12 February 2020 وَمَا أُوتِيتُم مِّنَ الْعِلْمِ إِلَّا قَلِيلًا 18 جمادى الثانية 1441

Digital IC Design

Lecture 18
Sequential Circuit Design (2): Sequencing Static Circuits

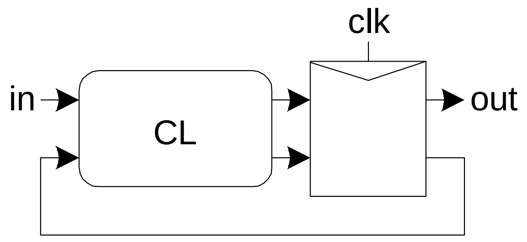
Dr. Hesham A. Omran

Integrated Circuits Laboratory (ICL)
Electronics and Communications Eng. Dept.
Faculty of Engineering
Ain Shams University

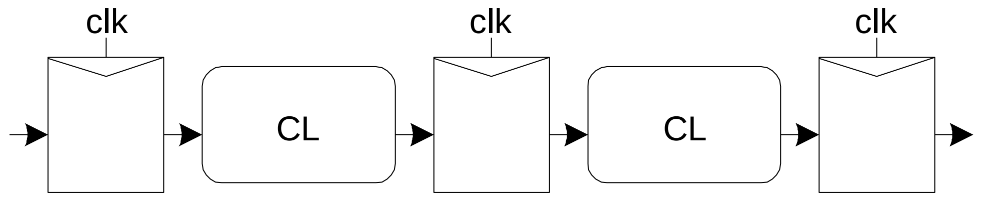
This lecture is mainly based on "CMOS VLSI Design", 4th edition, by N. Weste and D. Harris and its accompanying lecture notes

Combinational vs. Sequential Logic

- Combinational logic
  - Output depends on current inputs
- Sequential logic
  - Output depends on current and previous inputs
  - Uses flip-flops or latches (memory/sequencing elements)
  - Separates previous, current, and next states (tokens)
  - Ex: FSM, pipeline



Finite State Machine



Pipeline

Sequencing

- If tokens moved through pipeline at constant speed, no sequencing elements would be necessary
- Ex: fiber-optic cable
  - Light pulses (tokens) are sent down cable
  - Next pulse sent before first reaches end of cable
  - No need for hardware to separate pulses
  - But *dispersion* sets min time between pulses
- This is called *wave pipelining* in circuits
- In most circuits, dispersion is high
  - Delay fast tokens so they don’t catch slow ones.

Sequencing Cont.

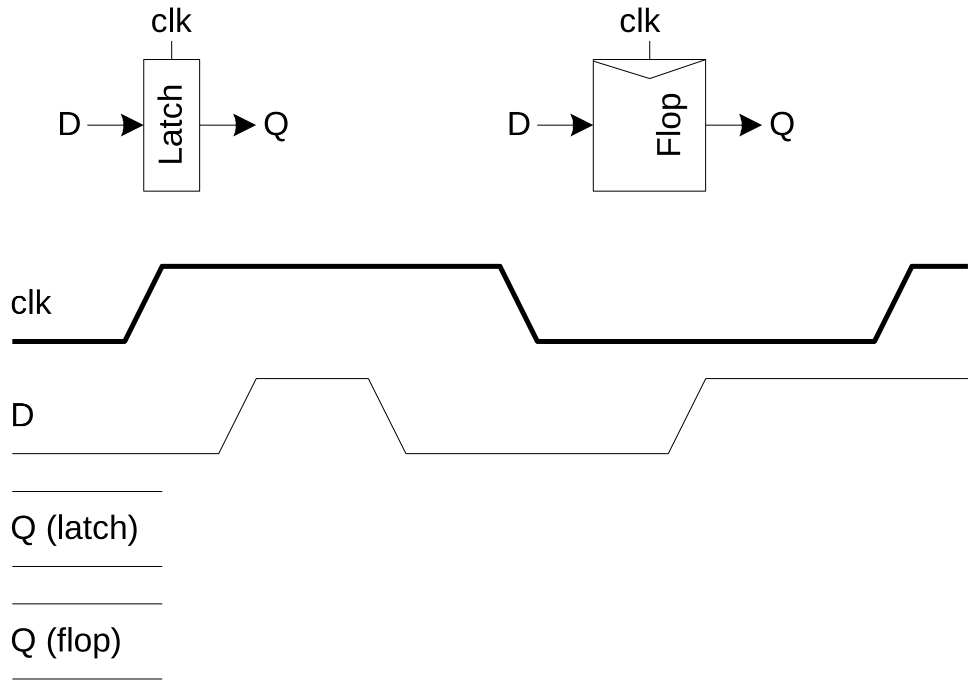
- Use flip-flops to delay fast tokens so they move through exactly one stage each cycle.
- Inevitably adds some delay to the slow tokens
- Makes circuit slower than just the logic delay
  - Called sequencing overhead
- Some people call this clocking overhead
  - But it applies to asynchronous circuits too
  - Inevitable side effect of maintaining sequence

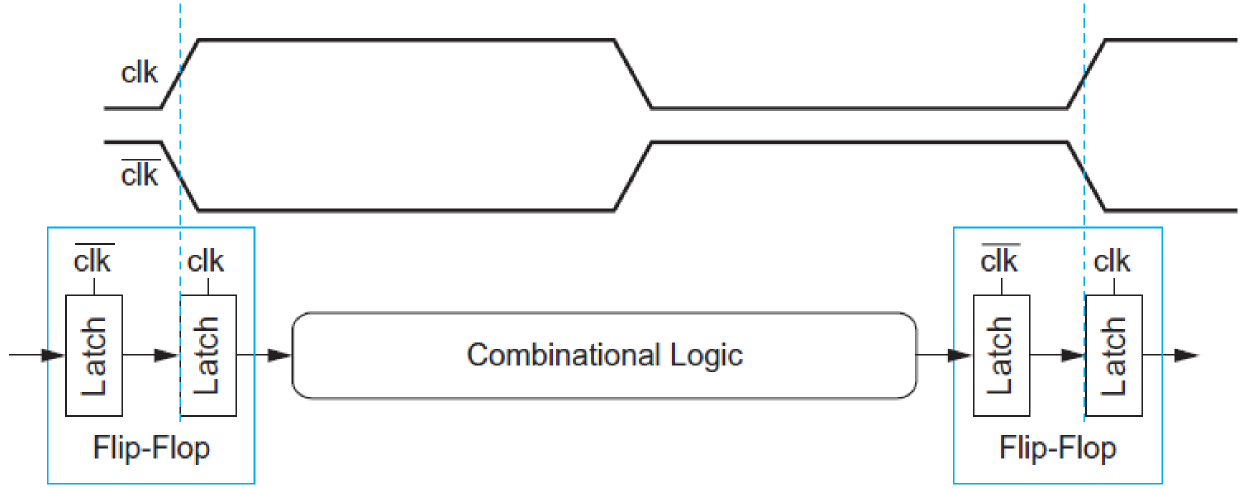
Static vs. Dynamic Logic

- Combinational circuits:
  - Static circuits: no clock input
    - e.g., complementary CMOS, pseudo NMOS, and PTL
  - Dynamic circuits: requires clock input
    - e.g., domino logic
- Sequential circuits:
  - Static storage: feedback used to retain output indefinitely
  - Dynamic storage: temporary charge on a capacitor
- Static/dynamic circuits can be sequenced using static/dynamic storage
- **We will focus on sequencing static circuits**

Sequencing Elements

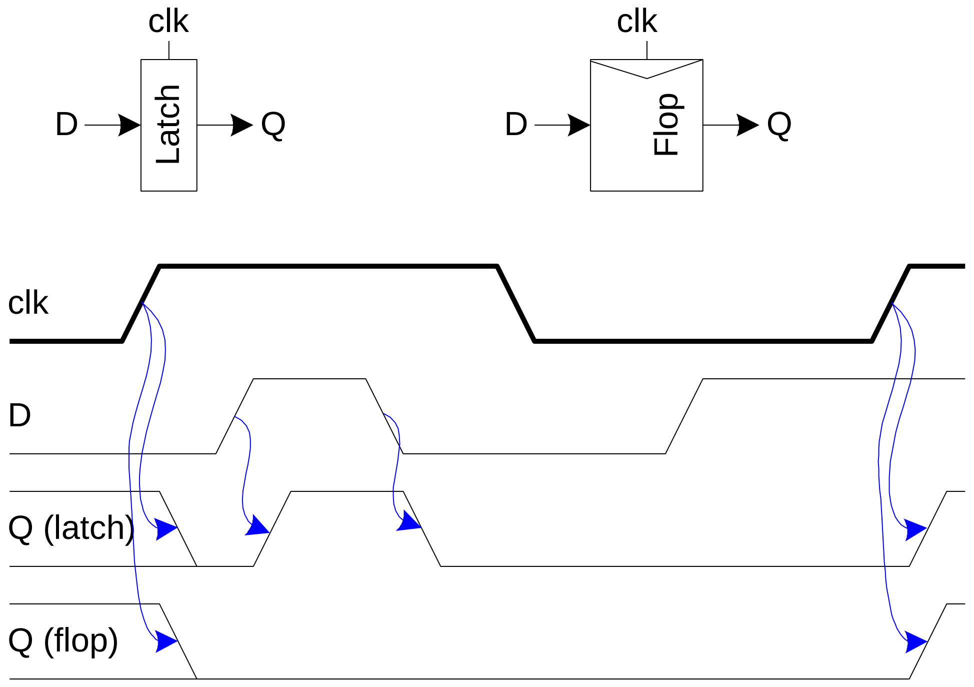
- **Latch**: Level sensitive
  - a.k.a. transparent latch, D latch
  - Transparent/opaque
- **Flip-flop**: edge triggered
  - a.k.a. master-slave flip-flop, D flip-flop, D register
  - Simply, a pair of latches using clk and its complement
  - Edge-triggered

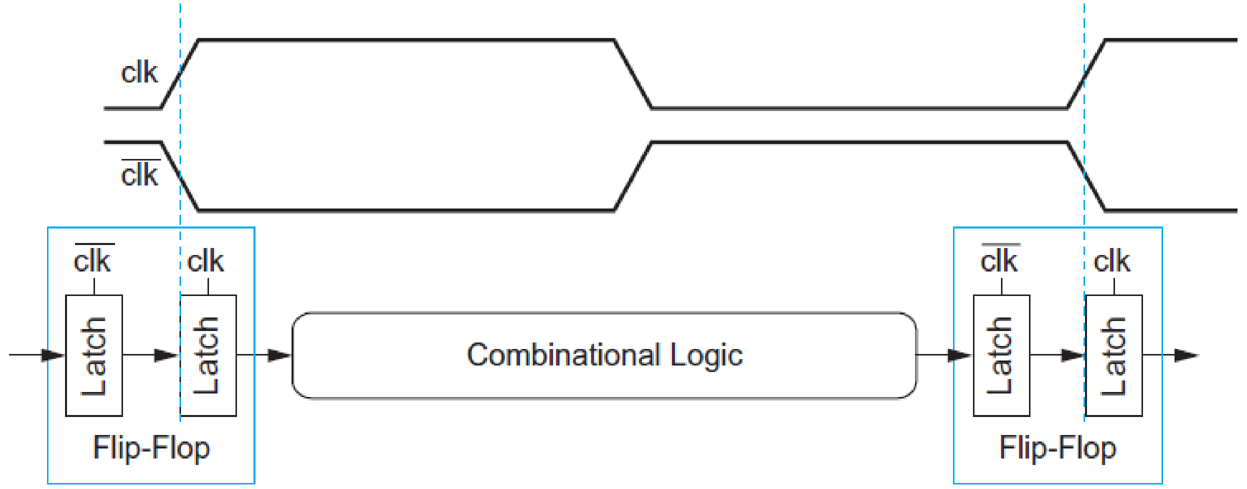




Sequencing Elements

- **Latch**: Level sensitive
  - a.k.a. transparent latch, D latch
  - Transparent/opaque
- **Flip-flop**: edge triggered
  - a.k.a. master-slave flip-flop, D flip-flop, D register
  - Simply, a pair of latches using clk and its complement
  - Edge-triggered





Sequencing Methods

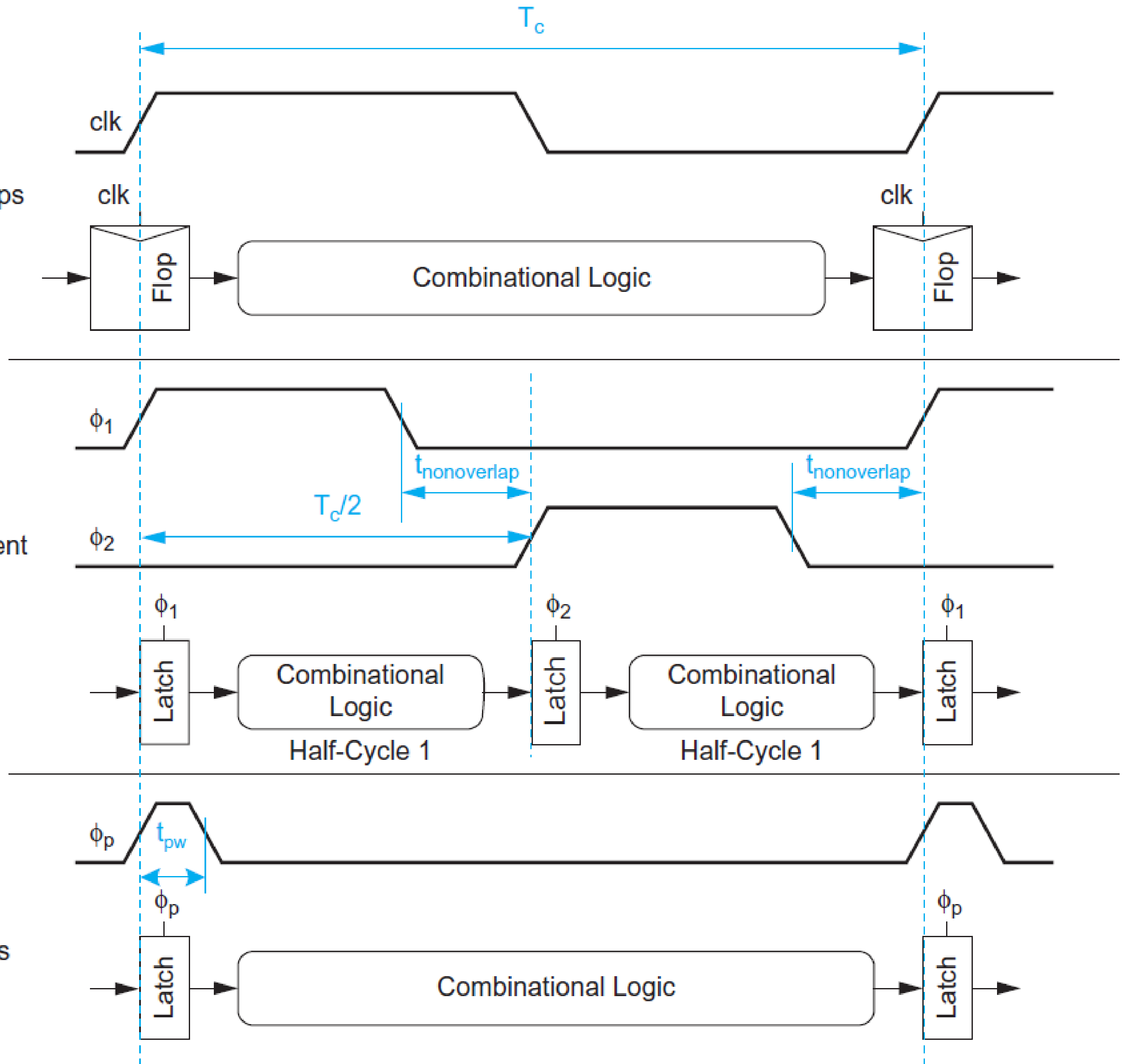
- Ideal sequencing methodology
  - Introduce no sequencing overhead
  - Allow sequencing elements back-to-back with no logic in between
  - Grant the designer flexibility in balancing the amount of logic in each clock cycle
  - Tolerate moderate amounts of clock skew without degrading performance
  - Consume zero area and power

Sequencing Methods for Static Circuits

1 Flip-Flops

2 2-Phase Transparent Latches

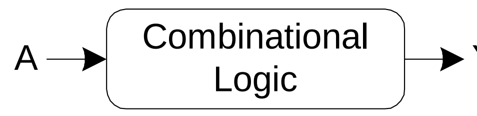
3 Pulsed Latches

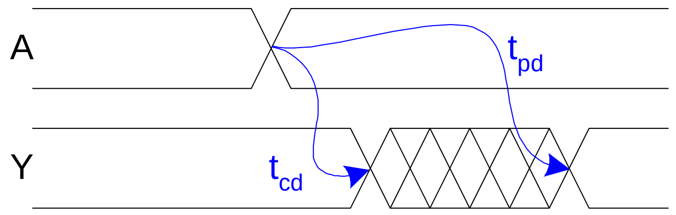


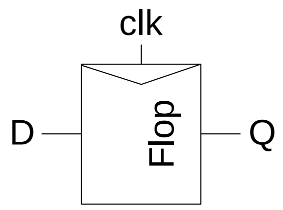
Timing Diagrams

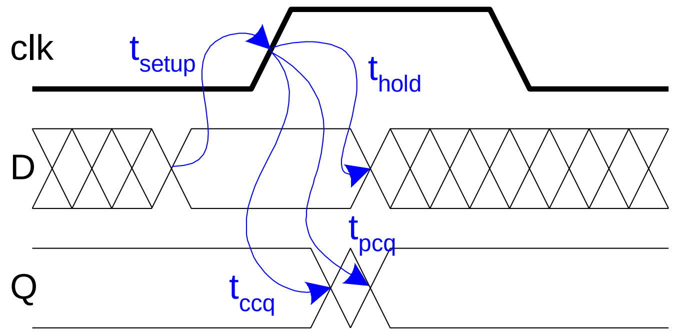
- ❑ Contamination delay: output may begin to change or glitch
- ❑ Propagation delay: output must have settled to final value

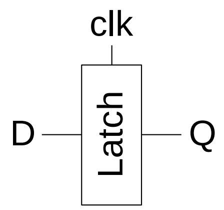
t_{pd}	Logic Prop. Delay
t_{cd}	Logic Cont. Delay
t_{pcq}	Latch/Flop Clk-to-Q Prop. Delay
t_{ccq}	Latch/Flop Clk-to-Q Cont. Delay
t_{pdq}	Latch D-to-Q Prop. Delay
t_{cdq}	Latch D-to-Q Cont. Delay
t_{setup}	Latch/Flop Setup Time
t_{hold}	Latch/Flop Hold Time
t_a	Aperture time

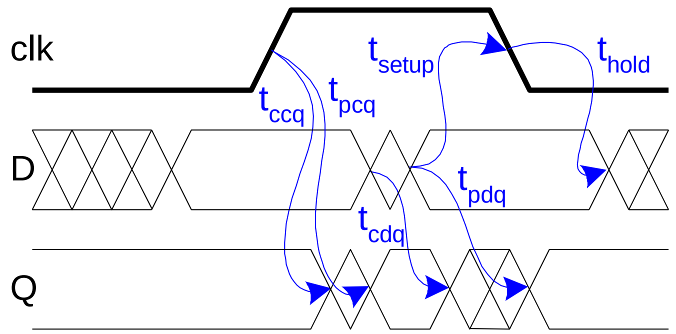






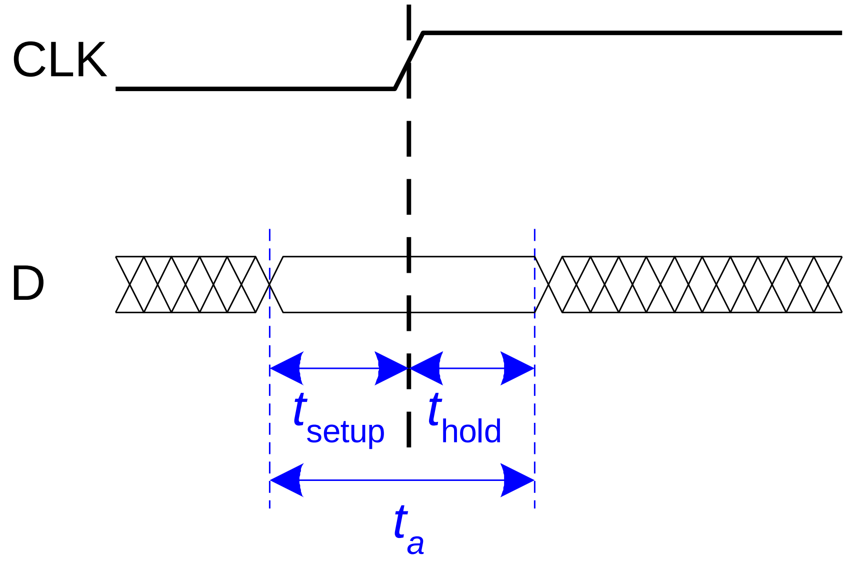






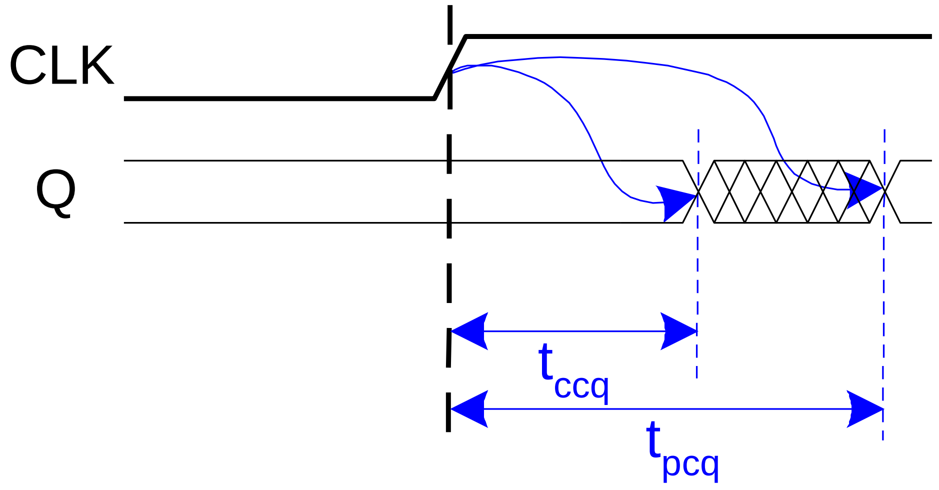
FF Input Timing Constraints

- Flip-flop samples *D* at clock edge
  - *D* must be stable when sampled
  - Similar to a photograph, *D* must be stable around clock edge
  - If not, metastability can occur
- **Setup time:** t_{setup} = time *before* clock edge data must be stable (i.e. not changing)
- **Hold time:** t_{hold} = time *after* clock edge data must be stable
- **Aperture time:** t_a = time *around* clock edge data must be stable ($t_a = t_{setup} + t_{hold}$)



FF Output Timing Constraints

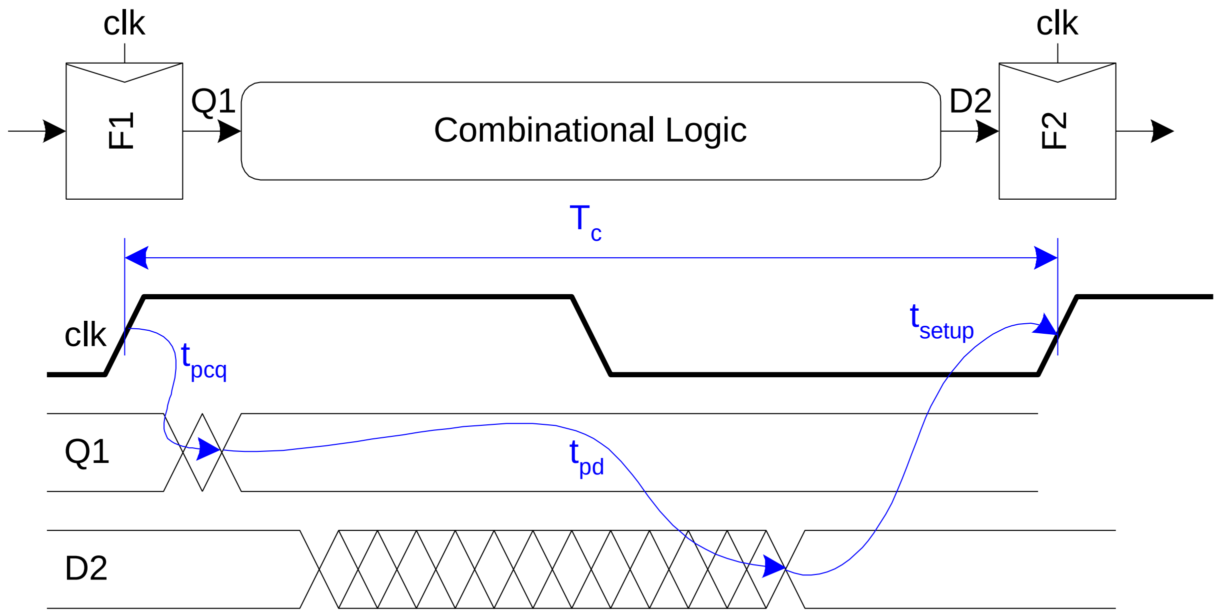
- **Propagation delay:** t_{pcq} = time after clock edge that the output Q is guaranteed to be stable (i.e., to stop changing)
- **Contamination delay:** t_{ccq} = time after clock edge that Q might be unstable (i.e., start changing)



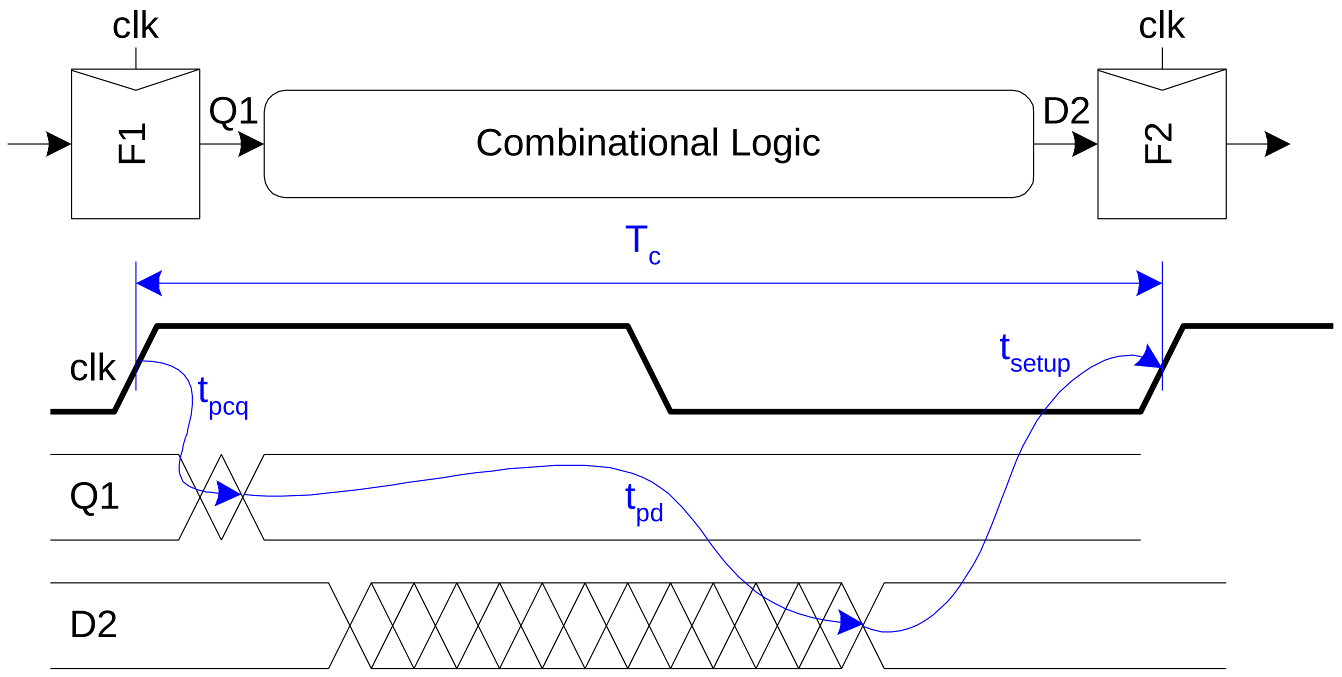
[Harris & Harris, DDCA, 2012]

Timing Constraints

- Dynamic discipline: Synchronous sequential circuit inputs must be stable during aperture (setup and hold) time around clock edge
  - The delay between registers has a **minimum** and **maximum** value, dependent on the delays of the circuit elements
- Max-delay constraints
  - Maximum propagation delay
  - Setup time failure
- Min-delay constraints
  - Minimum contamination delay
  - Hold time failure (race condition)



Max-Delay: Flip-Flops



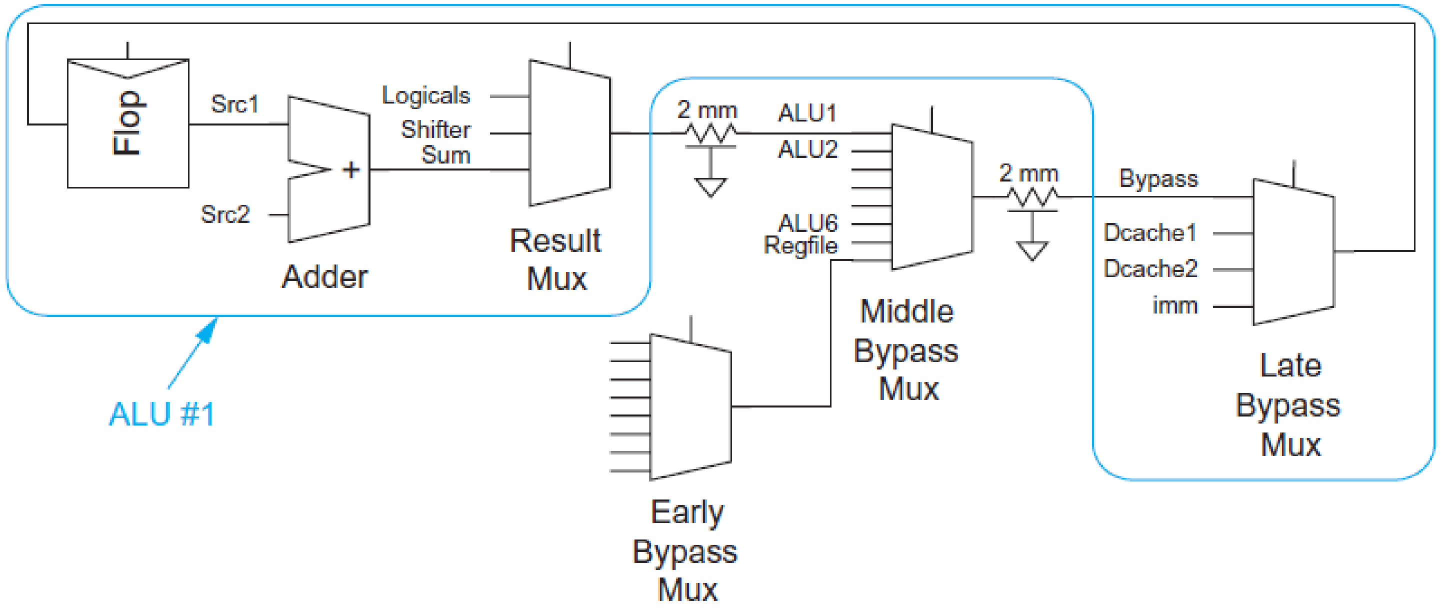
$$T_c \geq t_{pcq} + t_{pd} + t_{\text{setup}}$$

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

Example: Intel Itanium 2

- The ALU self-bypass is the critical path
- $t_{setup} = 62p, t_{pcq} = 90ps, t_{ccq} = 75ps$
- $T_c = t_{pd} + t_{cq} + t_{setup} = ?$

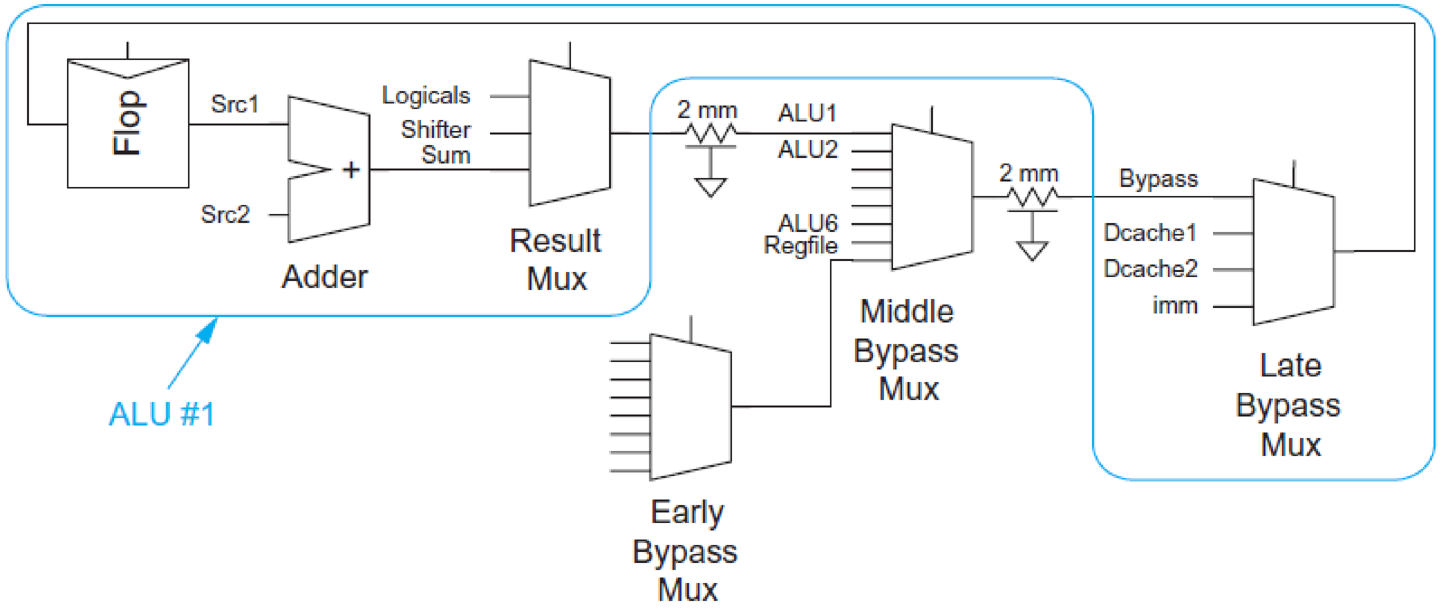
Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps



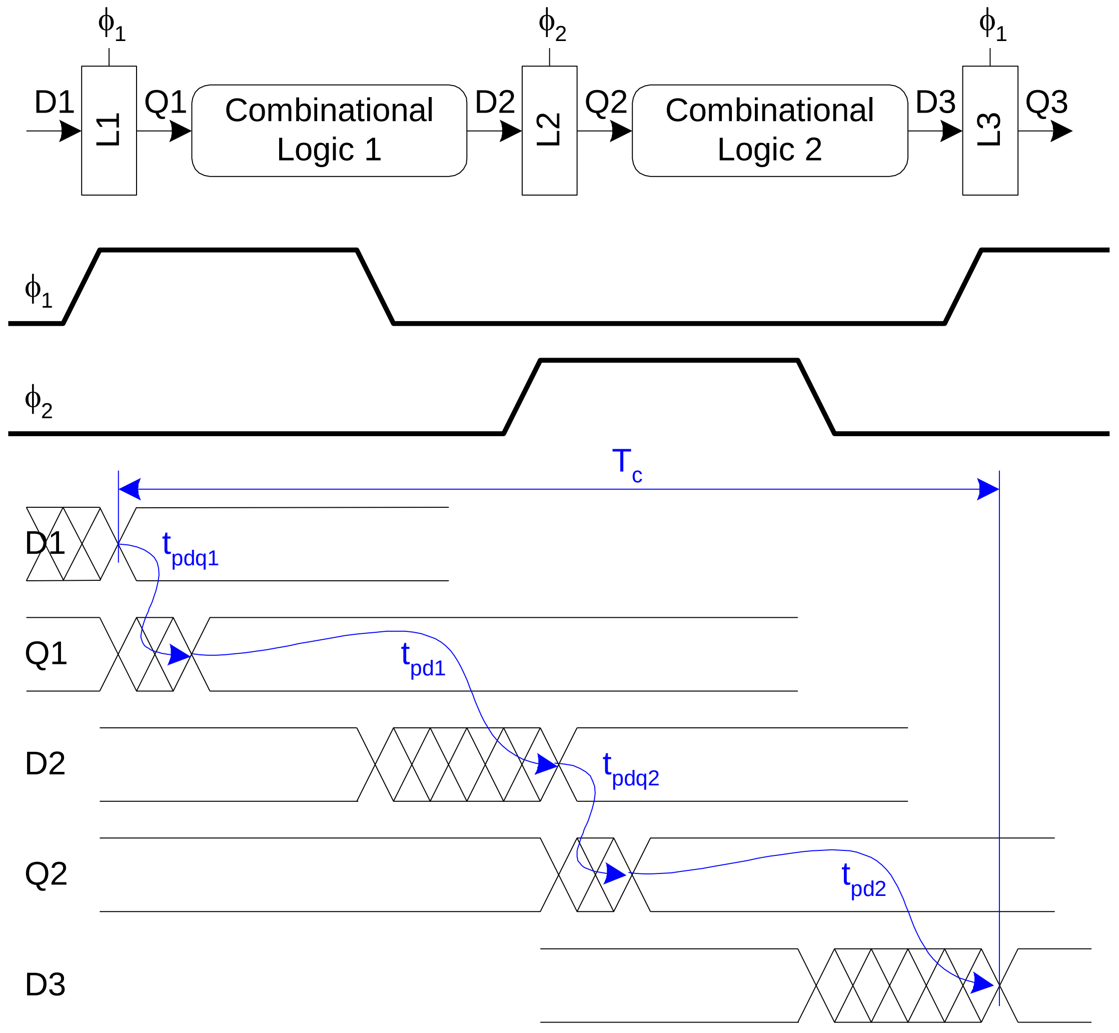
Example: Intel Itanium 2

- The ALU self-bypass is the critical path
- $t_{setup} = 62p, t_{pcq} = 90ps, t_{ccq} = 75ps$
- $T_c = t_{pd} + t_{cq} + t_{setup} = (590 + 60 + 100 + 80 + 100 + 70) + 90 + 62 = 1152ps$

Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps



Max Delay: 2-Phase Latches

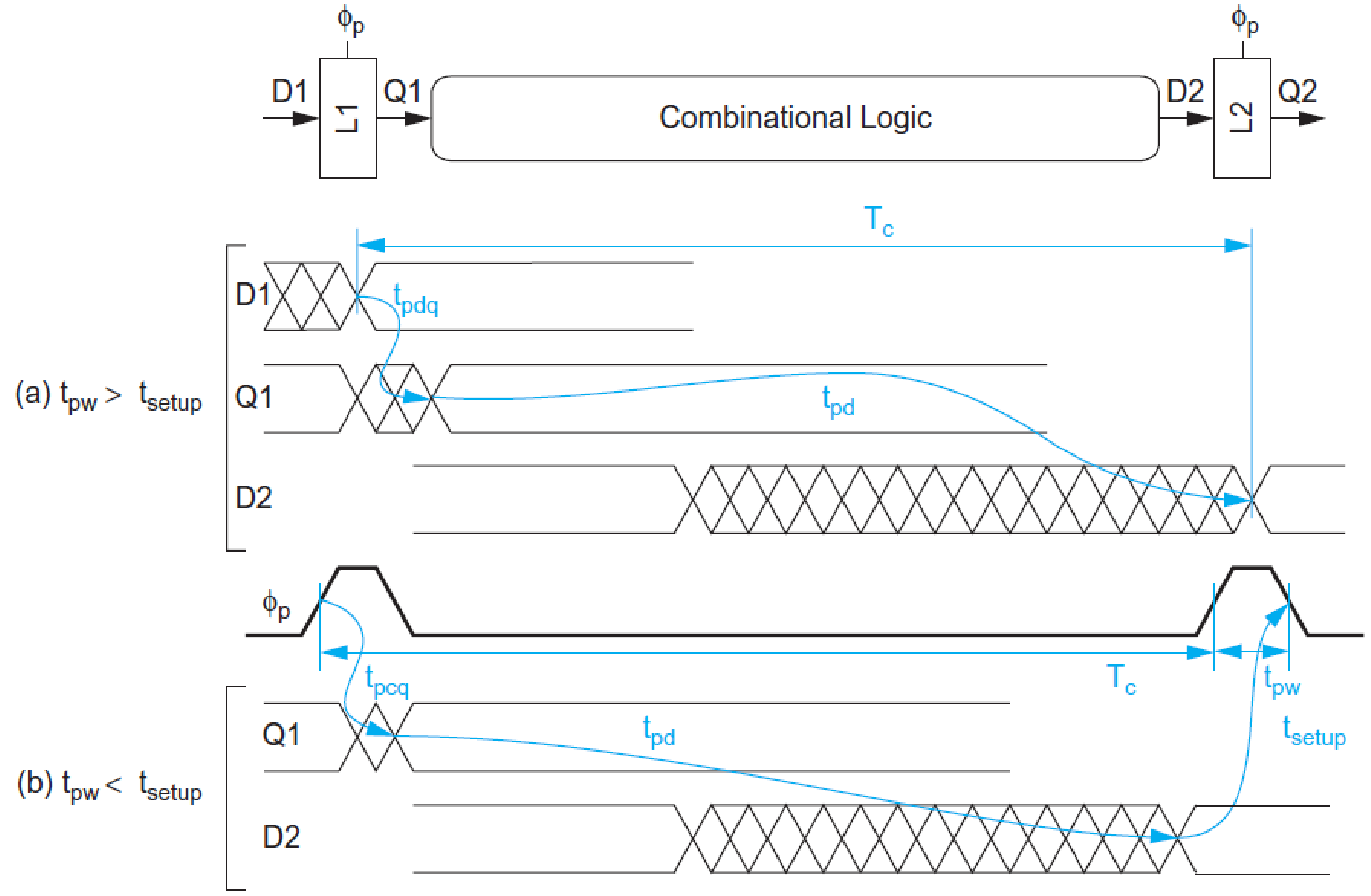


The nonoverlap between clocks does not degrade performance: data continues to propagate through the combinational logic

$$T_c \geq t_{pdq1} + t_{pd1} + t_{pdq2} + t_{pd2}$$

$$t_{pd} = t_{pd1} + t_{pd2} \leq T_c - \underbrace{\left(2t_{pdq}\right)}_{\text{sequencing overhead}}$$

Max Delay: Pulsed Latches



Input transition inside pulse

Input transition before pulse

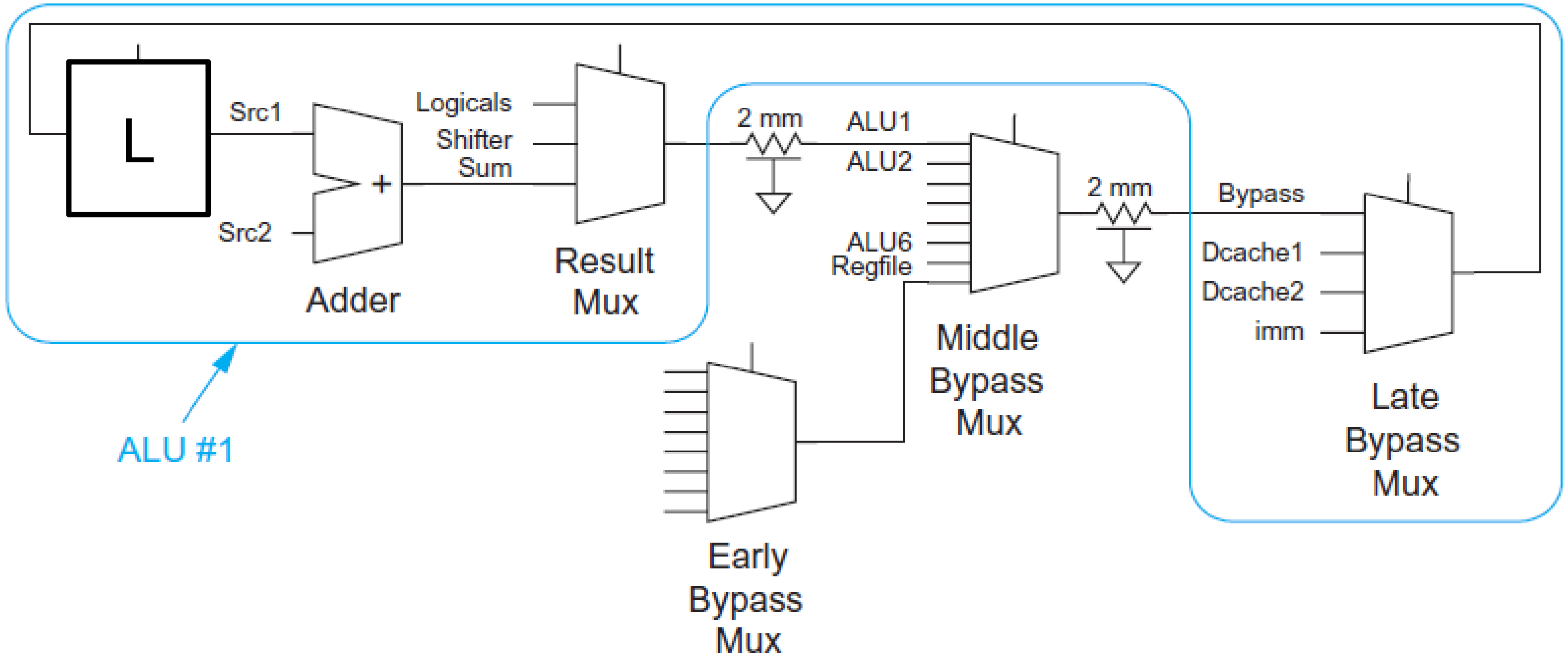
$$T_c \geq \max\left(t_{pdq} + t_{pd},\ t_{pcq} + t_{pd} + t_{\text{setup}} - t_{pw}\right)$$

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

Example: Intel Itanium 2

- The ALU self-bypass is the critical path
- Use Pulsed latch: $t_{pw} = 150ps, t_{setup} = 40ps, t_{pcq} = 82ps, t_{ccq} = 52ps, t_{pdq} = 92ps$
- $t_{pdq}\ ?\ t_{pcq} + t_{setup} - t_{pw}$
- $T_c = ?$

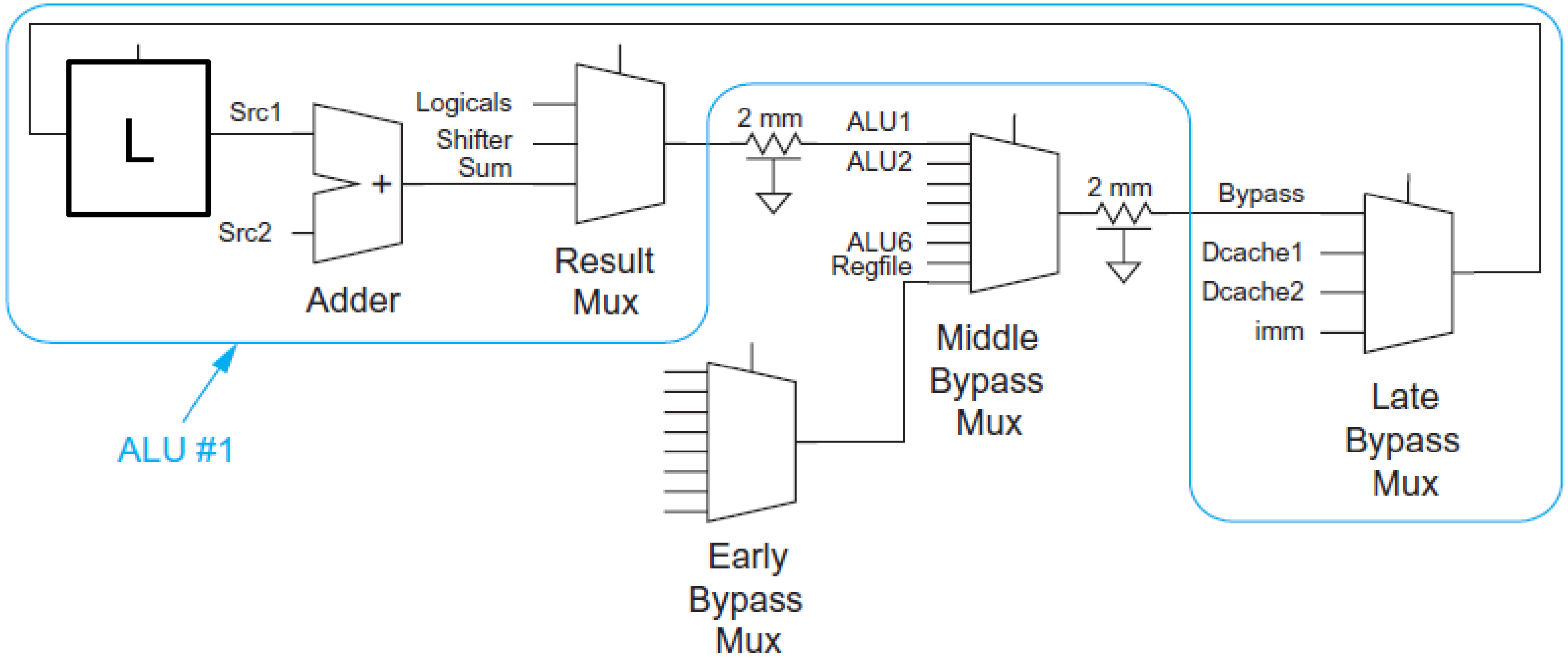
Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps



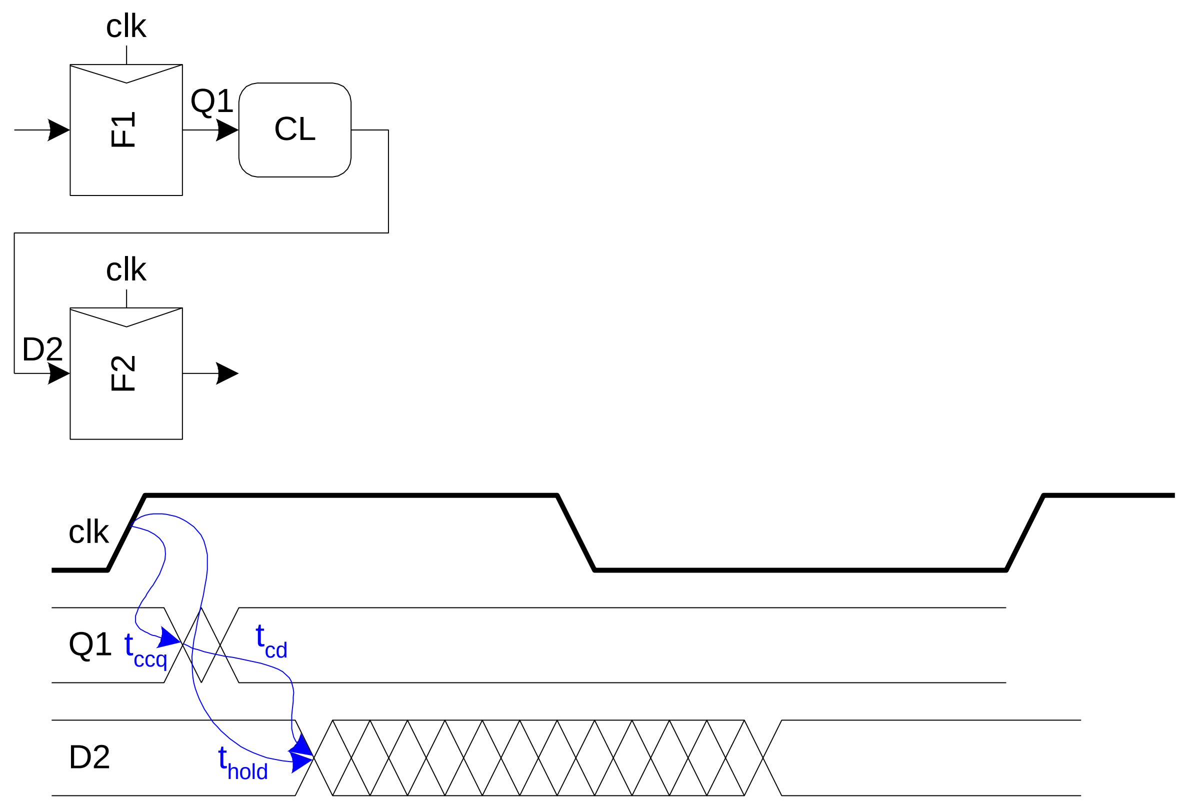
Example: Intel Itanium 2

- The ALU self-bypass is the critical path
- Use Pulsed latch: $t_{pw} = 150ps, t_{setup} = 40ps, t_{pcq} = 82ps, t_{ccq} = 52ps, t_{pdq} = 92ps$
- $t_{pdq} > t_{pcq} + t_{setup} - t_{pw}$
- $T_c = t_{pdq} + t_{pd} = 92 + (590 + 60 + 100 + 80 + 100 + 70) = 1092ps$

Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps



Min-Delay: Flip-Flops

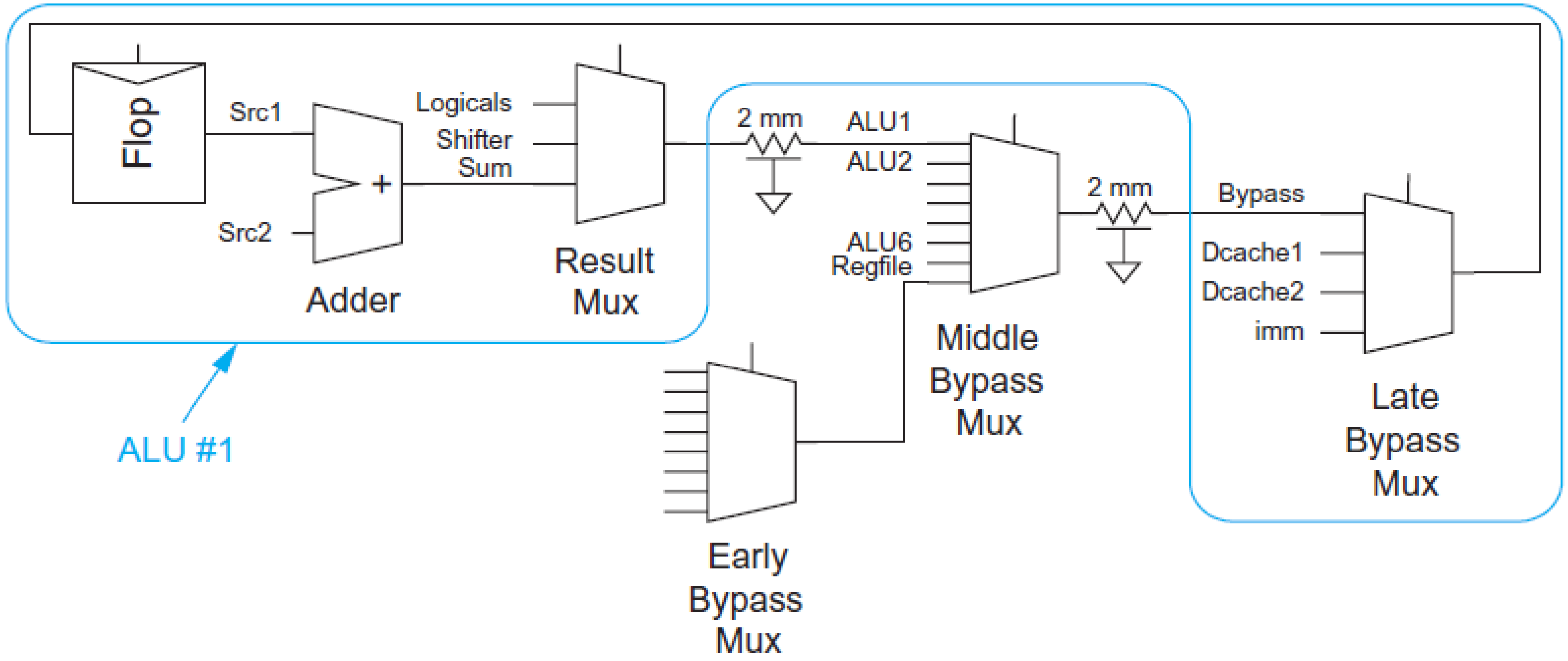


$$t_{ccq} + t_{cd} \geq t_{hold}$$

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

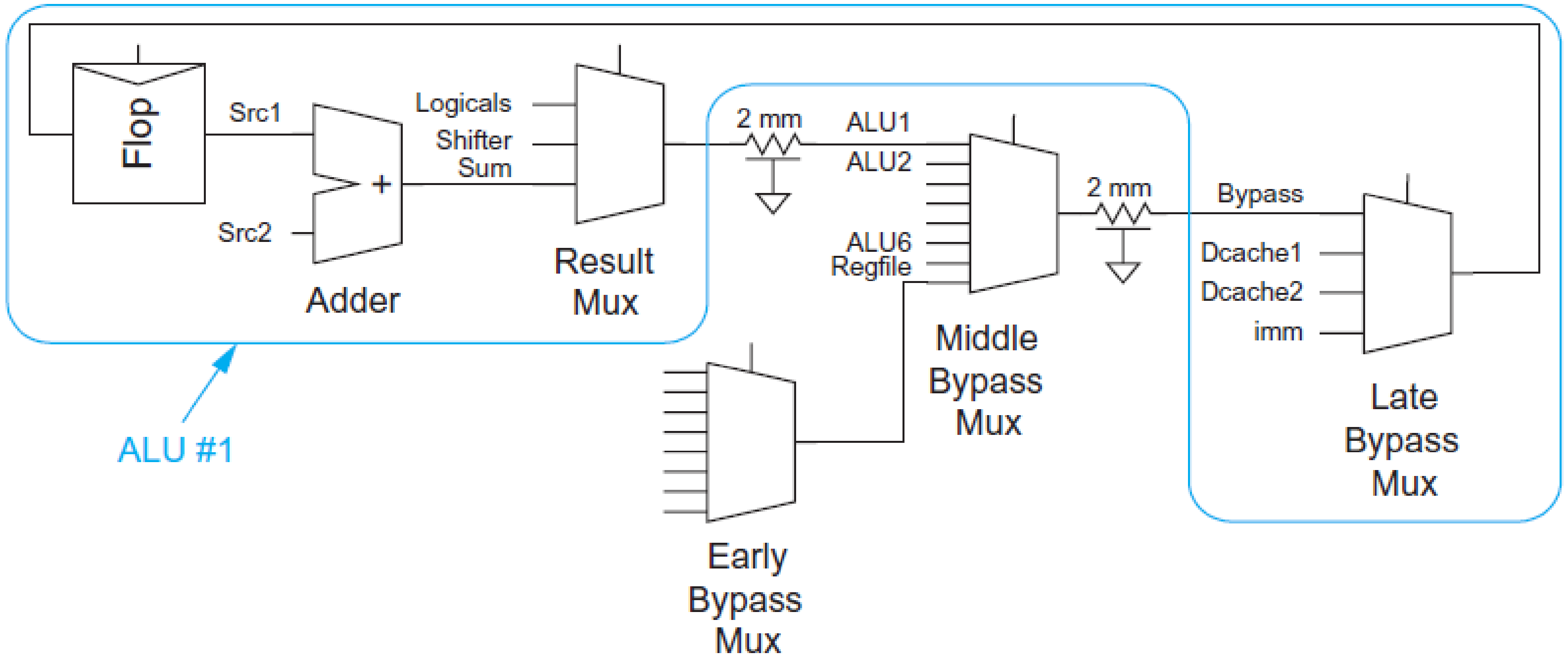
Example: Intel Itanium 2

- The earliest input is the *imm* at the late bypass mux, coming from another flip-flop
- $t_{cd} = 45ps, t_{hold} = -10ps, t_{ccq} = 75ps$
- Hold time violation?

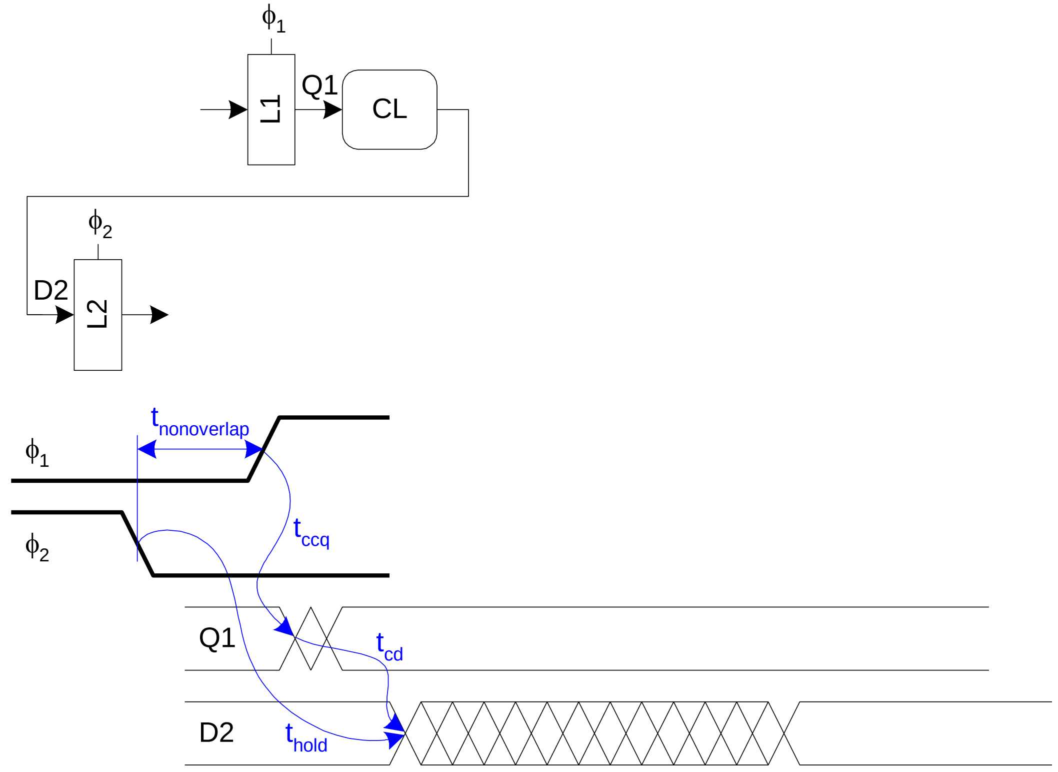


Example: Intel Itanium 2

- ❑ The earliest input is the *imm* at the late bypass mux, coming from another flip-flop
- ❑ $t_{cd} = 45ps, t_{hold} = -10ps, t_{ccq} = 75ps$
- ❑ Hold time is negative → Min delay condition is easily satisfied



Min-Delay: 2-Phase Latches

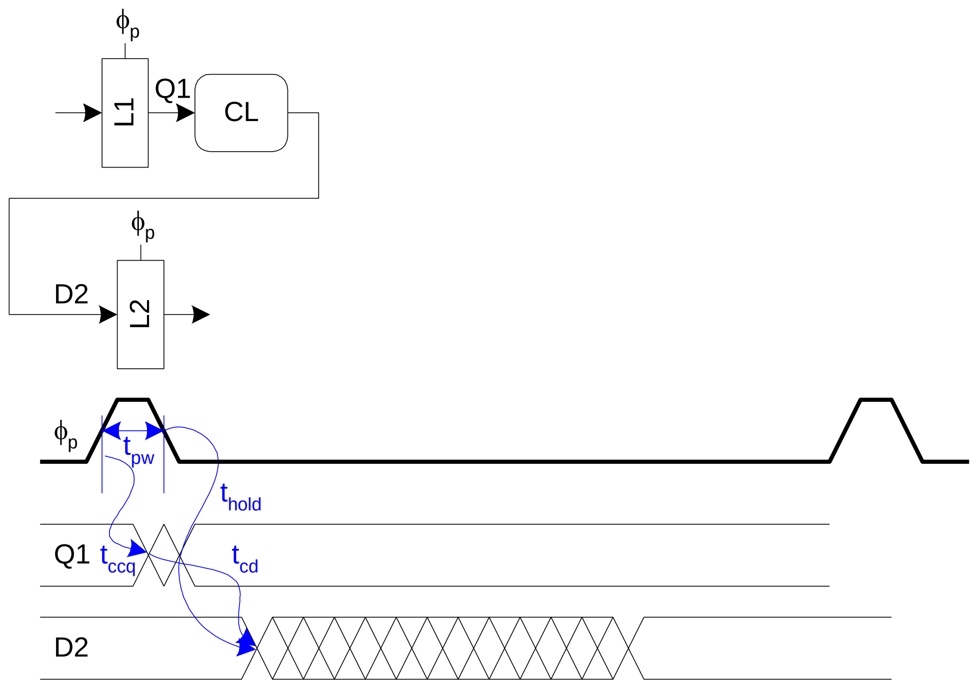


$$t_{nonoverlap} + t_{ccq} + t_{cd} \geq t_{hold}$$

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

Non-overlapping clocks help avoid race conditions, but difficult to generate and distribute at high speed

Min-Delay: Pulsed Latches



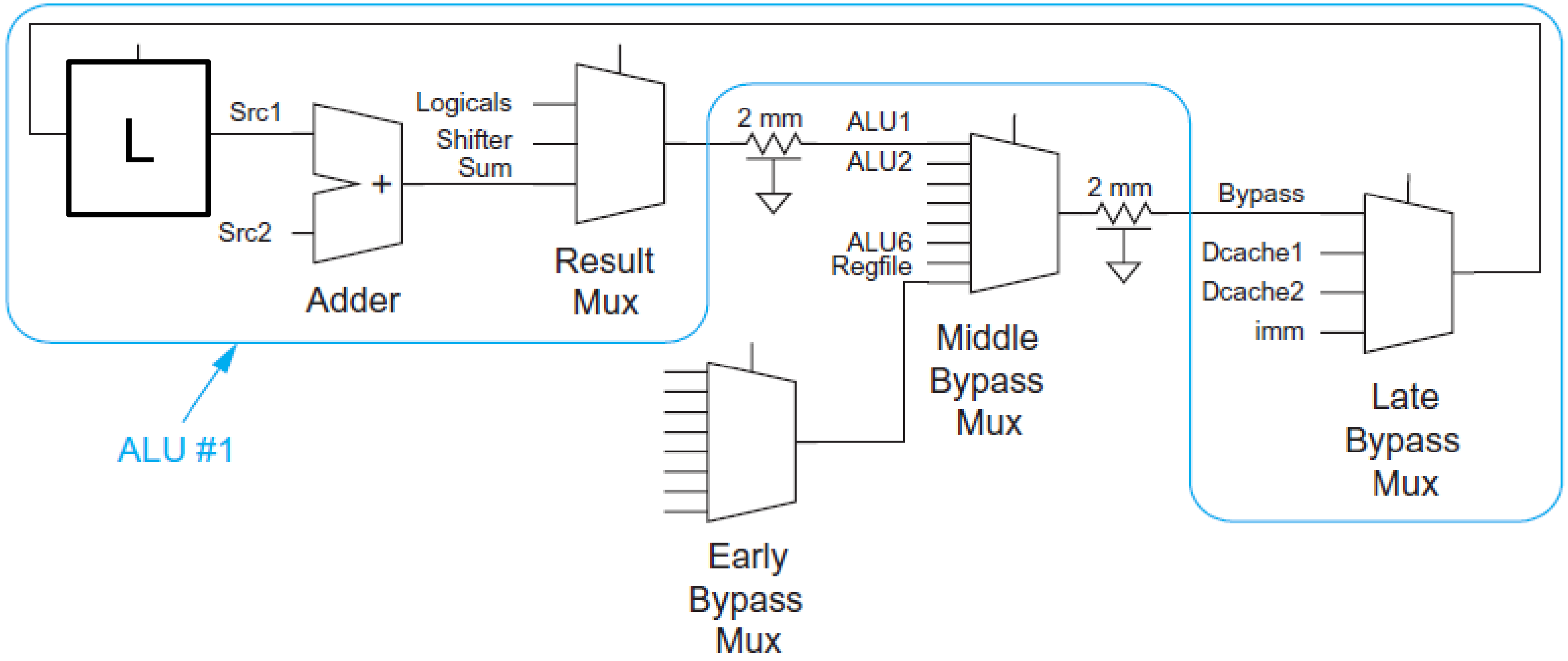
$$t_{ccq} + t_{cd} \geq t_{pw} + t_{hold}$$

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

Hold time risk: effectively increased by t_{pw} compared to flip-flops

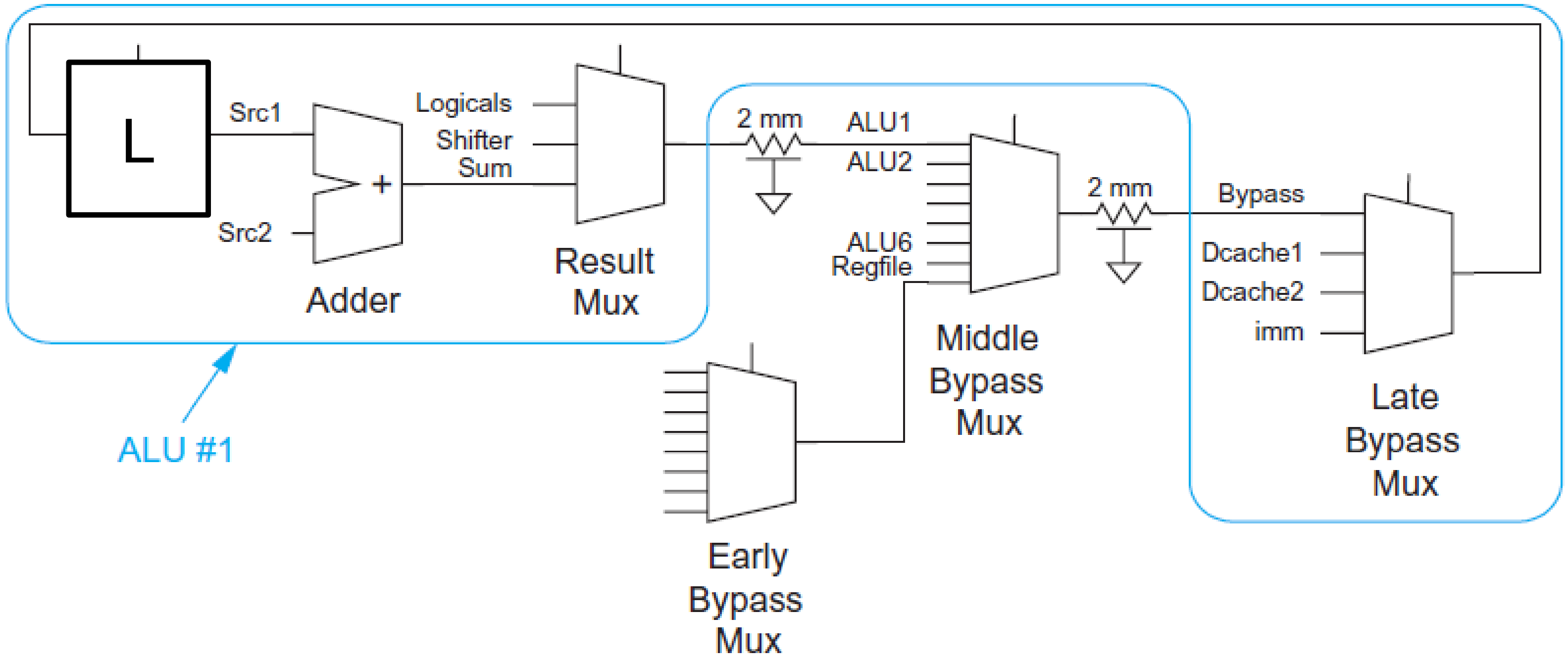
Example: Intel Itanium 2

- The earliest input is the *imm* at the late bypass mux, coming from another pulsed latch
- $t_{cd} = 45ps, t_{hold} = 5ps, t_{pw} = 150ps, t_{ccq} = 52ps$
- Hold time violation?

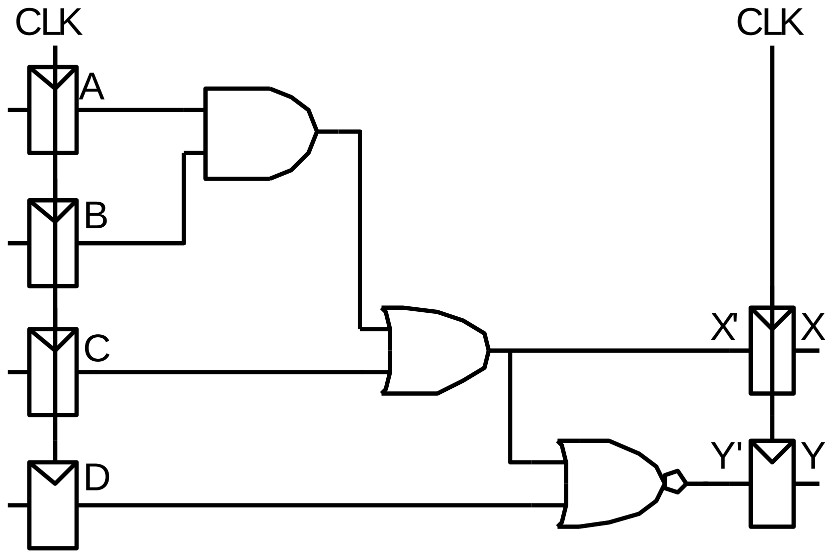


Example: Intel Itanium 2

- The earliest input is the *imm* at the late bypass mux, coming from another pulsed latch
- $t_{cd} = 45ps, t_{hold} = 5ps, t_{pw} = 150ps, t_{ccq} = 52ps$
- $t_{hold} - t_{ccq} + t_{pw} = 103ps > t_{cd}$ → Hold time violation
- Buffers must be added to delay *imm* by > 103 – 45 = 58ps



Example: Timing Analysis



Timing Characteristics

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} =

t_{cd} =

Setup time constraint:

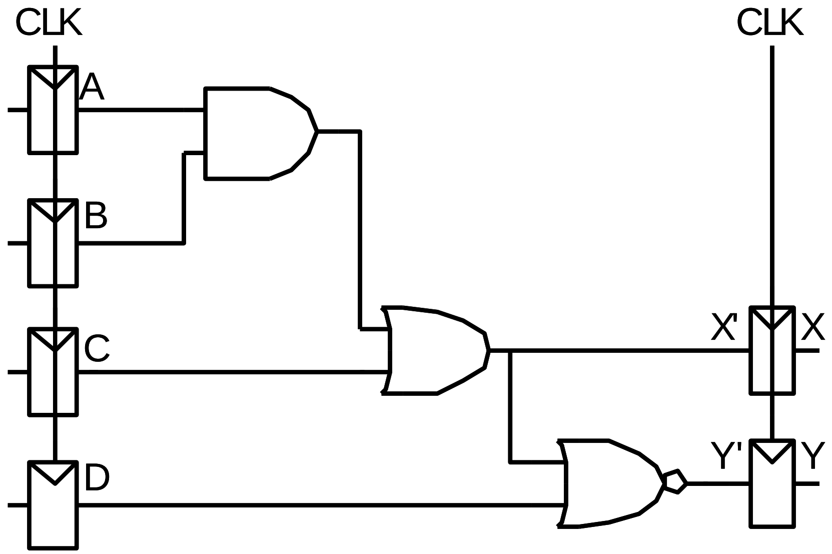
$T_c \geq$

f_c =

Hold time constraint:

$t_{ccq} + t_{cd} > t_{hold}$?

Example: Timing Analysis



Timing Characteristics

t_{ccq} = 30 ps

t_{pcq} = 50 ps

t_{setup} = 60 ps

t_{hold} = 70 ps

t_{pd} = 35 ps

t_{cd} = 25 ps

t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 25 ps

Setup time constraint:

$T_c \geq (50 + 105 + 60)$ ps = 215 ps

$f_c = 1/T_c$ = 4.65 GHz

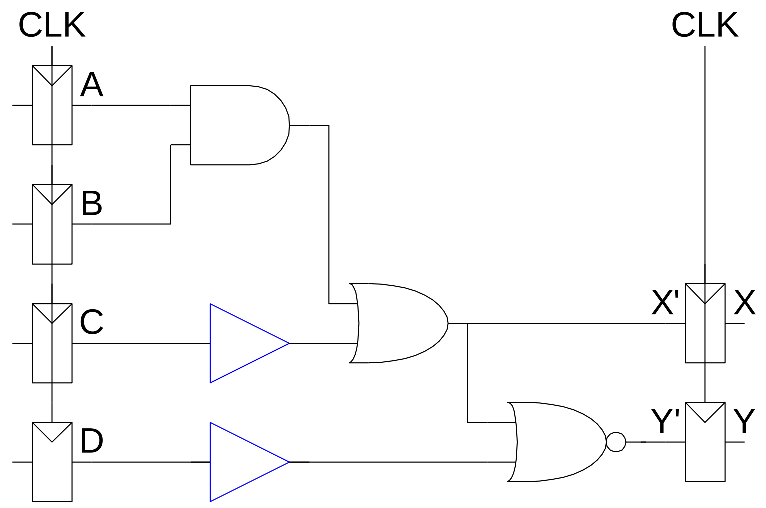
Hold time constraint:

$t_{ccq} + t_{cd} > t_{hold}$?

(30 + 25) ps > 70 ps ? **No!**

Example: Timing Analysis

Add buffers to the short paths:



t_{pd} =

t_{cd} =

Setup time constraint:

$T_c \geq$

f_c =

Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd}	= 35 ps
t_{cd}	= 25 ps

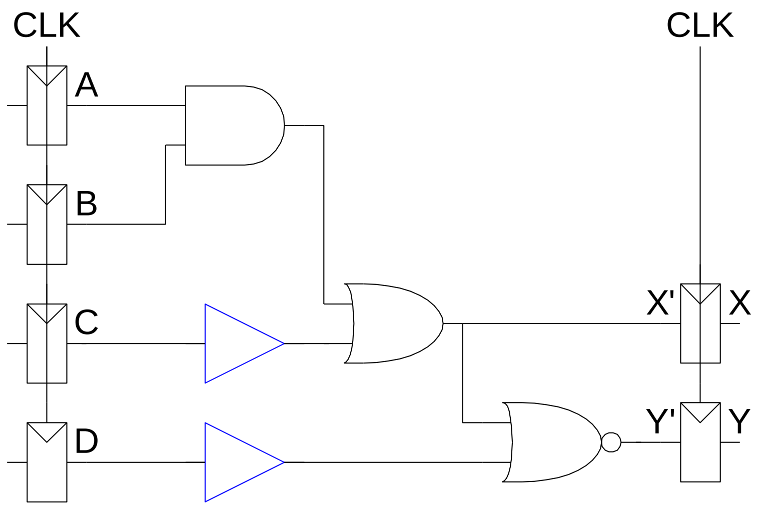
Hold time constraint:

$t_{ccq} + t_{cd} > t_{hold}$?

[Harris & Harris, DDCA, 2012]

Example: Timing Analysis

Add buffers to the short paths:



t_{pd} = 3 x 35 ps = 105 ps

t_{cd} = 2 x 25 ps = 50 ps

Setup time constraint:

$T_c \geq$ (50 + 105 + 60) ps = 215 ps

$f_c = 1/T_c$ = 4.65 GHz

Timing Characteristics

t_{ccq}	= 30 ps
t_{pcq}	= 50 ps
t_{setup}	= 60 ps
t_{hold}	= 70 ps
t_{pd}	= 35 ps
t_{cd}	= 25 ps

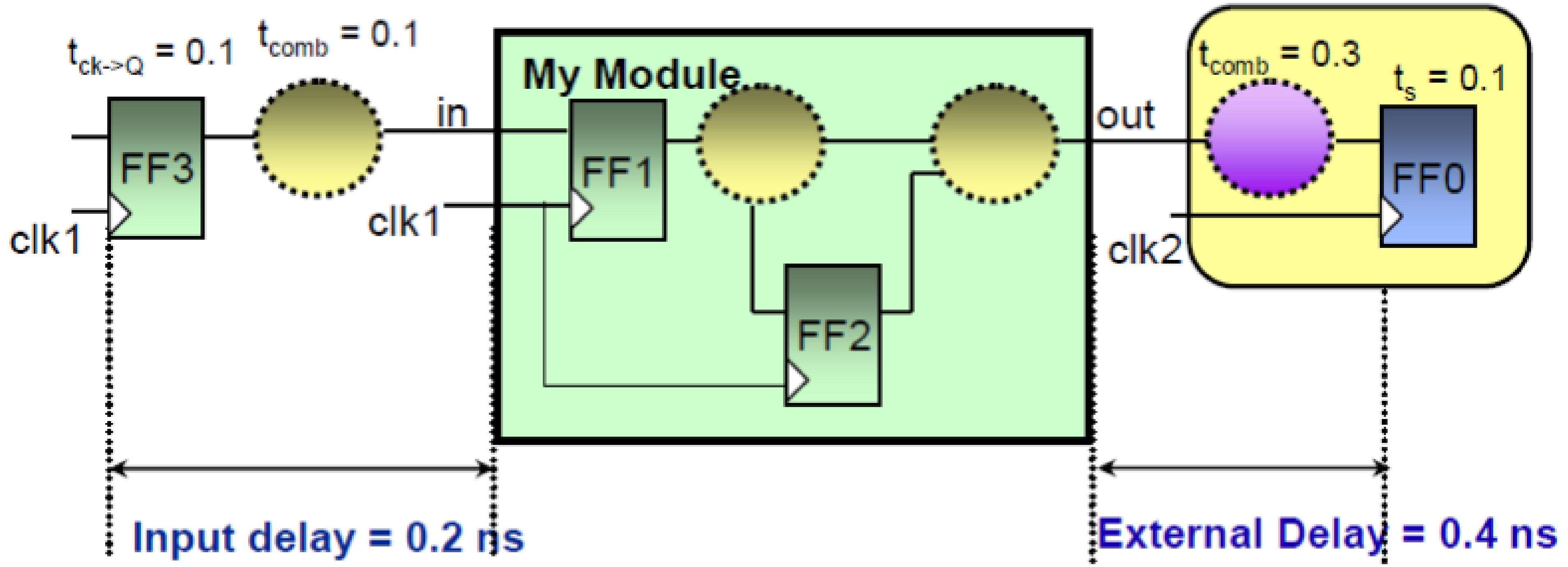
Hold time constraint:

$t_{\text{ccq}} + t_{cd} > t_{\text{hold}}$?

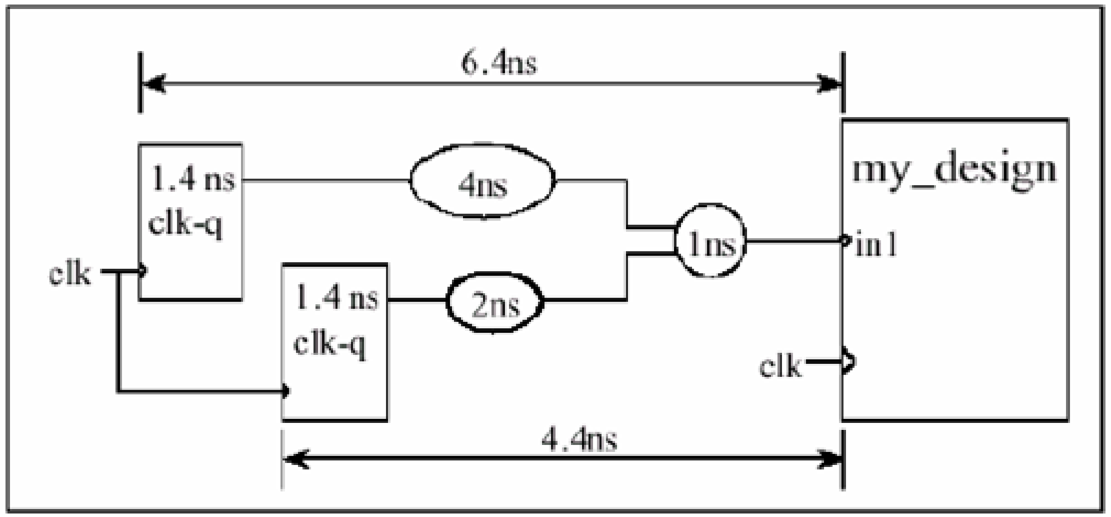
(30 + 50) ps > 70 ps ? **Yes!**

Timing Constraints in Synthesis Tools

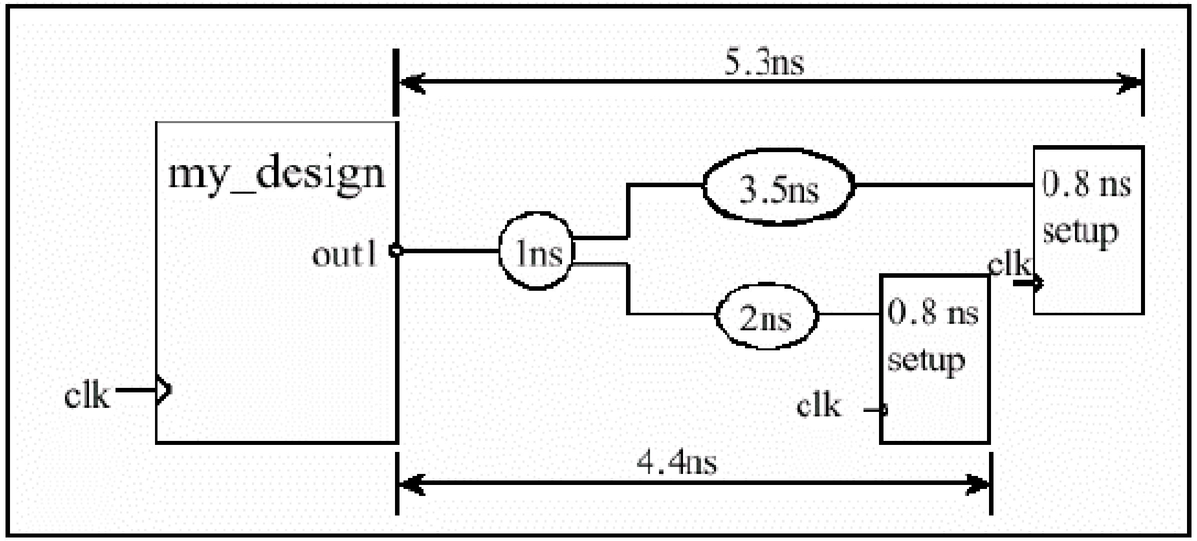
```
external_delay -clock clk1 -input 200 -name in_con [all_inputs]
external_delay -clock clk2 -output 400 -name out_con [all_outputs]
```



Timing Constraints in Synthesis Tools



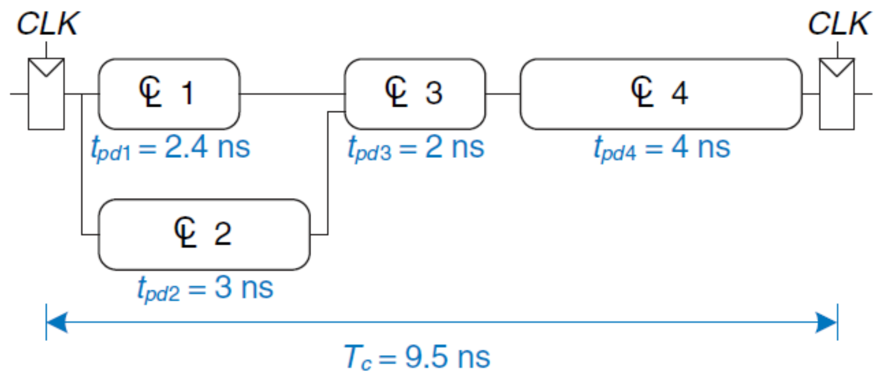
```
dc_shell>set_input_delay -clock clk -max 6.4 in1
dc_shell>set_input_delay -clock clk -min 4.4 in1
```



```
dc_shell>set_output_delay -clock clk -max 5.3 out1
```

Pipelining Revisited

- Assume sequencing overhead = 0.5 ns

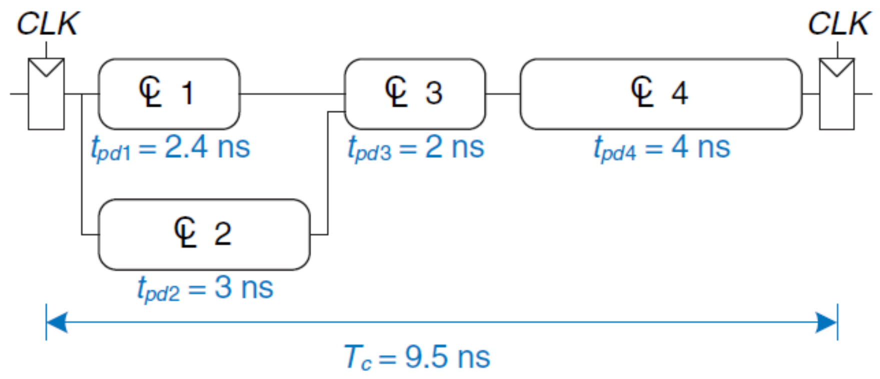


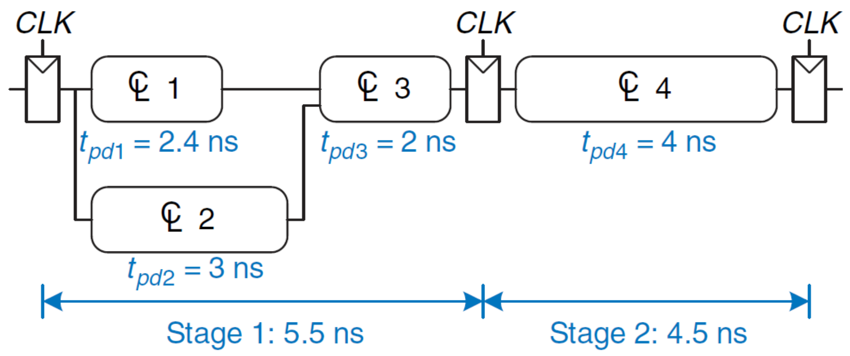
Token: Group of inputs processed to produce group of outputs
Latency: Time for one token to pass from start to end
Throughput: Number of tokens produced per unit time

[Harris & Harris, DDCA, 2012]

Pipelining Revisited

- Assume sequencing overhead = 0.5 ns



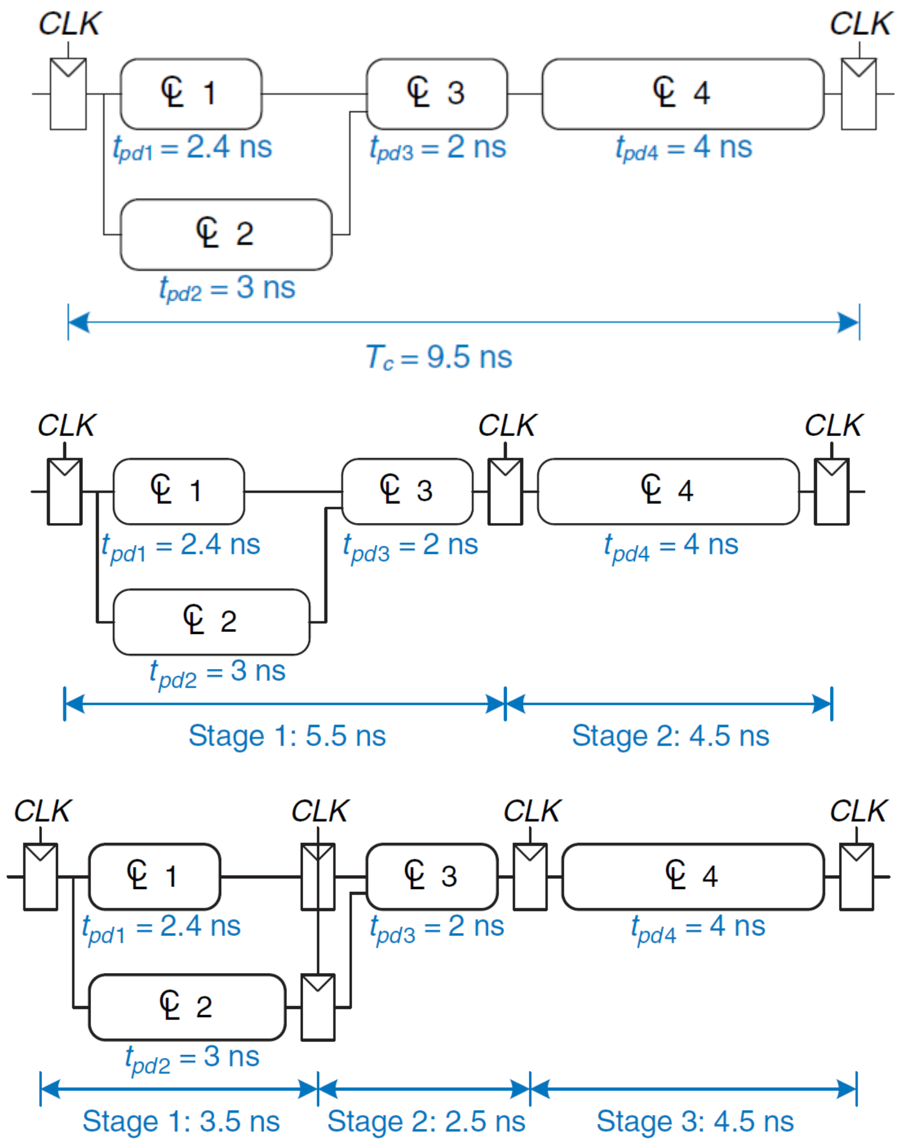


Token: Group of inputs processed to produce group of outputs
Latency: Time for one token to pass from start to end
Throughput: Number of tokens produced per unit time

[Harris & Harris, DDCA, 2012]

Pipelining Revisited

- Assume sequencing overhead = 0.5 ns



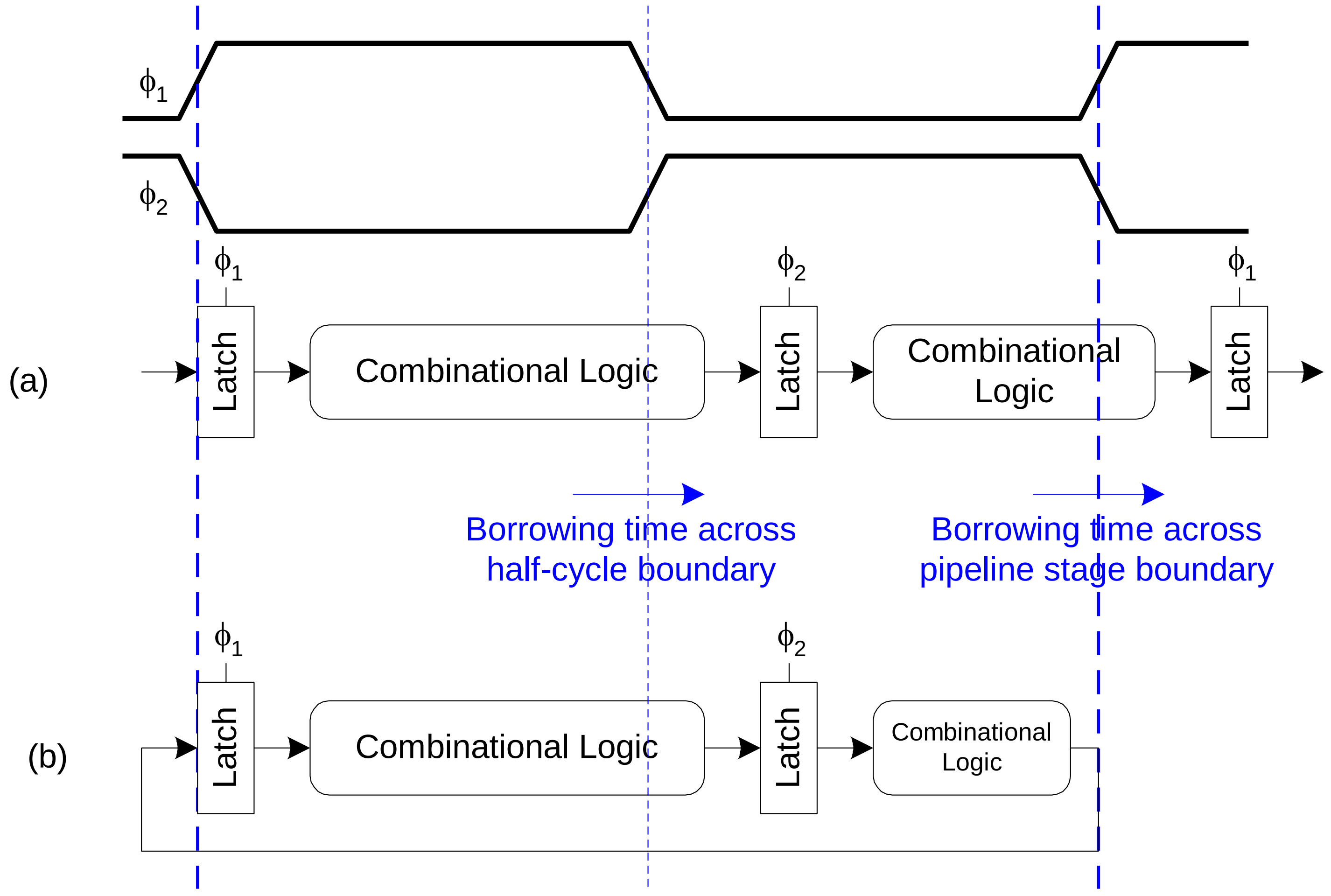
Token: Group of inputs processed to produce group of outputs
Latency: Time for one token to pass from start to end
Throughput: Number of tokens produced per unit time

[Harris & Harris, DDCA, 2012]

Time Borrowing

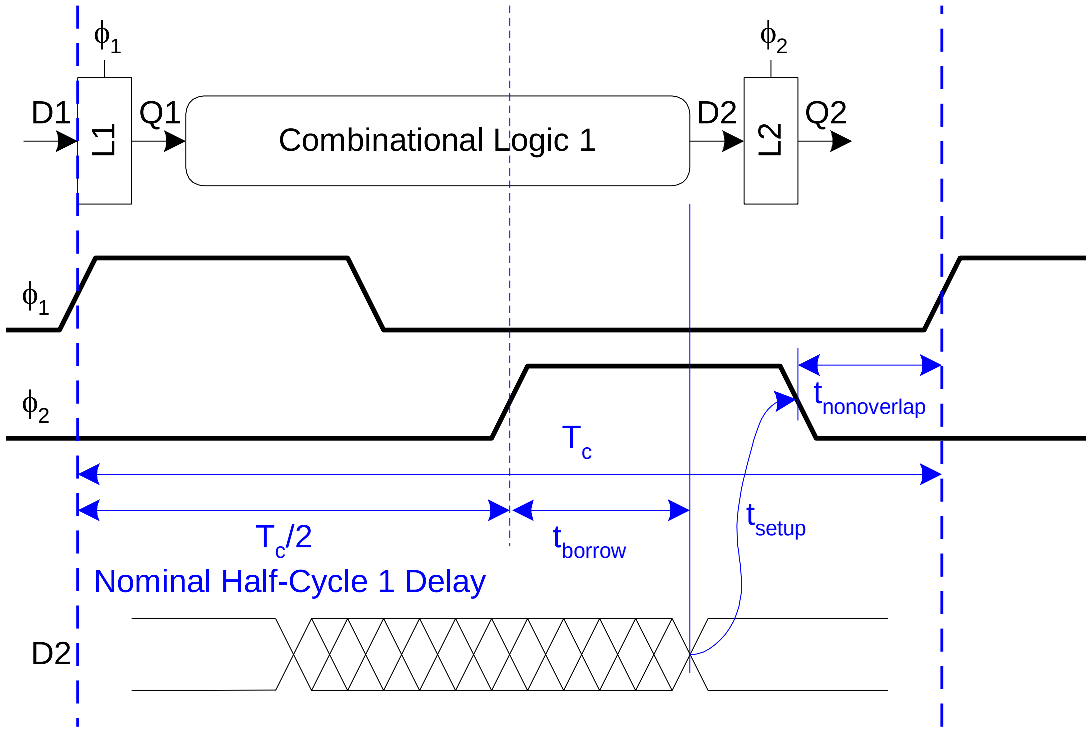
- In a flop-based system:
  - Data launches on one rising edge
  - Must setup before next rising edge
  - If it arrives late, system fails
  - If it arrives early, time is wasted
  - Flops have hard edges
- In a latch-based system
  - Data can pass through latch while transparent
  - Long cycle of logic can borrow time into next
  - As long as each loop completes in one cycle

Time Borrowing Example



Loops may borrow time internally but must complete within the cycle

How Much Borrowing?



2-Phase Latches

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

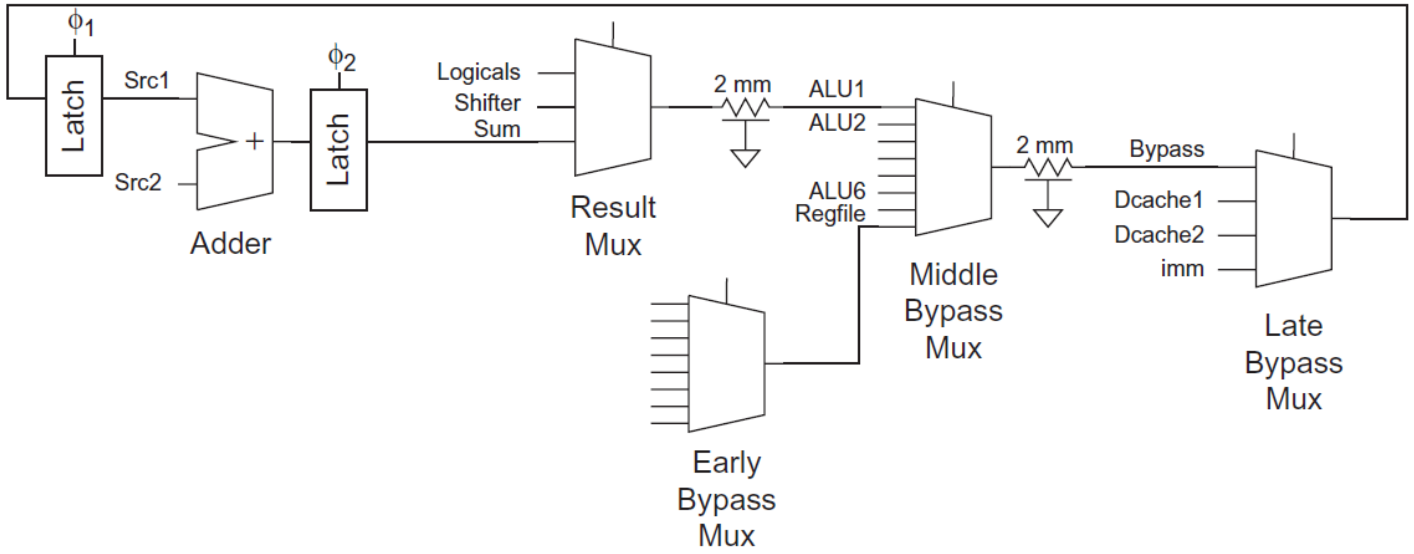
Pulsed Latches

$$t_{\text{borrow}} \leq t_{pw} - t_{\text{setup}}$$

$t_{borrow} = 0$ for a feedback cycle

Example: Intel Itanium 2

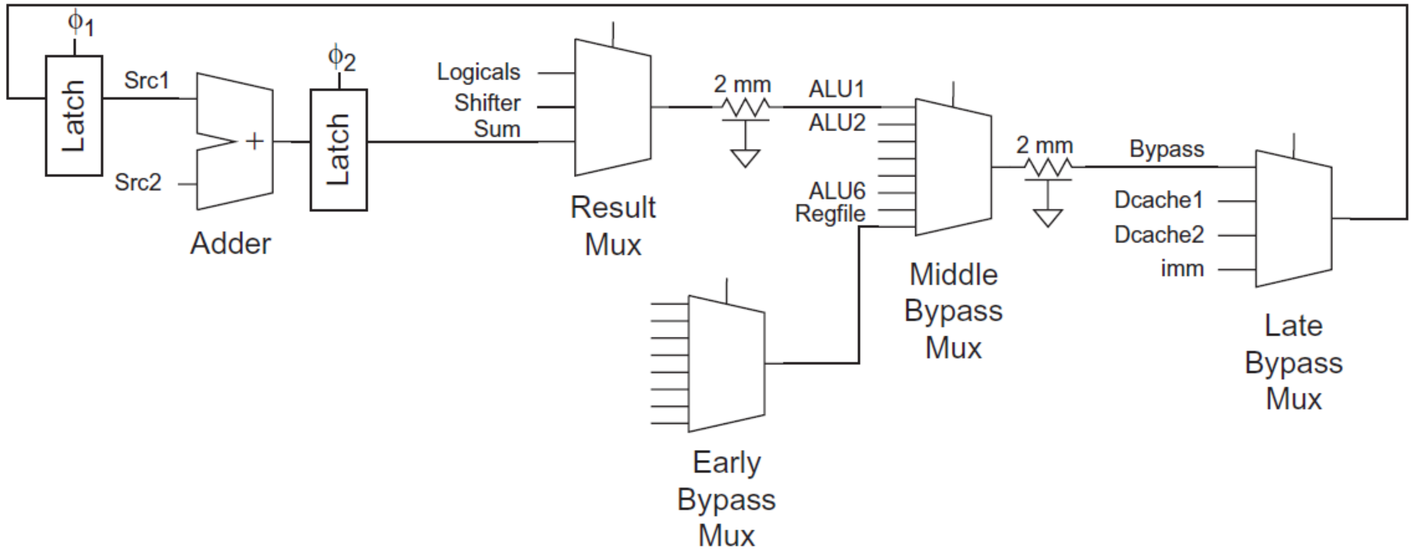
- $t_{setup} = 40ps, t_{hold} = 5ps, t_{pcq} = t_{pdq} = 82ps, t_{ccq} = 52ps$
- How much time is borrowed through the mid-cycle latch at min T_c? If the cycle time is increased to 2000 ps, how much time is borrowed?



Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps

Example: Intel Itanium 2

- $t_{setup} = 40ps, t_{hold} = 5ps, t_{pcq} = t_{pdq} = 82ps, t_{ccq} = 52ps$
- How much time is borrowed through the mid-cycle latch at min T_c? If the cycle time is increased to 2000 ps, how much time is borrowed?
- $T_c = t_{pdq1} + t_{pd1} + t_{pdq2} + t_{pd2} = 82 \times 2 + (590 + 60 + 100 + 80 + 100 + 70)$ $= 1164ps$
- Time borrowed $= t_{pdq1} + t_{pd1} - T_c/2 = 90ps$



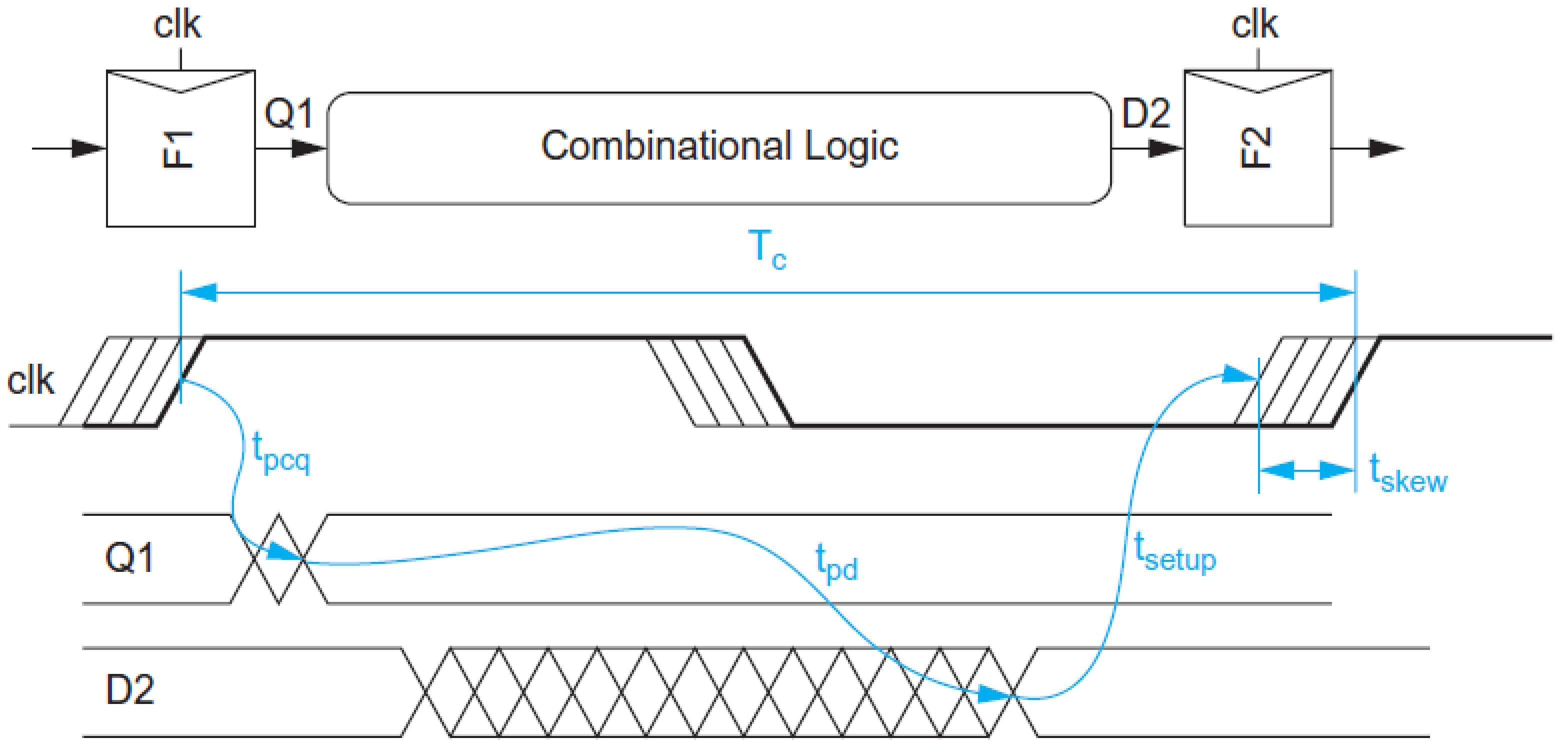
Element	Propagation Delay	Contamination Delay
Adder	590 ps	100 ps
Result Mux	60 ps	35 ps
Early Bypass Mux	110 ps	95 ps
Middle Bypass Mux	80 ps	55 ps
Late Bypass Mux	70 ps	45 ps
2-mm Wire	100 ps	65 ps

Clock Skew

- We have assumed zero clock skew
- Clocks really have uncertainty in arrival time
  - Decreases maximum propagation delay
  - Increases minimum contamination delay
  - Decreases time borrowing
- It is better to differentiate between
  - Skew and jitter
    - Skew: spatial variation across chip
    - Jitter: temporal variation at the same element
  - Positive/negative skew

Skew: Max-Delay: Flip-Flops

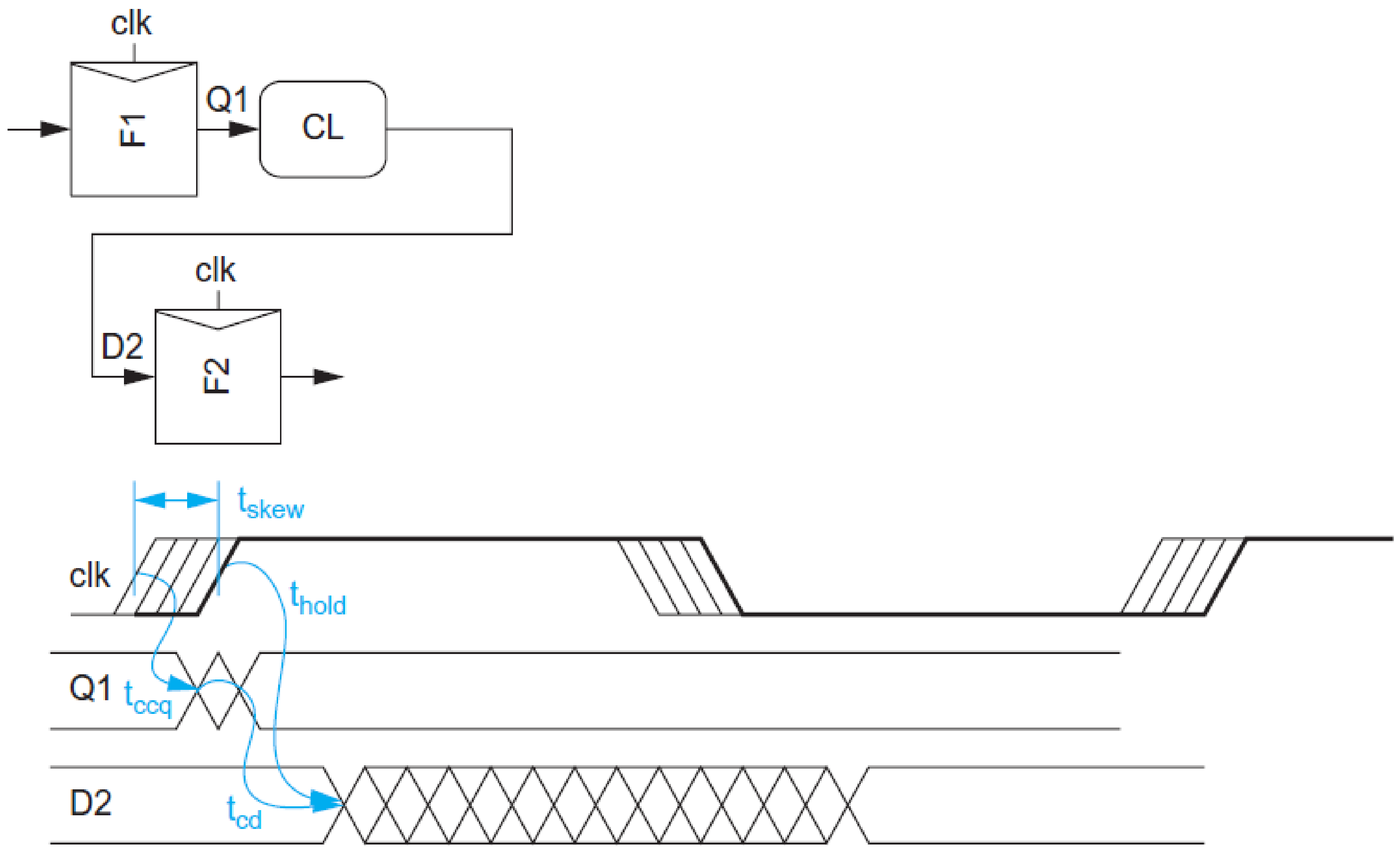
- Worst scenario: launching flop clock is late and the receiving flop clock is early.
  - Positive skew is good, negative skew is bad



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

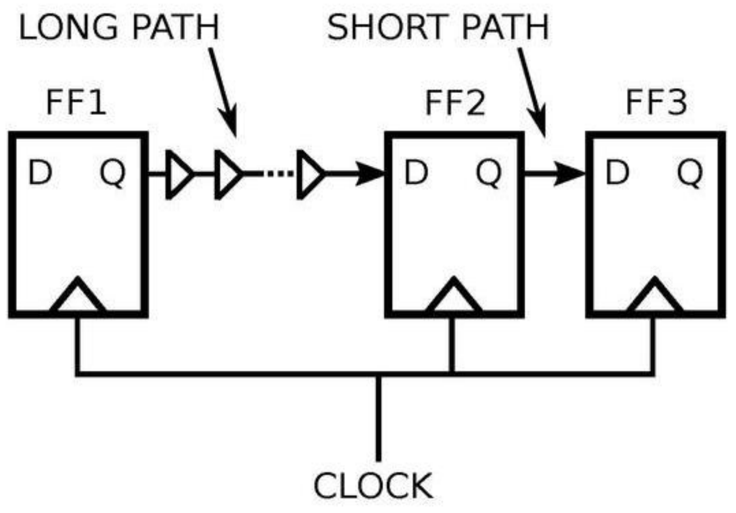
Skew: Min-Delay: Flip-Flops

- Worst scenario: launching flop clock is early and the receiving flop clock is late.
  - Positive skew is bad, negative skew is good



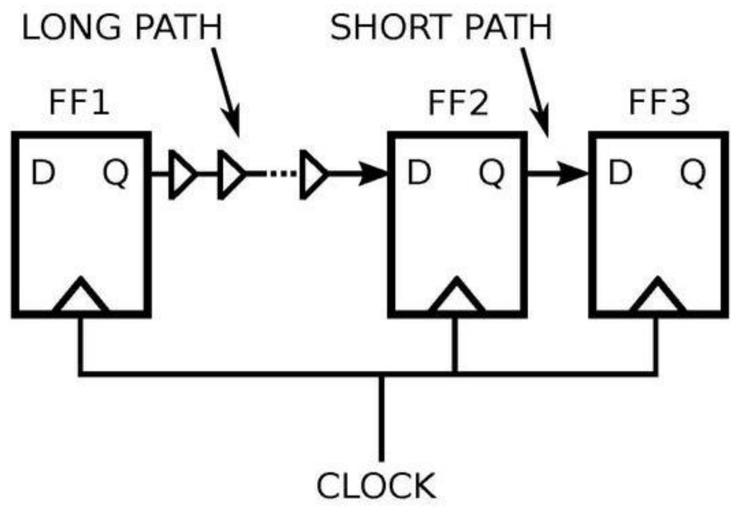
$$t_{cd} \geq t_{\text{hold}} - t_{ccq} + t_{\text{skew}}$$

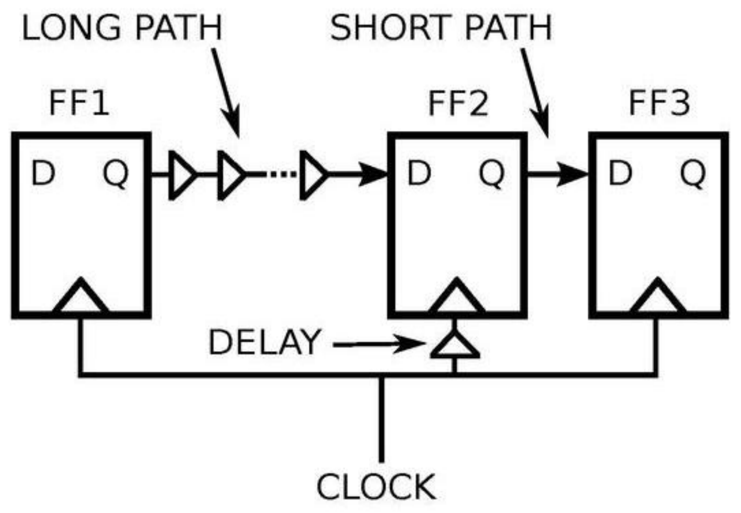
Example



[Wikipedia: Clock Skew]

Example

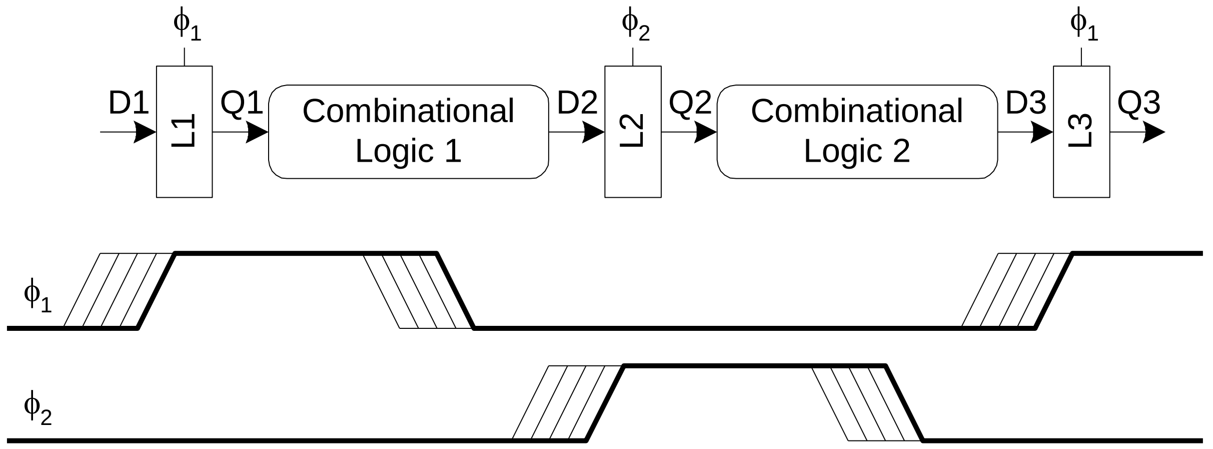




[Wikipedia: Clock Skew]

Skew: Latches

- For transparent latches, clock skew does not degrade performance (skew tolerant)
- But increases effective hold time and reduces t_{borrow}



2-Phase Latches

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

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

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

Pulsed Latches

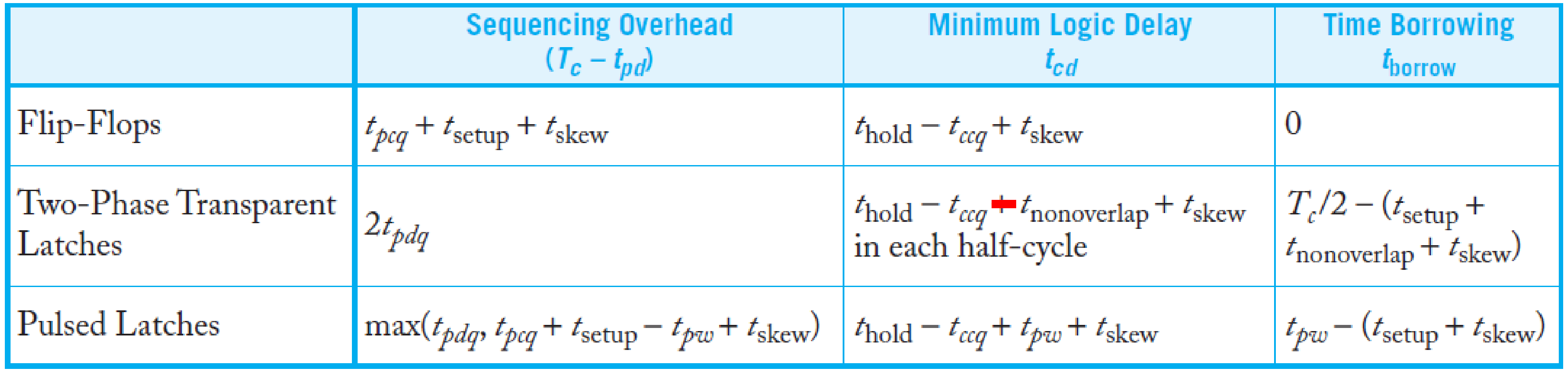
$$t_{pd} \le T_c - \underbrace{\max\left(t_{pdq}, t_{pcq} + t_{\text{setup}} - t_{pw} + t_{\text{skew}}\right)}_{\text{sequencing overhead}}$$

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

$$t_{\text{borrow}} \le t_{pw} - \left(t_{\text{setup}} + t_{\text{skew}}\right)$$

Sequencing Summary

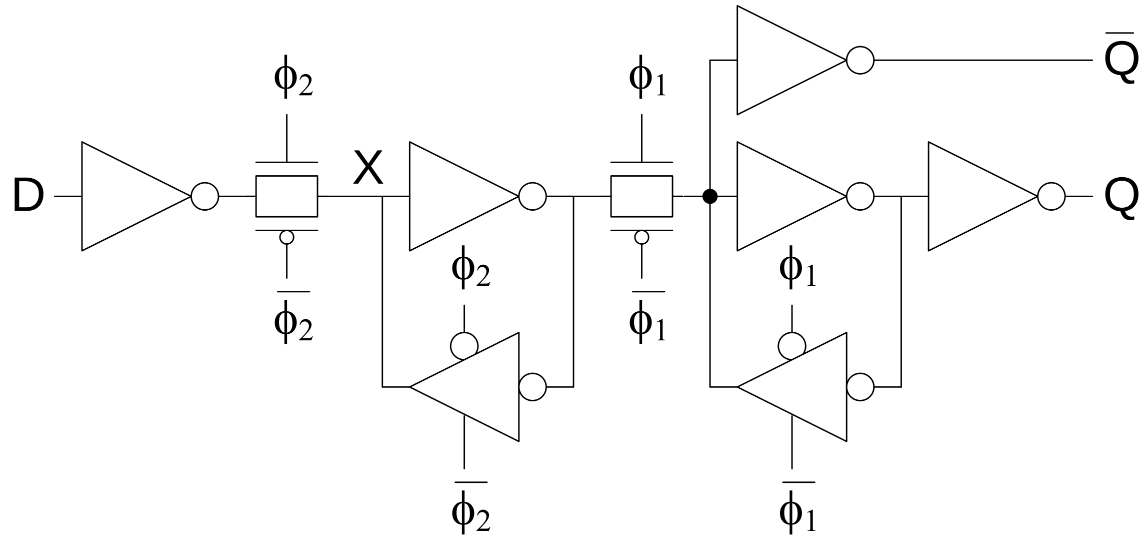
- Flip-Flops: **mainstream automated flow**
  - Very easy to use, supported by all tools
  - The right choice if performance is not near the process edge
- 2-Phase Transparent Latches: rarely used today
  - Lots of skew tolerance and time borrowing
  - More design effort to partition logic into half cycles
- Pulsed Latches: only for extreme performance – full custom (ex: Pentium 4 and Itanium 2)
  - Fast, lower power, some skew tolerance and time borrowing, **hold time risk**



	Sequencing Overhead $(T_c - t_{pd})$	Minimum Logic Delay t_{cd}	Time Borrowing t_{borrow}
Flip-Flops	$t_{pcq} + t_{setup} + t_{skew}$	$t_{hold} - t_{ccq} + t_{skew}$	0
Two-Phase Transparent Latches	$2t_{pdq}$	$t_{hold} - t_{ccq} + t_{nonoverlap} + t_{skew}$ in each half-cycle	$T_c/2 - (t_{setup} + t_{nonoverlap} + t_{skew})$
Pulsed Latches	$\max(t_{pdq}, t_{pcq} + t_{setup} - t_{pw} + t_{skew})$	$t_{hold} - t_{ccq} + t_{pw} + t_{skew}$	$t_{pw} - (t_{setup} + t_{skew})$

Academia Safe Flip-Flop

- If setup times are violated, reduce clock speed
- If hold times are violated, chip fails at any speed
- An easy way to guarantee hold times is to use 2-phase latches with big non-overlap times
  - Slow: nonoverlap adds to setup time
  - But no hold times
- In industry, use a better timing analyzer
  - Add buffers to slow signals if hold time is at risk



Thank you!