Clock Strategy

- Clocked Systems
- Latch and Flip-flops
- System timing
- Clock skew
- High speed latch design
- Phase locked loop
- Dynamic logic
- Multiple phase clock
- Clock distribution
Clocked Systems

Most VLSI systems are a combination of
- **pipelines and**
- **finite state machines (FSM)**

**Pipelined systems**

```
Input → D Q → Logic → D Q → Logic → ... → D Q → output
```

**Finite state machine**

```
CLK or CLKS → CLK or CLKS → CLK or CLKS → CLK or CLKS → CLK or CLKS
```

Logic

D  Q

CLK

Input

output
Single-phase clock timing waveforms

- **Clock Cycle Time (Tc)**: The time it takes for the clock signal to complete one cycle.
- **Setup Time (Ts)**: The time before the clock edge during which the data input (D) has to be stable.
- **Hold Time (Th)**: The time after the clock edge during which the data input (D) has to remain stable.
- **Clock-to-Q Delay (Tq)**: The delay from the clock edge to the Q output.

Ts: setup time -- the time before the clock edge during which the data input (D) has to be stable.

Th: hold time -- the time after the clock edge during which the data input (D) has to remain stable.

Tq: clock-to-Q delay -- The delay from the clock edge to the Q output.
Latches and flip-flops

(a) Negative Latch (Level sensitive)

(b) Positive Latch (Level sensitive)
Latches and flip-flops

(c) Positive edge-triggered register (single-phase clock)

(d) Pass transistor/inverter implementation
System timing

(A) Positive-edge triggered

(B) Alternatively, one may use latches as storage elements to save area.
System timing (con’t)

\[ T_{c1} > T_{qa} + T_{da} + T_{sb} \quad \text{and} \quad T_{co} > T_{qb} + T_{db} + T_{sc} \]

If \( T_c = T_{c1} + T_{co} \) and \( T_{c1} = T_{co} \), then \( T_{qa} = T_{qb} \), \( T_{sb} = T_{sc} \).

\[ \Rightarrow \text{The limit is} \quad T_c = T_{da} + T_{db} + 2(T_{q} + T_{s}) \]
Racing due to clock skew

1. If $T_{c2} > T_{c1} + t_{q1} + t_{d2}$
   - $M_2$ may sample a wrong data (current data)
   - Transparency problem

2. If $T_{c1} + t_{q1} + t_{d2} > T_c$ (cycle time)
   - $M_2$ cannot sample the previous data
Single-phase clock using D-FF

1) \( C_2 = \overline{C_1} \)

2) \( C_1 = \overline{C_2} \)

Wrong data in L2 → Correct data

Wrong only if

FSM → “feedback” or
Pipeline → “feedthrough”
To eliminate (reduce) time skew (clock skew)

1. Balanced delay clock driver

2. Use buffers where necessary

3. Very careful simulation (HSPICE)

4. Very small rise and fall time on the clock -- large buffer for large load

5. Multiple clocking strategies
Some Implementations of clocked latches

• Use a weak trickle inverter
  
  ![Weak Trickle Inverter Diagram]

  -ck
  clk
  D
  Q

  small inverter (low-gain, smaller W or large L)

  ➔ eliminate a metal connection
  ➔ smaller area

• Transmission-gate latch
  
  ![Transmission-Gate Latch Diagram]

  buffered input

  compared with a tri-state buffer
Typical symbolic layouts for latches

(a)  
(b)  
(c)
logic gate based latches

(a) Level sensitive

(b) Edge triggered
Asynchronously settable and resettable F/Fs
Dynamic single clock latches

- The feedback inverter and transmission gate are eliminated.
- The latched value is stored on the capacitance of the input to the inverter (mainly gate capacitance)
- Clock-to-Q ($T_q$) is very small need to be very careful to prevent transparency problem.
- Internal inversion of the clock is often necessary.
- Dynamic nodes should be always refreshed or clamped to a known state when in stand-by or low-power mode.

![Diagram](image.png)
Dynamic single clock latches (con’t)

(c) Tristate inverter

(d) Master-slave F/F

(e)
Refreshing for Dynamic latches

- Dynamic storage nodes are usually a gate capacitance.
- Assume the leakage current = 1 nA and the storage capacitance = 0.02PF.

Even if the storage of the correct state is unimportant, the leakage may cause the storage node to assume a level that causes the inverter to draw significant current.

\[ C \times \frac{\Delta V}{\Delta i} = 0.02 \times 10^{-12} \times \frac{5}{10^{-9}} = 100 \mu s \]

\[ 5V \rightarrow 2.5V \]

\[ \text{large current} \]
Phase locked loop (PLL) clock techniques

1. To synchronize internal and external clocks.
2. To synchronize data transfers between chips.
3. To operate the internal clock at a higher rate.

(a) Without PLL

(a) With PLL
PLL techniques

To ensure the output of chips are synchronized with each other.

Clock rate at $d_{clk}$

\[
\frac{4}{\text{clock}} \Rightarrow d_{clk} = 4 \times \text{clock}
\]
Block diagram of a PLL circuit

Phase detector: detect the difference between $F_{fb}$ and $F_{in}$.
- If $F_{fb} > F_{in}$ => D pulse
- If $F_{fb} > F_{in}$ => U pulse

Charge pump: charge or Discharge a capacitor according to D and U.

Filter: filter the capacitor output (smoother).

VCO: Change the oscillation frequency depending on the control voltage.
(Voltage Control Oscillator)
Phase Detector

If F1 falls before F2
=> UP=1
If F2 falls before F1
=> DN=1
Charge Pump

Bias circuit

CAP charges when CHGUP=1
discharge when CHGDN=1
Metastability Problem

If the setup or hold time is not satisfied, i.e., D changes at the activation edge of the clock, then the output Q will have a state depending on the timing relation between D and CLK.
Metastable state in a pair of inverters

To Solve the metastability problem:

- Setup time is shorter than the clock-to-Q delays in a synchronization system.
- For asynchronous input: need a special circuit called synchronizer.
Single-phase N-P CMOS dynamic logic

- Combine N-P section of domino logic with clocked CMOS ($C^2$MOS) latch as the output stage.

(a) Inputs from -clk stages

From n or buffered p-logic
n-p CMOS clk logic stage

(b) n-p CMOS -clk logic stage

From n or buffered p-logic
p-logic block

From n or buffered p-logic
n-logic block

(c) Evaluation

Precharge

Precharge

Evaluation

Precharge

$0 \quad 1$

$1 \quad 0$
Design rules for N-P CMOS dynamic logic

Two problems to be solved

1. Each section must be internally race free.
2. When different sections are cascaded from pipelined system, clock skew should not cause a problem.

R1: During precharge, logic-blocks must be switched off.

R2: During evaluation, internal inputs can make only one transition.

When a static logic is used in a N-P CMOS dynamic logic, it should be placed after dynamic logic (I.e., one should keep the static logic up to the C²MOS latch.

Reason: static logic after creates a glitch at its output.
R3: There exists in each logic block at least one dynamic gate that is separated from the previous C\(^2\)MOS output stage by an even number of inventions.

or

R4: The total number of inversions between two consecutive C\(^2\)MOS stage is even.

Reason:

The same evaluation phase in a section
Two phase clocking

(a) phi₁ = 1
    phi₂ = 0

(b) phi₁ = 0
    phi₂ = 1

(c) small delay

(d) slow rise time

phi₁ = 1
phi₂ = 0
Two phase clock generation

1. Globally distribute two clocks with or without their complements.
3. A single global clock and locally generated two-phase clocks

Two-phase clock generator:

```
clk -> \phi_1

\phi_2
```

Delay for non-overlap period
Two phase registers

Both of these dynamic registers have to drive a local storage gate.

(a) high level \(=V_{DD} - V_{tn}\)

(b) p leakers

(c) clk
Two phase logic

1. Static logic with two phase registers

2. Dynamic logic

-phi₁
phi₁
-phi₂
phi₂

phi₁
n-logic

from phi₂ stage to phi₁ stage

phi₂
n-logic

phi₁
phi₂

phi₁
precharge phi₁ logic
latch phi₂ data
evaluate phi₁ logic
precharge phi₁ logic
latch phi₂ data
evaluate phi₂ logic
precharge phi₂ logic
latch phi₁ data
Four-Phase clock

nonoverlapping

Four-Phase logic clocking method
Clock distribution

1. A single large buffer
2. A distributed-clock-tree approach

Clock delays have to match between stages
Trends in clock strategy

• For first-time designer, use static logic, single-phase static registers.
• For standard cell and gate-array design, single-phase may be the only choice.
• Two phase clocking make timing design of RAMs, ROMs and PLAs easier.
• In modern process and circuits, cycle time is the main concern => single phased Processes are extremely dense => single phase