 8	-1 11	-10 2
G: 0	-1 11	2 14	5 17	8 20	-1 11

The four delays applicable to the logic blob are shaded.
The negative (minimum) delays are not needed here.
Note: flip-flop output Q_F to D_G in register G, has a maximum of 2 ns extra delay.

Note clock G is the same as A. They could have been combined in the table.

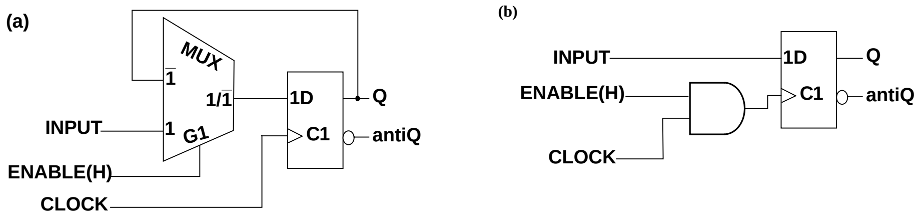
Gating the Clock

Many designer succumb to the temptation to gate the clock.
It is a simple way to disable a D flip-flop or a group of flip-flops.
This may save area or power.

This is a design method has three problems:
1) It will add to the clock skew.
2) It can cause a false clock edge.
3) Full-scan testing will not test it

FIG. 4-18

(a) The "approved" method of enabling/disabling a D flip-flop. There is no clock skew and no false clock edges.
(b) Gating the clock is the "high risk" way to enable/disable the flip-flop. However the gate saves power by not clocking nonflipping flip-flops.



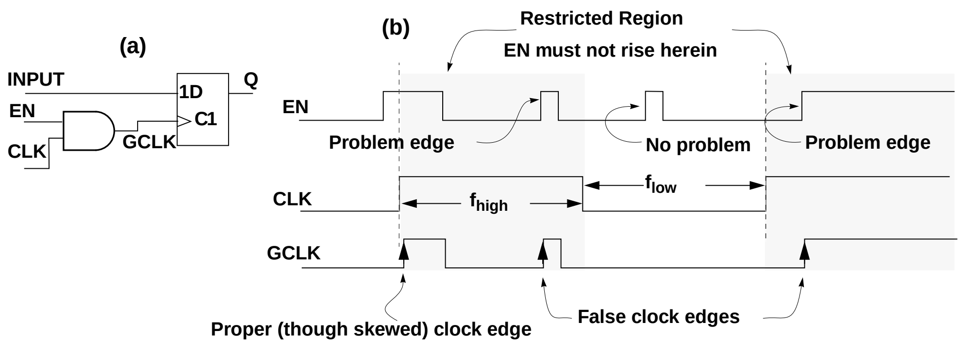
Clock Skew From Gating the Clock

Clock skew was just covered.
One can compensate for clock skew with extra work.

False Clock Edges.

FIG. 4-19 shows how false clock edges are generated.
It also shows how changes in EN must be restricted to avoid false clock edges.

FIG. 4-19 False clock edges caused by the EN signal rising while the clock is high.
One has a restricted region during the time the clock is high (ϕ_{high}) where the EN signal must not rise.



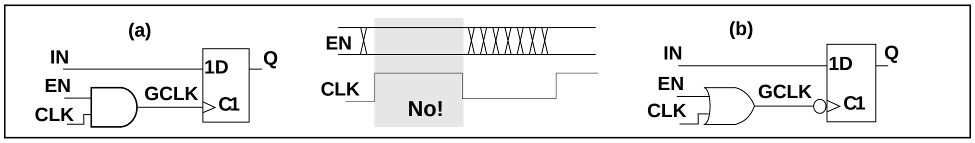
The four cases of clock gating

To avoid false clock edges, The clock EN signal must be restricted.

There are 2 groups of 2 cases each.

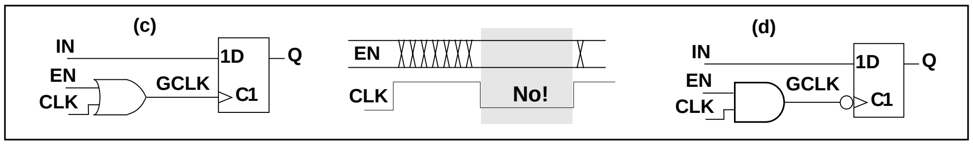
1) EN must not change in the first half cycle

Cases (a) and (b) allow EN changes in the last half of the clock cycle.



2) EN must not change in the second half cycle

Cases (c) and (d) allow EN changes in the first half of the clock cycle.



Or one could say

(a) and (d) allow changes only during ϕ_{low}

(b) and (c) allow changes only during ϕ_{high}

FIG. 4-20 False clock edges illustrating why the first half-clock-cycle is restricted

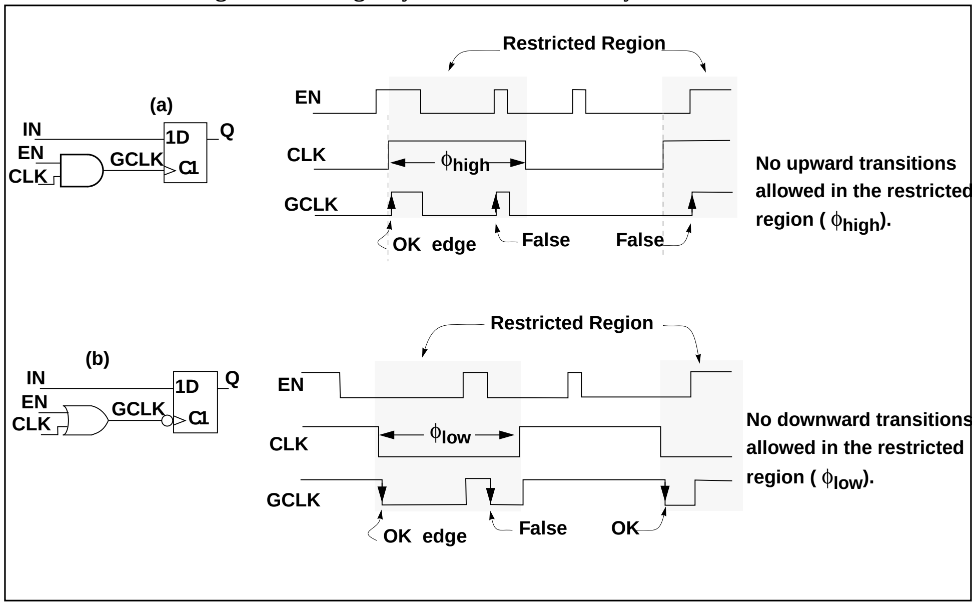
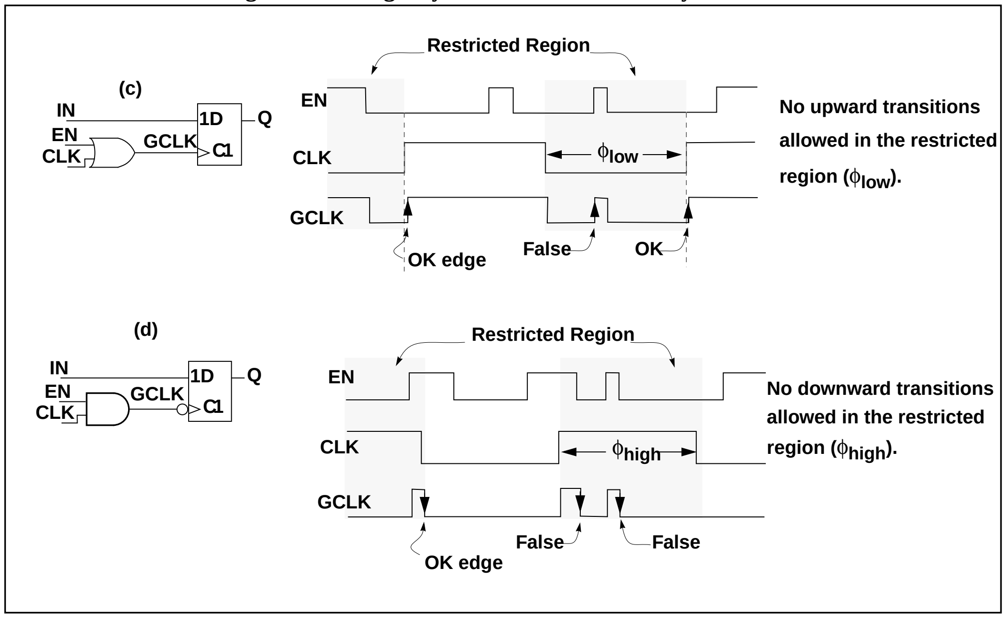


FIG. 4-21 **False clock edges illustrating why the second half-clock-cycle is restricted**



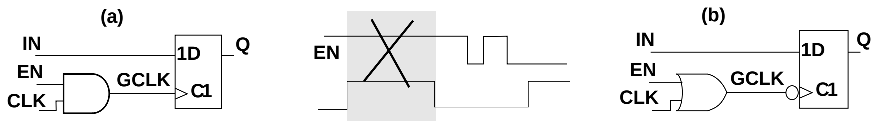
Clock Gating Summary

1) EN must not change in the first half cycle

EN changes in the last half of the clock cycle.

This allows more time to generate the EN, but
It has both an upper and lower bound on its delay.
It must be glitch free in the first half cycle *before it settles down*

This is difficult to design

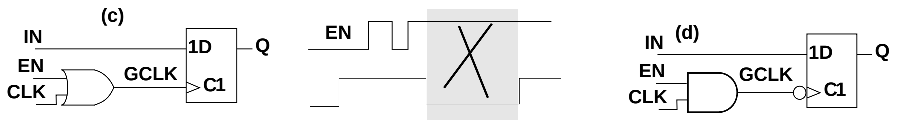


2) EN must not change in the second half cycle

EN changes in the first half of the clock cycle.

The EN signal must be generated quickly within 1/2 cycle
It must be glitch free in the last half cycle but there it has settled down.

This is simple to design except for speed requirement.



Safe Clock-Gating Using a Latch. (safe except for skew!)

Suppose the clock-gating signal EN-RAW has glitches.

EN-RAW is latched and applied to an AND gate.
The AND gate suppresses glitches in the 2st half clock cycle (method (a) above).

- **Glitches in the 1st half of EN-RAW are stopped because the latch is in store mode.**
- **Glitches in the 2nd half of EN-RAW are stopped by the AND gate (clock is low).**

This method is good for shutting down a subcircuit for several cycles.
For example shutting off the floating point unit in a microcomputer.

FIG. 4-22 **A clock gating method with no false clock edges.**

Q1 and Q2 are enabled for clock cycles when EN-RAW is high.
The latch blocks glitches when the clock is high.
The AND blocks glitches when the clock is low.

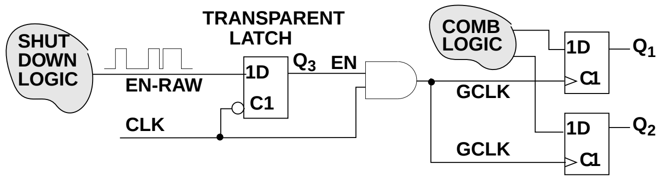
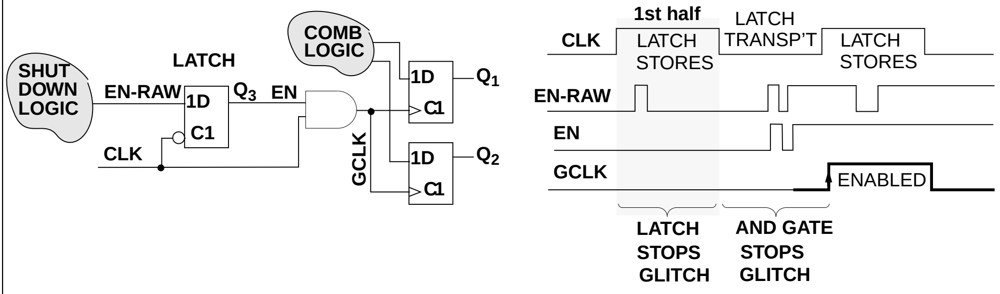


FIG. 4-23 **Waveforms for a gated clock with no false edges.**

Q1 and Q2 are enabled for clock cycles when EN-RAW is high.
The latch ensures the EN will not change in the restricted region for AND type gating.



When gating the clock to save power:

- **One normally gates many flip-flops at once.**
  For one flip-flop, the power for the extra latch may be more than the saving.
- **One normally shuts off the clock for many cycles at a time.**

The clock skew is minimized if an AND is placed in every clock line.
The full-scan test people will make it hard to do this.

Clock Dividers

Why Two Frequencies?

The Pentium, the DEC Alpha, and other modern microprocessors run internally at a clock frequency which is too high for the board level circuitry.

Where a system has both slow and fast logic, considerable circuitry and power can be saved by using a slower clock for part of the logic.

Basic Methods

- Clock all flip-flops at high-speed and enable the flip-flops at a lower speed.
  Is the safest (easiest) method.
  Will not give much power saving.
- Gate the high-speed clock with a slower signal.
  Will give medium power reduction.
  The lower frequency will use less power in the flip-flops,
  but charging and discharging the clock line at high speed will waste power.
  Gated clock designs are subject to false edges and skew.
- Divide the clock
  Divider can supply different frequencies to different flip-flops.
  There must not be skew between the main and divided clocks.
  Many divider/counters give glitches which are poisonous on clock lines.

Binary Counters As Clock Dividers

Many synchronous (not ripple) binary counters acts as a fairly good clock dividers.

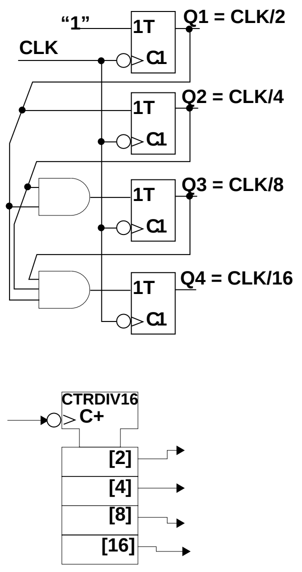
The output bits form a natural division chain.
Q1 = divide-by-2,
Q2 = divide-by-4,
--- -------- -- --
Qn = divide-by 2^n.

Skew
The Flip-flop outputs are the counter outputs
so all counter outputs change a flip-flop propagation-delay (t_{CHQV}) after clock.
If the flip-flops are identical, at the same temperature, and as physically close,
the skew between the divided clocks *should be* small.

The original clock (CLK) has a larger skew with respect to the divided clocks.
CLK rises (falls) t_{CHQV} before the divided clock edges.

One should only use CLK for clocking the divider chain.
Otherwise, resynchronize it (to be discussed).

FIG. 4-24 Falling-edge clock-divider



A binary counter suitable for a clock divider.

Toggle flip-flops make a simpler counter than D-flip-flops.

This is a falling-edge-triggered up counter.
which makes flip-flops flip together on falling edges.

The divided clocks are suitable for falling-edge-triggered flip-flops.

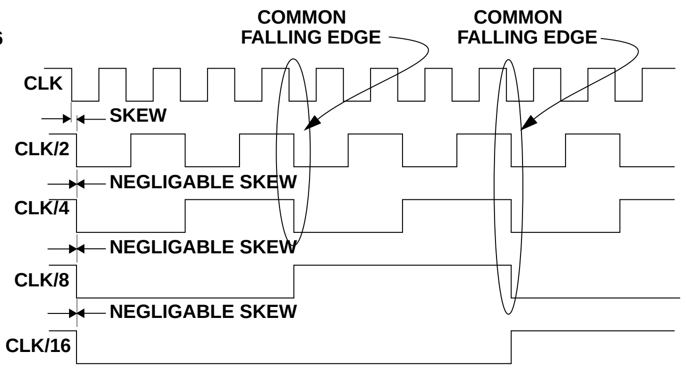
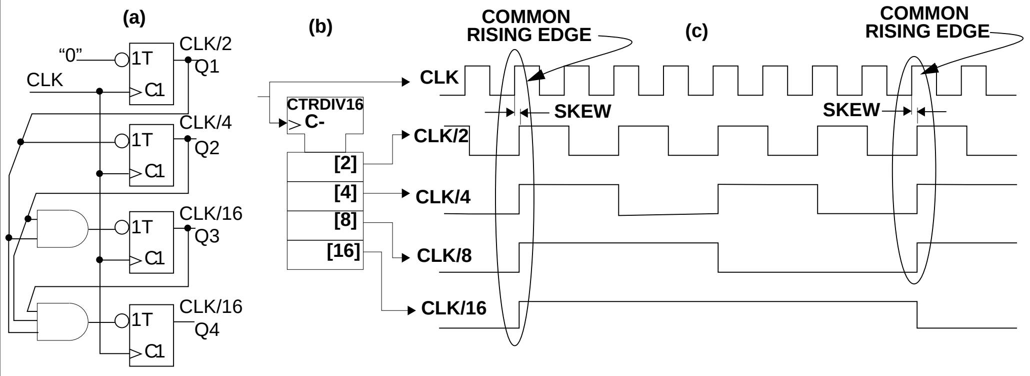


FIG. 4-25 A clock-divider with common rising-edges

(a) A binary down-counter suitable for a clock divider.

(b) The IEEE symbol for a down counter.
The blocks ▭ represent individual flip-flops and associated gates.
These blocks have a common clock which is shown entering the common control block ⊓ at the top.
The "-" after the clock input shows it counts down. An up counter would have a "+".

(c) The waveforms showing that down-counter flip-flops flip together rising edges.
The divided clocks are suitable for rising-edge-triggered flip-flops.



Resynchronization

Making all clocks tick together

Necessary for circuits clocked by both divided clocks and the main clock.

Widely distributed divided-clock signals may need local adjustments so their active edges all change together.

When divided clock signals are used over a large physical area.
It may be easier to resynchronize at each locality,
than to distribute low-skew divided clocks over the large area.

Resynchronizing latches should be physically close together for low skew between them.

FIG. 4-26 Examples of where divider resyncronization might be used.
(a) Using one or more divided clocks. and the original clock.
(b) Using one main clock distributed with special care to avoid skew,
and divided clocks which can have skew because they are resynchronized locally.

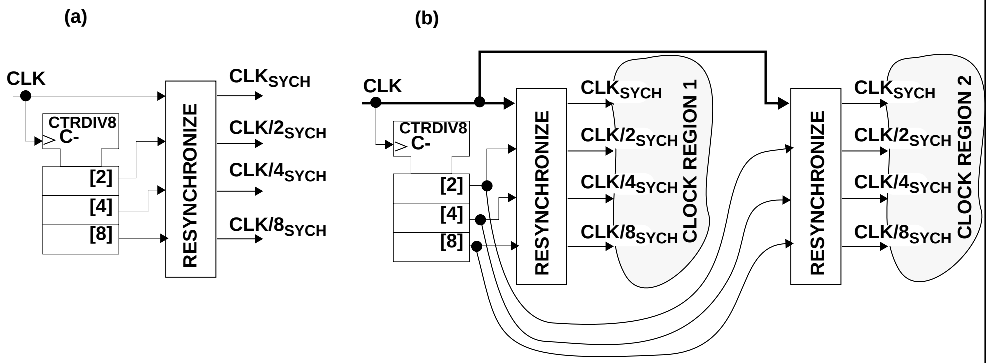
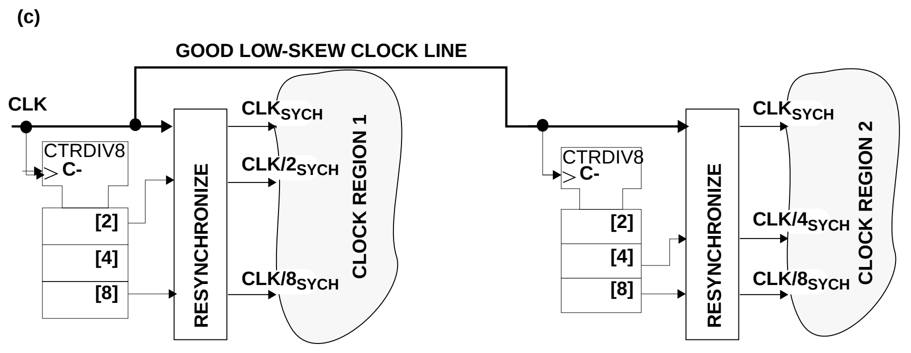
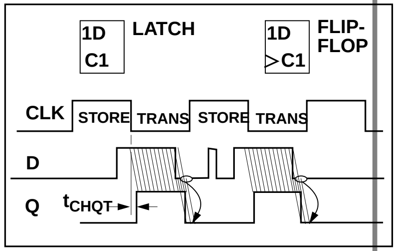


FIG. 4-27 **Examples of where divider resyncronization might be used**

**((c) Distributing one low-skew main clock,
generating divided clocks as needed in different localities,
and resynchronizing to a reduce skew between regions.**



Construction of a resynchronizer



The resynchronizer uses transparent latches, rather than edge-triggered flip-flops.

A transparent latch:
- acts like a piece of wire when the clock is low,
- stores the last passed Q value when the clock goes high.

Reason for latches:
Latches are simpler than flip-flops.
Flip-flops cannot synchronize the highest frequency signal, CLK.

All latches must have the same delay t_{CLQT},
(time from clock low to Q transparent)

The resynchronizer input CLK/n signals:
Except for the latch's setup or hold times,
Input signals may rise/fall or glitch anywhere in the store-mode half clock-cycle.

To resynchronize CLK:
Use an inverted *advanced clock* which rises more than a setup-time before the latch goes transparent.
This delay is shown as one inverter. It may take three.

The actual delay must be carefully done.
The high duration of CLK_{SYNCH} is always under a half cycle.
The shortening of this high pulse is linearly related the advance in ADV-CLK.

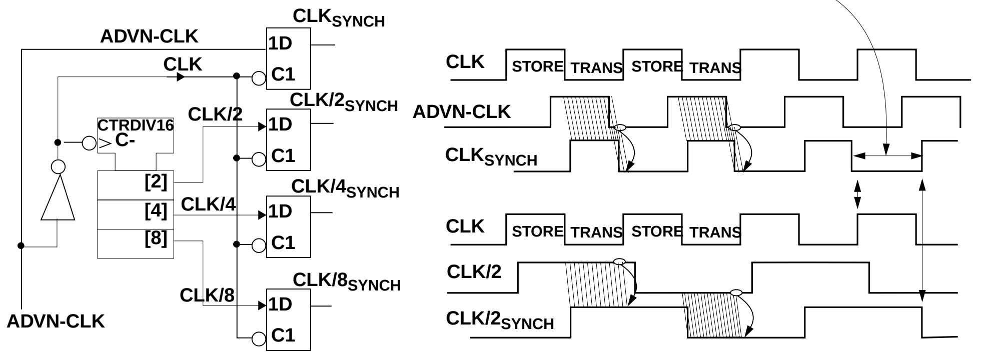
FIG. 4-28 Resynchronizer

Resynchronization positions the rising edges of all clocks very close together.
Resynchronization using transparent latches.

The closely-spaced lines show the latches D-to-Q transfer when the latch is transparent.
The arrow shows the value stored when the latch enters stored mode.

To synchronize the original CLK, one needs an inverted clock (ADVN-CLK) which leads CLK.

Note CLK_{SYNCH} is not symmetric



Example:

<u>Using a ripple counter as a divider; Then resynchronizing the outputs.</u>

FIG. 4-30 shows a ripple counter used as a clock divider.
The ripple counter outputs are delayed 3, 6, and 9 ns.

When the latches input changes inside the “store” state of the latch, the synchronizing latch will wait until CLK makes it transparent before its output will change.
The synchronized signals emerge when all the latches go transparent together.

The ripple counter puts a 9 ns delay in CLK/8.
This is over half a clock cycle and is too much for the synchronizer.
CLK/8 enters its latch after the latch has gone transparent.
Thus $CLK/8_{SYNCH}$ is delayed 1ns and is skewed.

Synchronization narrows the CLK_{SYNC} pulse by the 4 ns delay of the three inverters.
The one inverter delay would be only be 1.3 ns.
This delay is to short to counteract the D-to-Q delay in the synchronizer[1].

1. This spec says that a high output will not come out of the latch until a D =1 signal has been applied for t_{DHQH} = 5 ns. This would be 3.7 ns after the active CLK edge in stead of 3 ns, and CLK_{SYNC} would rise at 11.7 ns, not 11ns.

FIG. 4-29 A clock divider circuit using a ripple counter.

The flip-flops in the ripple counter have a large clock skew.
The resynchronizer will try synchronize the rising edges.
However the delay in CLK/8 is over half of CLK and is too much to be synchronized.
The shaded blocks show how the clock is delayed at various points.

$t_{CHQV} = t_{CLQT}$ = 3 ns max for both flip-flops and latches

t_{DHQH} = 5 ns max

$t_{PD\text{-}INVERTER}$ = 1.3 ns max

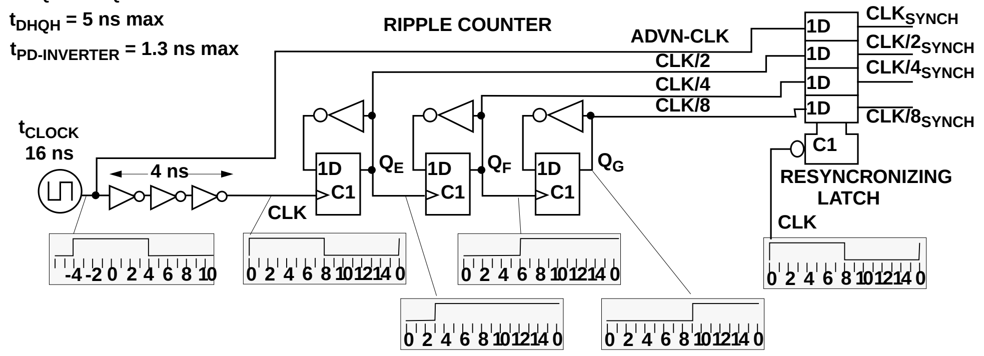
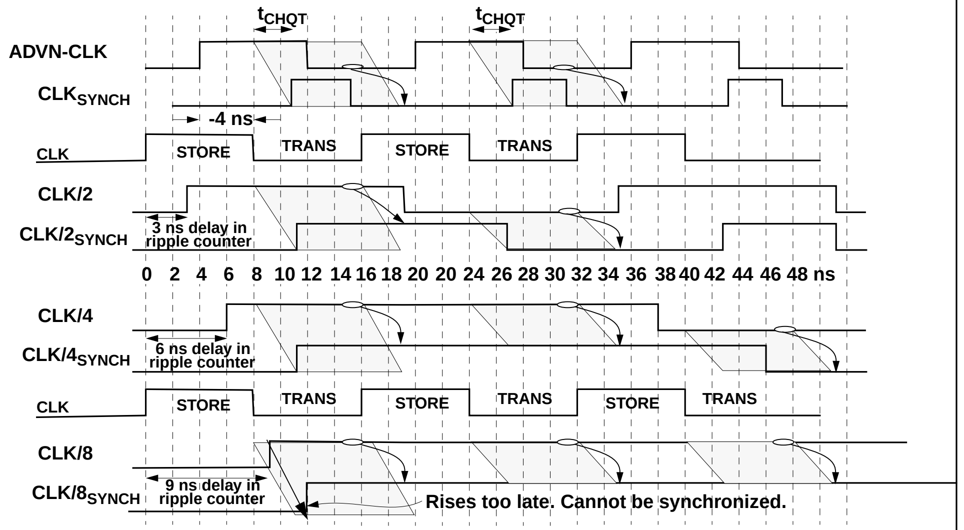


FIG. 4-30 The resynchronizer waveforms to scale

The latches are transparent in the shaded areas.
The slope of the grey area represents the latch prop. delay t_{CHQT}. Here t_{CHQT} = 3 ns.
CLK/8 rises after the latch goes transparent. The ➘ shows t_{CHQT}, for the rising edge.
It shows CLK/8 will have a 1ns skew.



Data Resynchronization Between Skewed Clocks

Isochronic Regions And Data Signals Between Them

Consider a circuit so large that skew cannot be controlled over the whole circuit.

Within certain regions of the circuit the skew is small and causes no problem.
We call such regions *isochronic regions*.

The problem to be considered here is synchronizing the *data signals* that pass between isochronic regions.

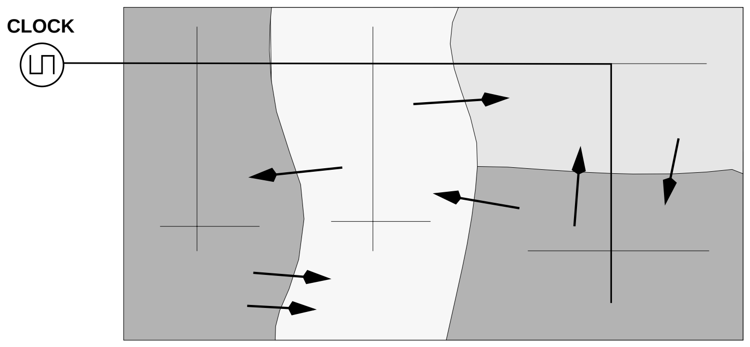
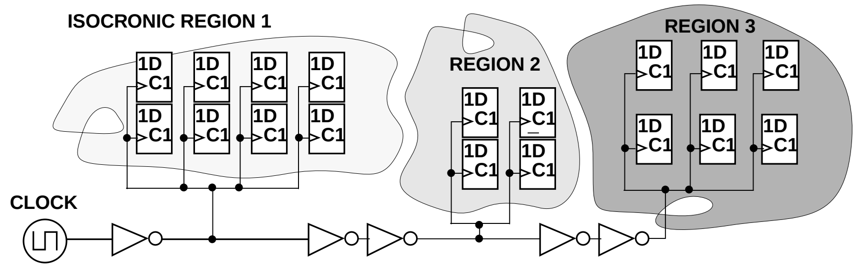


FIG. 4-31 **One master clock with isochronic regions**

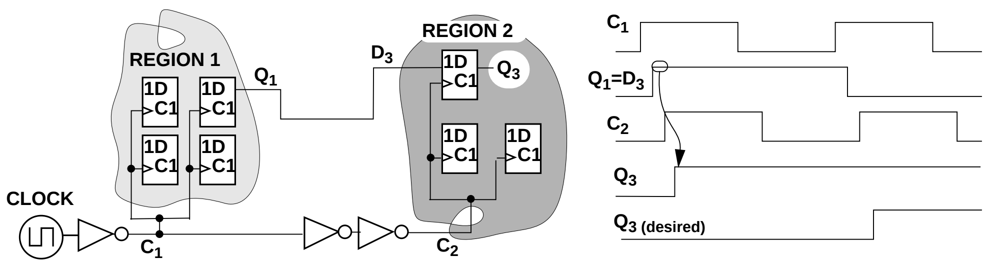
Here the different delays are caused by a different number of clock buffers.
No Engineering graduate from Carleton would design this!
More likely, skew would be caused by:
- grossly different lengths of clock lead,
- a different type of lead to different areas;
perhaps coax to some areas and circuit board track to others,
- or differences in the buffer loadings; perhaps a different number of flip-flops on each buffer.



Cycle Skipping

FIG. 4-32 Cycle skipping in data flowing from an early clock region to a late clock region.

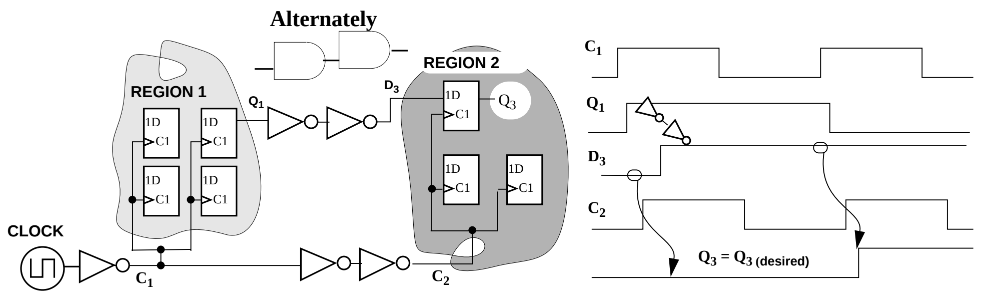
Clock C_1 in Isochronic region 1, leads the clock in region 2.
A Q_1 change clocked by C_1, will get through to D_3 before C_2 rises.
Then Q_3 will change on the delayed rising-edge of C_2.
Clearly the designer planned to delay Q_3 as shown by Q_3 (desired).



Adding Delay To Avoid Cycle Skipping

FIG. 4-33 Delaying the data to compensate for clock skew.

Here the clock skew is compensated by a data skew.
The data is delayed by at least as much as the clock.
Then Q_3 will be properly delayed and there will be no cycle skipped.



Alternately choose a path that has several gates anyway.

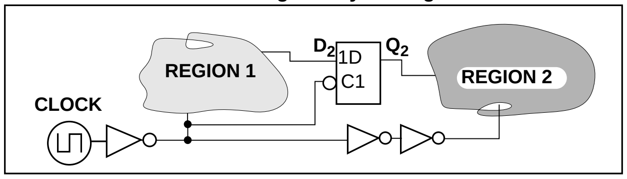
Recall t_{PD} was the propagation delay for gates between Q_1 and D_3
Make sure $t_{PD} > t_{SKEW}$.

Resynchronizing The Data Lines To Accommodate Clock Skew

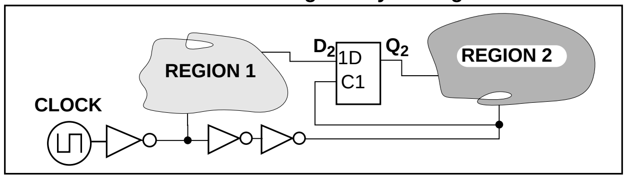
There are two circuits, for resynchronizing data to compensate for clock skew.

They both insert a latch in each data line going between isochronic regions.

In the first circuit the latch is gated by the region which send the data.



In the second circuit the latch is gated by the region which receives the data.



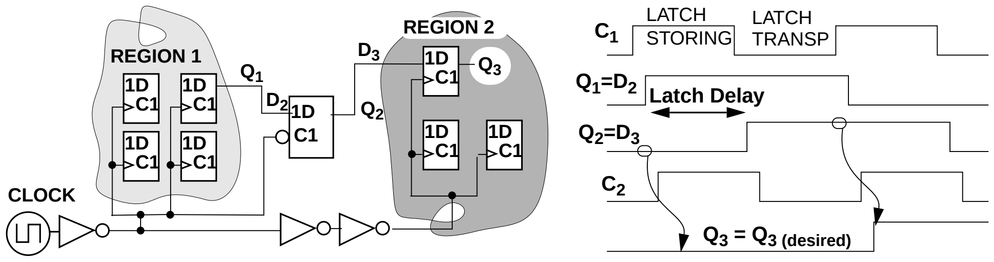
Latches can be dynamic.
They are recharged every cycle.

Method 1, for resynchronizing data

FIG. 4-34 **Resynchronizing data with a D latch, to compensate for clock skew.**

Here the delay is put in by a D latch.
Such a latch delays the signal half a clock period,
that is until the latch t goes transparent.

This allows a more automatic design than inserting gate-delays as in FIG. 4-33.
Half a clock cycle should be more than adequate delay for deskewing.



Method 1: Analysis of data resynchronization using latches in detail

FIG. 4-35 **Resynchronizing data signals.**

Timing limitations of using latches in data lines to compensate for clock skew.
The clock feed is changed so the skew might be positive or negative.
Also gates have been inserted in the latch leads.
See the timing diagram in FIG. 4-36.

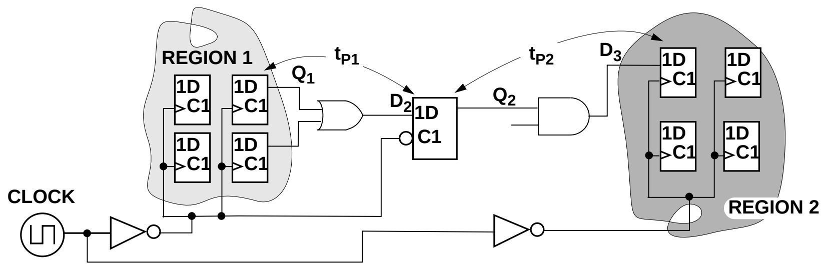
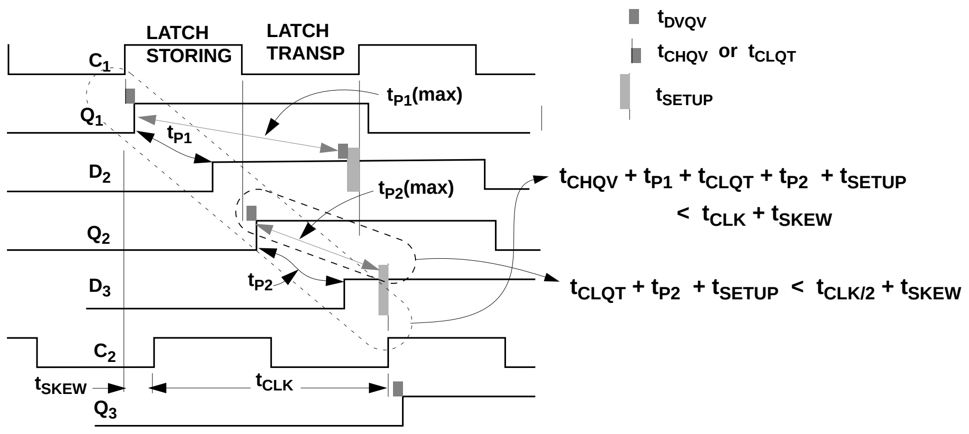


FIG. 4-36 **Timing diagram for data resyncronization**

The timing diagram when there are delays in the circuitry around the latch.
It is used to derive bounds for the amount of skew the circuit can resynchronize.
The dark squares represent the flip-flop/latch input-to-output propagation delays.
The lighter gray rectangles represent setup times.



Timing details for skew correction with latches in the data paths

Consider the data path Q_1-> D_2 -> Q_2 -> D_3 in the dashed oval in FIG. 4-36.
The shaded squares represent either a setup or a clock-to-output delay.
The delays along this data path, starting at the rising edge of C_1, are-

$$t_{CHQV} + t_{P1} + t_{CLQT} + t_{P2} + t_{SETUP}$$

This must happen before the second rising edge of C_2 which is at -

$$t_{CLK} + t_{SKEW}$$

after C_1 rises. Thus[1]

$$t_{CHQV} + t_{P1} + t_{CLQT} + t_{P2} + t_{SETUP} < t_{CLK} + t_{SKEW} \qquad \text{(EQ 4)}$$

The first gate(s) delay, t_{P1}, may extend into the transparent half clock cycle.
Thus for $t_{P2} = 0$, and small internal latch/flip-flop delays,
$\boxed{t_{P1} \text{ can approach } t_{CLK}}$.

However, Q_2 can never change before the latch goes transparent, so t_{P2} is limited by

$$t_{CLQT} + t_{P2} + t_{SETUP} < t_{CLK/2} + t_{SKEW} \qquad \text{(EQ 5)}$$

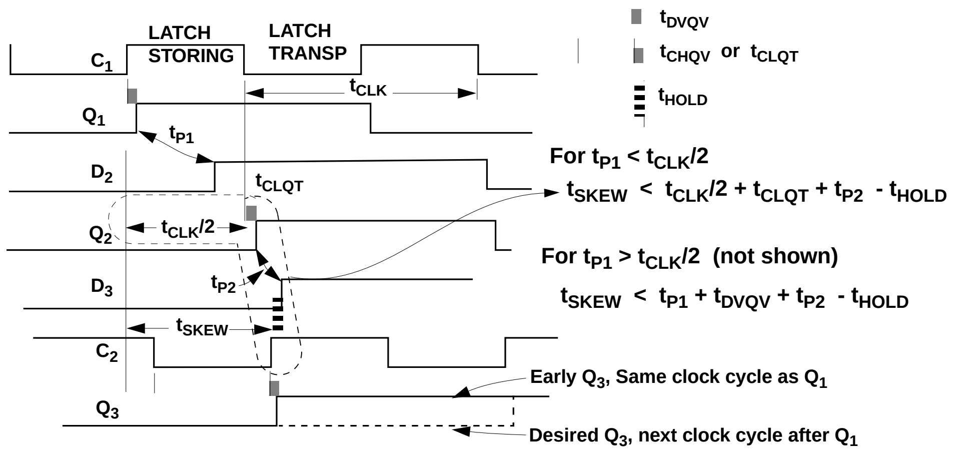
Thus for small internal latch/flip-flop delays and $t_{P1} < t_{CLK}/2$,
$\boxed{t_{P2} \text{ can approach } t_{CLK}/2.}$

This circuit can accommodate nearly 180° of negative skew (C_1 lagging).
and over 180° of positive skew (C_2 leading).

1. For t_{P1} extended into the transparent half of the clock cycle, replace t_{CLQT} (Clock Low to Q Transparent) with t_{DVQV} (D input data Valid to Q output Valid) in (EQ 4). They are usually about the same.

FIG. 4-37 Timing diagram showing upper positive skew limit.

The diagram is for positive skew over 180°.
The upper bound for cycle skipping is shown.
Note that D_3 can rise up to a hold time after clock C_2 and still be captured as Q_3.

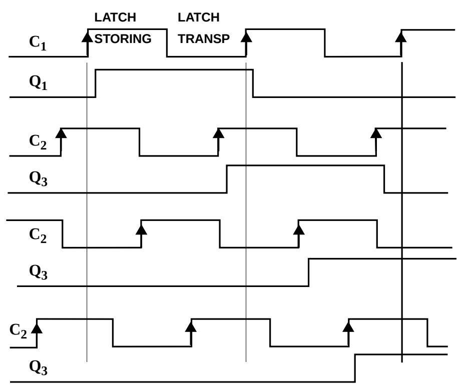

LATCH STORING
LATCH TRANSP
C1
tCLK
Q1
tP1
D2
tCLQT
Q2
tCLK/2
D3
tP2
tSKEW
C2
Q3
tDVQV
tCHQV or tCLQT
tHOLD
For tP1 < tCLK/2
tSKEW < tCLK/2 + tCLQT + tP2 - tHOLD
For tP1 > tCLK/2 (not shown)
tSKEW < tP1 + tDVQV + tP2 - tHOLD
Early Q3, Same clock cycle as Q1
Desired Q3, next clock cycle after Q1

The range of skew for which correction is possible

FIG. 4-38 **Timing diagram showing the meaning of positive and negative skew.**

Positive skew of over 180° is a special case.
In this case a positive skew of 360°-θ is not the same as a skew of θ.
The difference is in which cycle Q_3 is intended to change. See the Q_3 waveforms below.

The restrictions on the skew are stated beside the waveforms.



C_1 clock taken as fixed

Negative skew under 180°

$|t_{SKEW}| < t_{CLK/2} - t_{CLQT} - t_{SETUP} - t_{P2}$

Positive skew under 180°
(Same as for over 180°)

$t_{SKEW} < t_{CLK}/2 + t_{CLQT} + t_{P2} - t_{HOLD}$

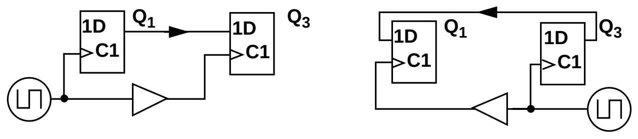
Positive skew over 180°

$t_{SKEW} < t_{CLK}/2 + t_{CLQT} + t_{P2} - t_{HOLD}$

FIG. 4-39 **When clock skew is a problem**

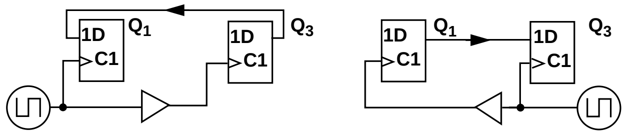
This shows that resynchronization is needed only if the clock and data delay are in the same direction.
If they are in opposite directions, it reduces the propagation delay Q_i to D_{i+1} .
If this is a problem, compensate by slowing the clock period by the amount of the skew.

+Skew; Clock Delay In Same Direction as Data Delay
Hold-time violation likely



t_{PD}(min) increased.
Need synchronization or
more delay in data path.

-Skew; Clock Delay In Opposite Direction from Data Delay
Setup-time violation likely



t_{PD}(max) decreased.
No synchronization needed
provided data-path delay is
within the reduced bound.

Summary: First circuit for resynchronizing on the data lines

Positive skew

- **If the data delay and the clock delay are in the same direction resyncronization is needed.**
- **It can be omitted if one can guarantee the data delay is larger than the clock delay.**

Negative skew

- **If the data delay and the clock delay are in opposite directions no resyncronization is needed.**
- **Setup time violations may become critical**

For latch resynchronized data

- **A negatively gated latch, ○1D C1 clocked from the data input circuit clock, can resynchronize for skews of nearly $t_{CLK}/2$.**
- **This latch is useful when the magnitude and sign of the skew are not well controlled.**
- **The logic delay at the latch input t_{P1}, can approach t_{CLK}. See (EQ 4)**
- **The logic delay at the latch output t_{P2}, can approach $t_{CLK}/2$, provided t_{P1} is correspondingly reduced. See (EQ 5)**
- **The sum of $t_{P1} + t_{P2}$ can approach t_{CLK}. See (EQ 4).**

Resynchronizing The Data Lines To Accommodate Clock Skew; Circuit 2

The second resynchronizing circuit is much like the first.
The timing will be analyzed in Figures etc.

FIG. 4-40 **Resynchronizing data with a D latch, to compensate for clock skew.**

Here the delay is put in by a D latch.
The latch delays the signal half a clock period until the latch goes transparent.
This allows a more automatic design than inserting gate-delays as in FIG. 4-33.
Half a clock cycle should be more than adequate delay for deskewing.

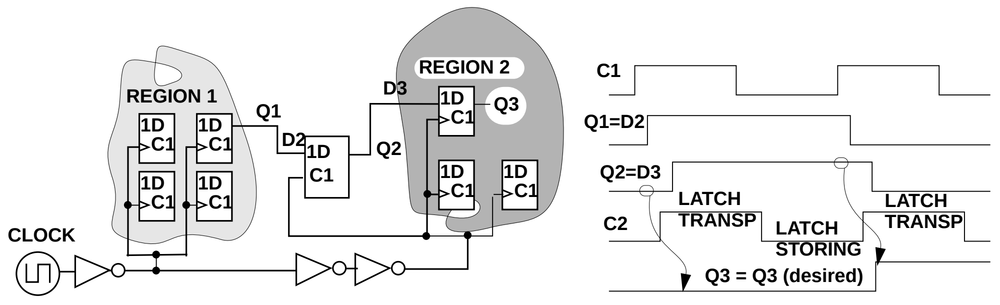


FIG. 4-41 **Resynchronizing data signals with a latch; shown in more detail.**

The schematic fro analysis of timing limitations of method 2.
The clock feed is changed so the skew might be positive or negative.
Also gates have been inserted in the latch leads.
The timing diagram for this circuit is given in FIG. 4-36.

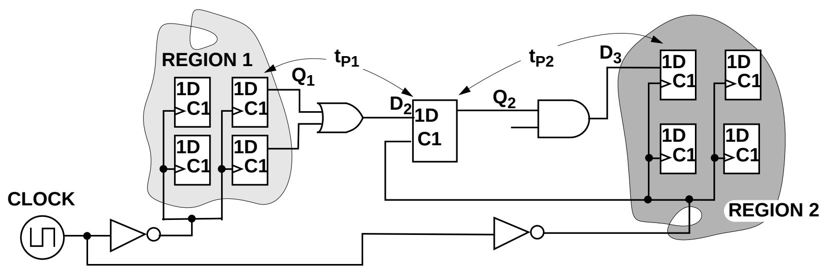


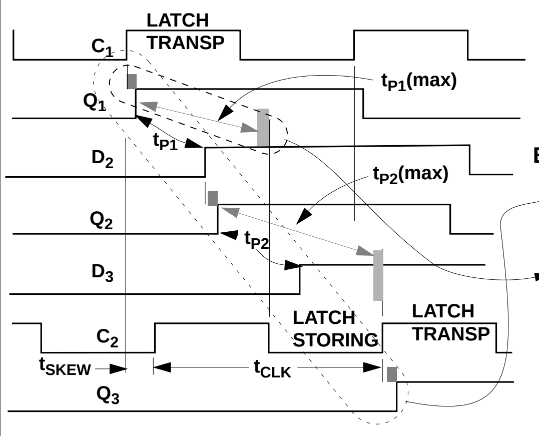
FIG. 4-42 **Timing diagram for the data resyncronization; 2nd circuit**

The timing diagram of the delays around the latch.
It is used to derive bounds for the propagation delays and skew.

▮ t_{CHQV} or t_{DVQV}

$t_{DVQV} = t_{DValidQValid}$ is the time to propagate through the transparent latch

▮ t_{SETUP}



Bounding equations for delays and skew

$$t_{CHQV} + t_{P1} + t_{DVQV} + t_{P2} + t_{SETUP} < t_{CLK} + t_{SKEW}$$

$$t_{CHQV} + t_{P1} + t_{DVQV} < t_{CLK}/2 + t_{SKEW}$$

For long t_{P1} near $t_{P1}(max)$, the timing may be limited by the latch t_{SETUP}, not t_{DVQV}.

$$t_{CHQV} + t_{P1} + t_{SETUP} + t_{P2} + t_{SETUP} < t_{CLK} + t_{SKEW}$$

$$t_{CHQV} + t_{P1} + t_{SETUP} < t_{CLK}/2 + t_{SKEW}$$

Circuit 2; timing details for skew correction with latches in the data paths

The data path Q_1-> D_2 -> Q_2 -> D_3 is inside the dashed oval in FIG. 4-42.
The delays along this data path, starting at the rising edge of C_1, are-

$$t_{CHQV} + t_{P1} + t_{DVQV} + t_{P2} + t_{SETUP}$$

This must finish before the second rising edge of C_2 which happens at -

$$t_{CLK} + t_{SKEW}$$

Thus

$$t_{CHQV} + t_{P1} + t_{DVQV} + t_{P2} + t_{SETUP} < t_{CLK} + t_{SKEW} \quad \text{(EQ 6)}$$

The signal propagation through the first gate(s) t_{P1} must leave time for D_2 to change and propagate to Q_3 before the latch stops being transparent.
Thus the logic propagation delay t_{P1} is also limited by[1]

$$t_{CLQT} + t_{P1} + t_{DVQV} < t_{CLK/2} + t_{SKEW} \quad \text{(EQ 7)}$$

For small t_{CHQV} and t_{SETUP},

- t_{P1} can approach $t_{CLK}/2$,
- t_{P2} can approach t_{CLK}.
- the sum t_{P1} + t_{P2} must be under t_{CLK}.

1. To be strictly correct, one should replace t_{DVQV} with Max(t_{DVQV}, t_{SETUP}) in both (EQ 6) and (EQ 7). This applies when t_{P1} is near t_{P1}(max) and the signal must be captured as the latch enters store mode.

Summary of Method 2

This circuit can accommodate nearly 180° of negative skew (C_2 lagging).
It can accommodate over 180° of positive skew (C_2 leading).

For negative skew the output Q_3 changes slightly under t_{CLK} after Q_1 changed, in the transparent part of C_1.
The Q_3 timing is just as it would be with no latch.

Bounds on the skew are found by rearranging (EQ 7) and drawing another diagram.

For negative skew -

$$|t_{SKEW}| < t_{CLQ}/2 - t_{CHQV} - t_{P1} - t_{DVQV} - t_{P2} - t_{SETUP} \quad \text{(EQ 8)}$$

For positive skew -

$$t_{SKEW} < t_{CLQT} + t_{P1} + t_{DVQV} + t_{CLK}/2 + t_{P2} - t_{HOLD} \quad \text{(EQ 9)}$$

Choice of Methods

Circuit 2 has similar properties to circuit 1.
The main difference is the allowable propagation delays t_{P1} and t_{P2}.

For long $t_{P1} >$ $t_{CLK}/2$ use circuit1.

For long $t_{P2} >$ $t_{CLK}/2$ use circuit2.

In both the sum t_{P1} + t_{P2} must be under t_{CLK}..