Dynamic latches

Dynamic Transmission-Gate Based Edge-triggered Registers

A fully dynamic positive edge-triggered register based on the master-slave concept is shown in figure. When CLK = 0, the input data is sampled on storage node 1, which has an equivalent capacitance of C1 consisting of the gate capacitance of I1, the junction capacitance of T1, and the overlap gate capacitance of T1.

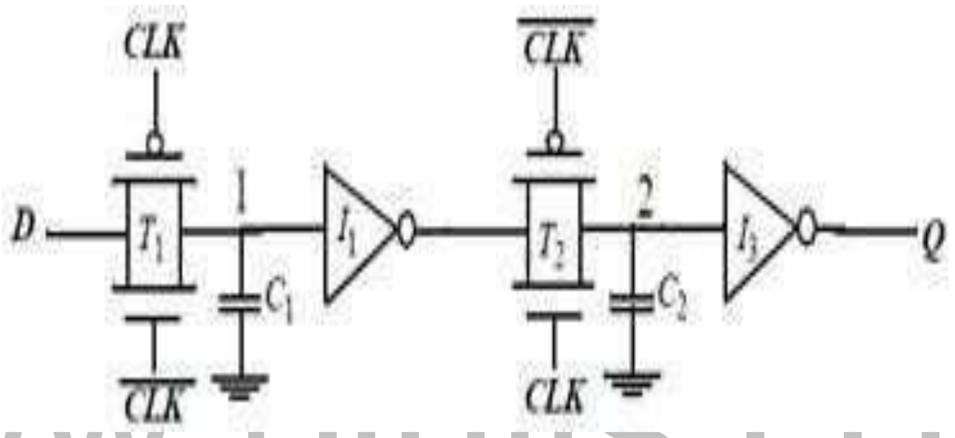


Figure 3.3.1: Dynamic Edge-triggered Registers

[Source: Jan M. Rabies, Anantha Chandrakasan, Borivoje. Nickelic, ‖Digital Integrated Circuits: A Design perspective …]

During this period, the slave stage is in a hold mode, with node 2 in a high-impedance (floating) state. On the rising edge of clock, the transmission gate T2 turns on, and the value sampled on node 1 right before the rising edge propagates to the output Q (note that node 1 is stable during the high phase of the clock since the first transmission gate is turned off).

Node 2 now stores the inverted version of node 1. This implementation of an edge- triggered register is very efficient as it requires only 8 transistors.

The sampling switches can be implemented using NMOS-only pass transistors, resulting in an even-simpler 6 transistor implementation. The reduced transistor count is attractive for high-performance and low-power systems.

The set-up time of this circuit is simply the delay of the transmission gate , and corresponds to the time it takes node 1 to sample the D input.The hold time is approximately zero, since the transmission gate is turned off on the clock edge and further inputs changes are ignored. The propagation delay (tc-q) is equal to two inverter delays plus the delay of the transmission gate T2.

One important consideration for such a dynamic register is that the storage nodes (i.e., the state) has to be refreshed at periodic intervals to prevent a loss due to charge leakage, due to diode leakage as well as sub-threshold currents. In datapath circuits, the refresh rate is not an issue since the registers are periodically clocked, and the storage nodes are constantly updated.

Clock overlap is an important concern for this register. Consider the clock waveforms shown in below figure. During the 0-0 overlap period, the NMOS of T1 and the PMOS of T2 are simultaneously on, creating a direct path for data to flow from the D input of the register to the Q output. This is known as race condition. The output Q can change on the falling edge if the overlap period is large

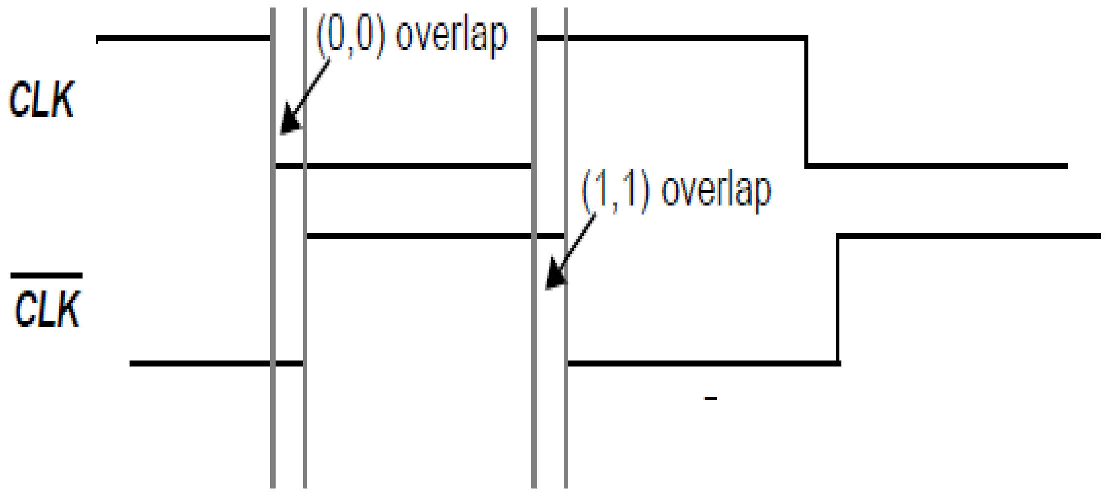


Figure 3.3.2: Impact of non-overlapping clocks

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design …]

CMOS Dynamic Register: A Clock Skew Insensitive Approach the CMOS Register

The following shows an ingenious positive edge-triggered register based on the master- slave concept which is insensitive to clock overlap. This circuit is called the C2MOS (Clocked CMOS) register. The register operates in two phases.

1. CLK = 0 (CLK = 1):

2. The roles are reversed when CLK = 1:

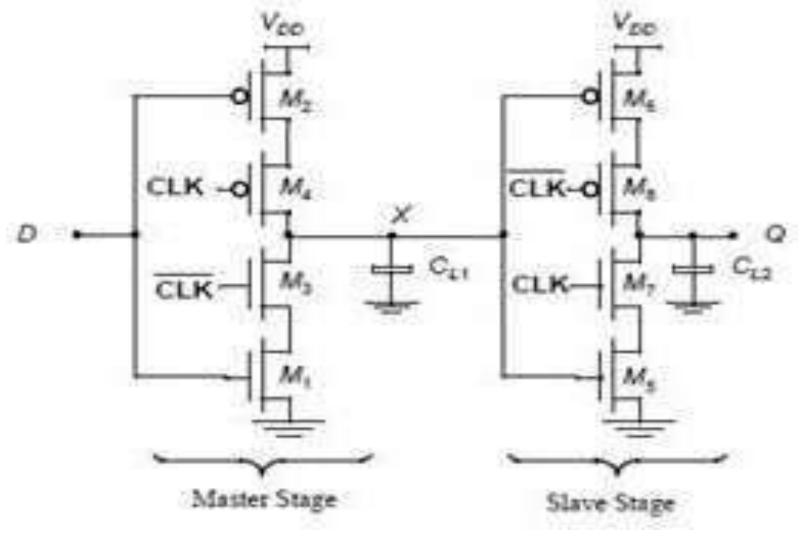


Figure 3.3.3: master slave edge-triggered register

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design …]

It can be stated that the C2MOS latch is insensitive to clock overlaps because those overlaps activate either the pull-up or the pull-down networks of the latches, but never both of them simultaneously. If the rise and fall times of the clock are sufficiently slow, however, there exists a time slot where both the NMOS and PMOS transistors are conducting. This creates a path between input and output that can destroy the state of the circuit.

Dual-edge Triggered Registers

So far, we have focused on edge-triggered registers that sample the input data on only one of the clock edges (rising or falling). It is also possible to design sequential circuits that sample the input on both edges. The advantage of this scheme is that a lower frequency clock (half of the original rate) is distributed for the same functional throughput, resulting in power savings in the clock distribution network.

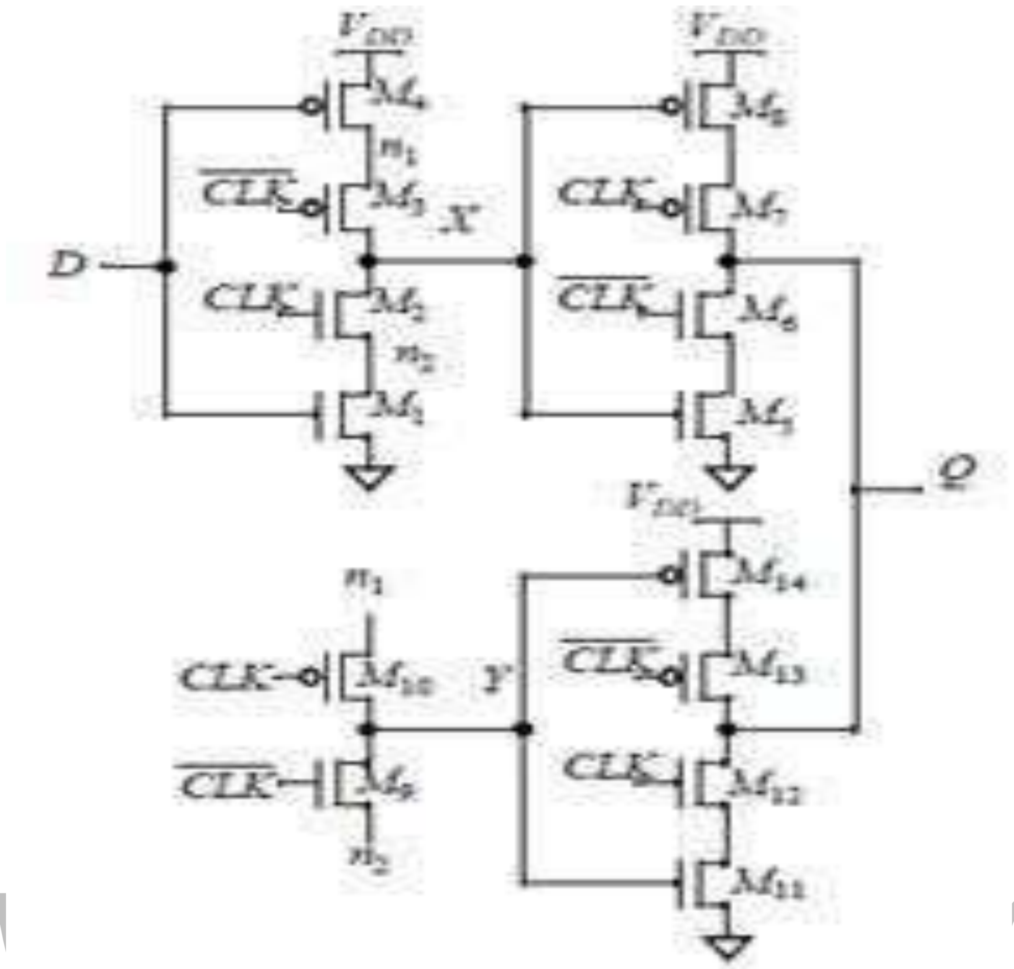


Figure 3.3.4: CMOS based dual-edge triggered register.

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design …]

…

The above figure shows a modification of the C2MOS register to enable sampling on both edges.

True Single-Phase Clocked Register (TSPCR)

In the two-phase clocking schemes described above, care must be taken in routing the two clock signals to ensure that overlap is minimized. While the C2MOS provides a skew- tolerant solution, it is possible to design registers that only use a single phase clock. The basic single-phase positive and negative latches are shown in figure.

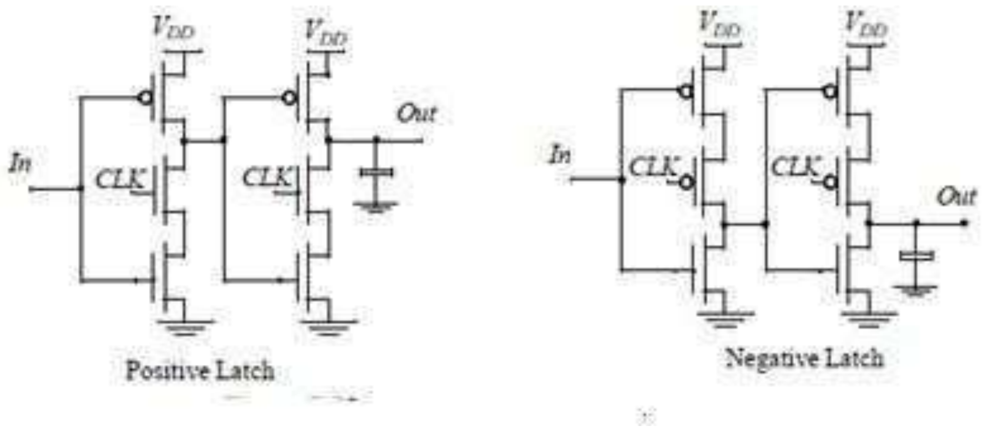


Fig 3.3.5: True Single-Phase Clocked Register

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design …]

For the positive latch, when CLK is high, the latch is in the transparent mode and corresponds to two inverters; the latch is non-inverting, and propagates the input to the output. On the other hand, when C LK = 0, both inverters are disabled, and the latch is in hold- mode. Only the pull-up networks are still active, while the pull-down circuits are deactivated. As a result of the dual-stage approach, no signal can ever propagate from the input of the latch to the output in this mode. A register can be constructed by positive and negative latches.

Synchronous Timing:

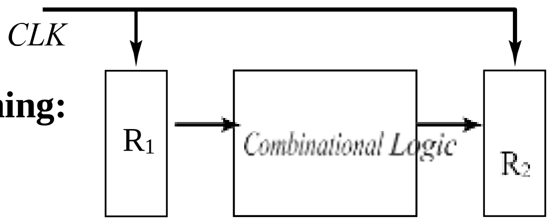


Fig 3.3.6: Synchronous Timing

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design …]

The following timing parameters characterize the timing of the sequential circuit.

- The contamination (minimum) delay t_{c-q}, cd, and maximum propagation delay of the register t_{c-q}.
- The set-up (t_{us}) and hold time (t_{hold}) for the registers.

- The contamination delay t_{Logic}, cd and maximum delay t_{logic} of the combinational logic.
- t_{Clk1} and t_{clk2}, corresponding to the position of the rising edge of the clock relative to a global reference.

Clock Non idealities:

Clock skew

✓ Spatial variation in temporally equivalent clock edges; deterministic + random, t_{SK}

Clock jitter

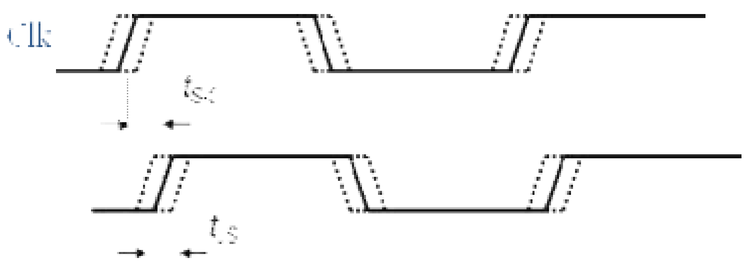
✓ Temporal variations in consecutive edges of the clock signal; modulation + random noise

✓ Cycle-to-cycle (short-term) t_{JS} Long term t_{JL}

Variation of the pulse width

✓ Important for level sensitive clocking

Clock Skew and Jitter:



✓ Both skew and jitter affect the effective cycle time

✓ Only skew affects the race margin

Sources of Skew and Jitter:

- **Clock-Signal Generation-** The generation of the clock signal itself causes jitter
- **Manufacturing Device Variations**
- **Interconnect Variations-**One important source of interconnect variation is the Inter-level Dielectric (ILD) thickness variations.
- **Environmental Variations-**Environmental variations are probably the most significant and primarily contribute to skew and jitter. The two major sources of environmental variations are temperature and power supply. Power supply variations is the major source of jitter in clock distribution networks.
- **Capacitive Coupling-**The variation in capacitive load also contributes to timing uncertainty. There are two major sources of capacitive load variations: coupling

Between the clock lines and adjacent signal wires and variation in gate capacitance.

Clock-Distribution Techniques:

It is necessary to design a clock network that minimizes skew and jitter. Another important consideration in clock distribution is the power dissipation.

Fabrics for clocking-one common approach to distributing a clock are to use balanced paths (or called trees). The most common type of clock primitive is the H-tree network (named for the physical structure of the network).

In this scheme, the clock is routed to a central point on the chip and balanced paths, that include both matched interconnect as well as buffers, are used to distribute the reference to various leaf nodes. Ideally, if each path is balanced, the clock skew is zero.

- Variation in gate capacitance.

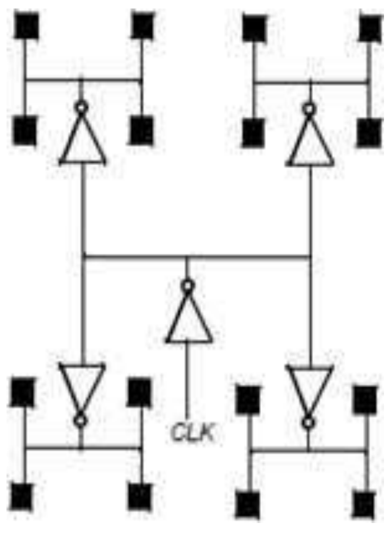


Fig 3.3.7: Example of H-tree clock distribution

[Source: Jan M. rabies, Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits: A Design perspective …]

That is, though it might take multiple clock cycles for a signal to propagate from the central point to each leaf node, the arrival times are equal at every leaf node.

Pipelining

Pipelining is a popular design technique often used to accelerate the operation of the data paths in digital processors. The idea is easily explained with the example of below figure .The goal of the presented circuit is to compute log (|a - b|), where both a and b represent streams of numbers, that is, the computation must be performed on a large set of input values.The minimal clock period TMin necessary to ensure correct evaluation is given as:

$$T_{min} = t_{c\text{-}q} + t_{pd,logic} + t_{su}$$

Where $t_{c\text{-}q}$ and t_{us} are the propagation delay and the set-up time of the register, respectively.

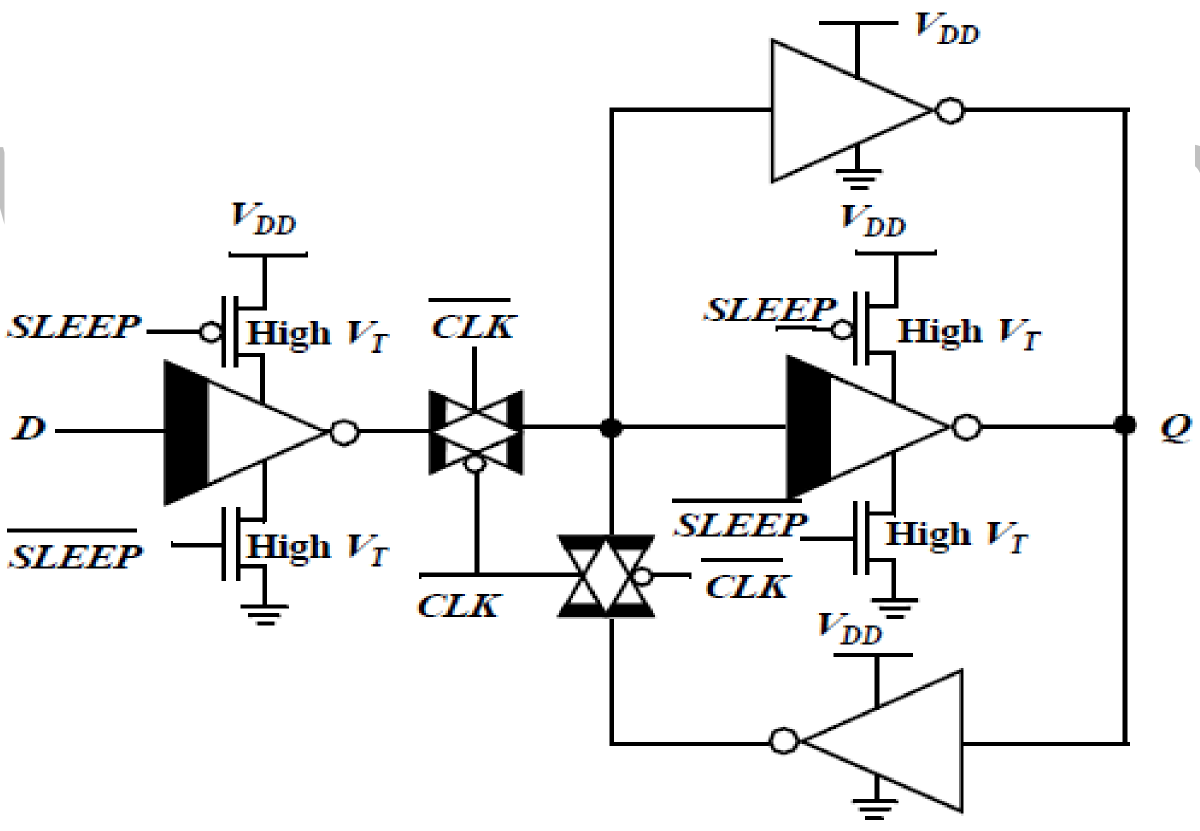


Fig 3.4.1: Data path for computation [Source: Sung-Mo kanga, Yusuf leblebici, Charlwood Kim —CMOS Digital Integrated Circuits: Analysis & Design …]

We assume that the registers are edge-triggered D registers. The term tpd, logic stands

For the worst-case delay path through the combinatorial network, this consists of the adder, absolute value, and logarithm functions. In conventional systems (that don't push the edge of technology), the latter delay is generally much larger than the delays associated with the registers and dominates the circuit performance. Assume that each logic module has an equal propagation delay. We note that each logic module is then active for only 1/3 of the clock period (if the delay of the register is ignored).

For example, the adder unit is active during the first third of the period and remains idle— this is, it does no useful computation— during the other 2/3 of the period. Pipelining is a technique to improve the resource utilization, and increase the functional throughput. Assume that we introduce registers between the logic blocks.

The result for the data set (a1, b1) only appears at the output after three clock- periods. At that time, the circuit has already performed parts of the computations for the next data sets, (a2, b2) and (a3, b3). The computation is performed in an assembly-line fashion, hence the name pipeline.

The advantage of pipelined operation becomes apparent when examining the minimum clock period of the modified circuit. The combinational circuit block has been partitioned into three sections, each of which has a smaller propagation delay than the original function. This effectively reduces the value of the minimum allowable clock period.

Latch- vs. Register-Based Pipelines

Pipelined circuits can be constructed using level-sensitive latches instead of edge-triggered registers. Consider the pipelined circuit of below figure. The pipeline system is implemented based on pass-transistor-based positive and negative latches instead of edge triggered registers. That is, logic is introduced between the master and slave latches of a Master- Slave system.

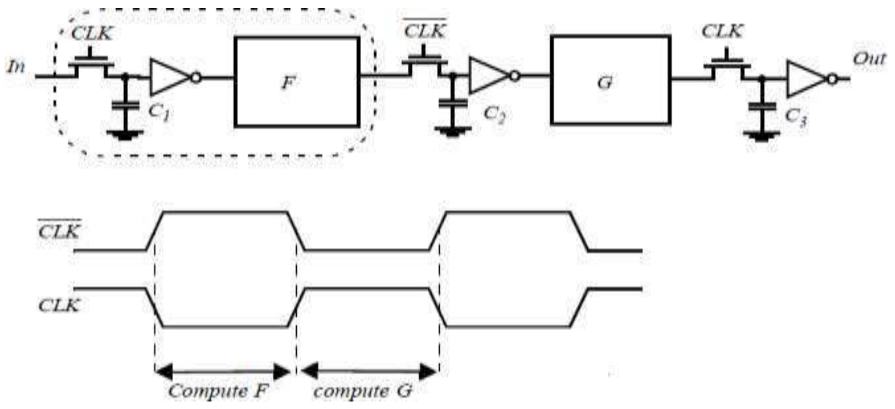


Fig 3.4.2: Operation of two-phase pipelined circuit using dynamic registers

[Source: Sung-Mo kanga, Yusuf leblebici, Charlwood Kim —CMOS Digital Integrated Circuits: Analysis & Design …]

In the following discussion, we use without loss of generality the CLK-CLK notation to denote a two-phase clock system. Latch-based systems give significantly more flexibility in implementing a pipelined system, and often offer higher performance.

When the clocks CLK and CLK are non-overlapping, correct pipeline operation is obtained. Input data is sampled on C1 at the negative edge of CLK and the computation of logic block F starts; the result of the logic block F is stored on C2 on the falling edge of CLK, and the computation of logic block G starts. The non-overlapping of the clocks ensures correct operation. The value stored on C2 at the end of the CLK low phase is the result of passing the previous input (stored on the falling edge of C LK on C1) through the logic function F.

When overlap exists between CLK and CLK, the next input is already being applied to F, and its effect might propagate to C2 before CLK goes low (assuming that the contamination delay of F is small). Which value wins depends upon the logic function F, the overlap time, and the value of the inputs since the propagation delay is often a function of the applied inputs.

NORA-CMOS-A Logic Style for Pipelined Structures

The latch-based pipeline circuit can also be implemented using CMOS latches, as shown in Figure. The operation is similar to the one discussed above. This topology has one additional, important property:

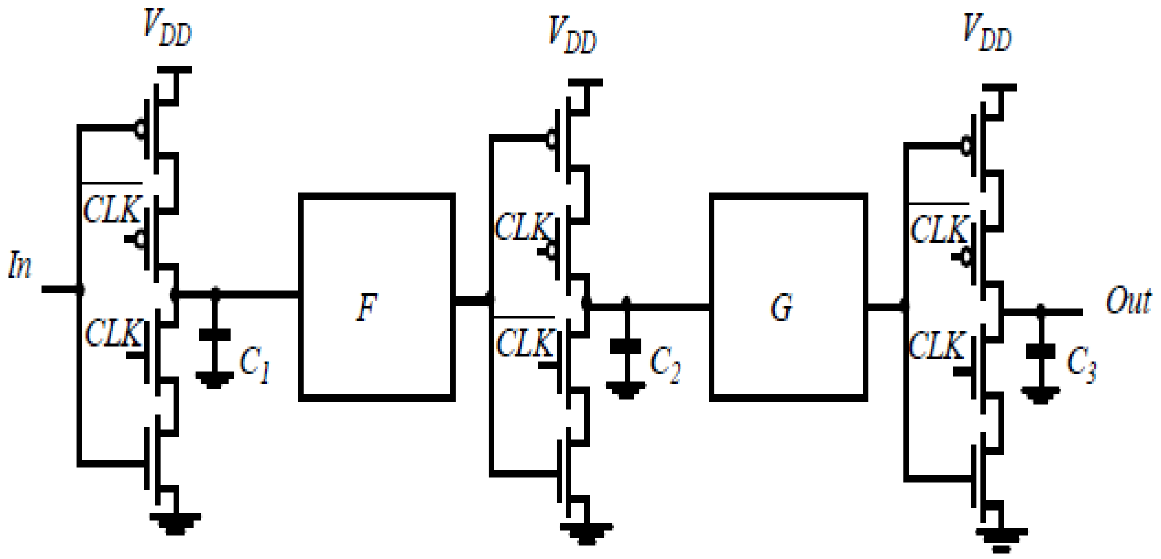


Fig 3.4.3: Pipeline data path using CMOS latches.

[Source: Sung-Mo kang, Yusuf leblebici, Chulwoo Kim ―CMOS Digital Integrated Circuits: Analysis & Design …]

A CMOS-based pipelined circuit is race-free as long as all the logic functions F (implemented using static logic) between the latches are non-inverting. The reasoning for the above argument is similar to the argument made in the construction of a C2MOS register. During a (0-0) overlap between CLK and CLK, all C2MOS latches, simplify to pure pull-up networks. The only way a signal can race from stage to stage under this condition is when the logic function F is inverting, as illustrated in below figure, where F is replaced by a single, static CMOS inverter.

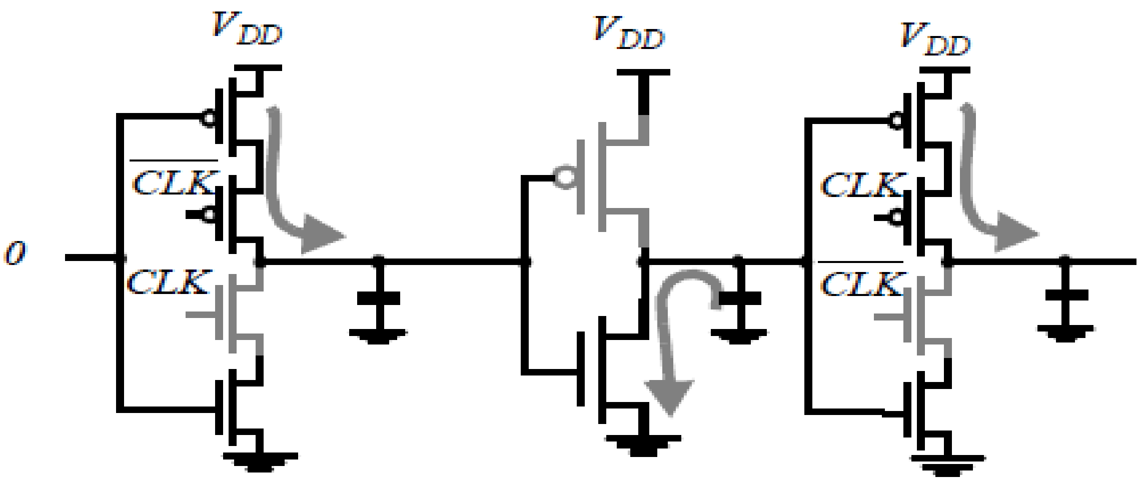


Fig 3.4.4: Potential race condition during (0-0) overlap in CMOS-based design.

[Source: Sung-Mo kang, Yusuf leblebici, Charlwood Kim —CMOS Digital Integrated Circuits: Analysis & Design …]

Similar considerations are valid for the (1-1) overlap. Based on this concept, a logic circuit style called NORA-CMOS; it combines CMOS pipeline registers and NORA dynamic logic function blocks. Each module consists of a block of combinational logic that can be a mixture of static and dynamic logic, followed by a CMOS latch. Logic and latch are clocked in such a way that both are simultaneously in either evaluation, or hold (pre charge) mode. A block that is in evaluation during CLK = 1 is called a CLK-module, while the inverse is called a CLK- module.

A NORA Data path consists of a chain of alternating CLK and CLK modules. While one class of modules is recharging with its output latch in hold mode, preserving the previous output value, the other class is evaluating. Data is passed in a pipelined fashion from module to module .NORA offers designers a wide range of design choices. Dynamic and static logic can be mixed freely, and both Clips and Cline dynamic blocks can be used in pipelined form. With this freedom of design, extra inverter stages, as required in DOMINO-CMOS, are most often avoided in order to ensure correct operation, two important rules should always be followed:

- **The dynamic-logic rule:** Inputs to a dynamic Cline (Clips) block are only allowed

To make a single (1 0) transition during the evaluation period.

- **The CMOS rule:** In order to avoid races, the number of static inversions between CMOS latches should be even

Definition:

In Synchronous sequential circuit, the output depends on present and previous states of the inputs at the clocked instances. The circuits use a memory element to store the previous state. The memory elements in these circuits will have clocks. All these clock signals are driven by the same clock signal.

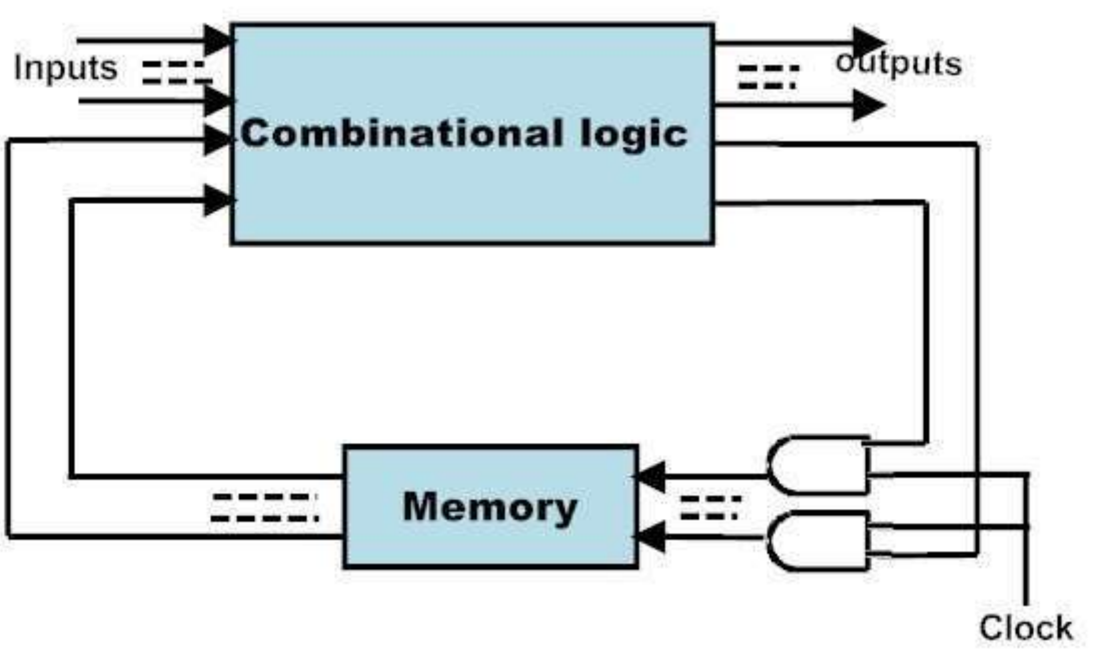


Figure 3.1.1: Synchronous Sequential Circuits

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

- Using clock signal, state changes will occur across all storage elements.
- These circuits are bit slower compared to asynchronous because they wait for the next clock pulse to arrive to perform the next operation.
- These circuits can be clocked or pulsed.
- The Synchronous sequential circuits that use clock pulses in their inputs are called clocked-sequential circuits. They are very stable.
- The sequential circuits that change their state using the pulse and these are called pulsed or un-clocked sequential circuits.

Where we use Synchronous Sequential Circuits??

• Used in the design of MOORE-MEALY state management machines.

• They are used in synchronous counters, flip flops etc.

Limitations of Synchronous Sequential Circuits

- All the flip – flops in synchronous sequential circuits must be connected to clock signal. Clock signals are very high frequency signals and clock distribution consumes and dissipated a large amount of heat.
- Critical path or the slowest path determines the maximum possible clock frequency. Hence they are slower than asynchronous circuits.

Asynchronous Sequential Circuits

Definition

The Sequential circuits which do not operate by clock signals are called "Asynchronous sequential circuits".

- These circuits will change their state immediately when there is a change in the input signal .
- The Circuit behaviour is determined by signals at any instant in time and the order in which input signals change.

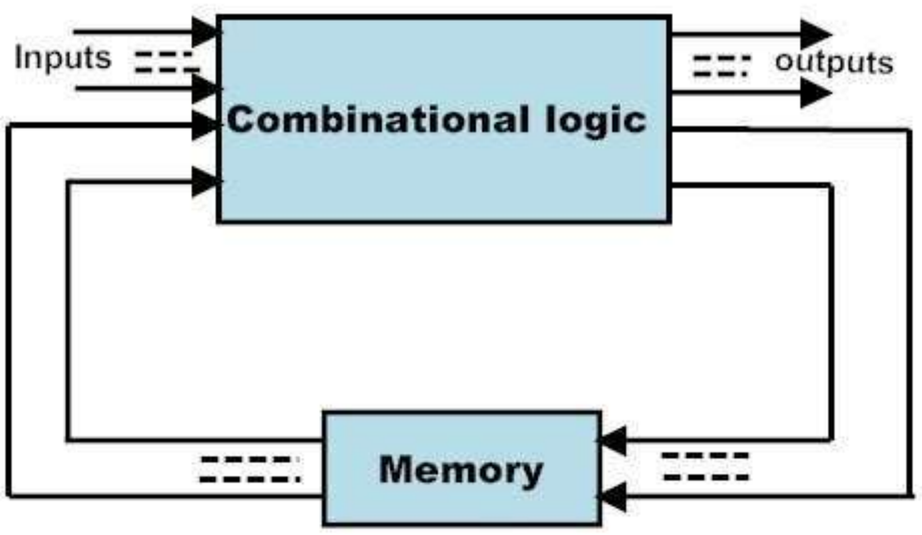


Figure 3.1.2: Asynchronous Sequential Circuits

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

- They do not operate in pulse mode.
- They have better performance but hard to design due to timing problems.
- Mostly we use the asynchronous circuits when we require the low power operations.
- They are faster than synchronous sequential circuits as they do not need to wait for any clock signal.

Where we use Asynchronous Sequential Circuits??

These are used when speed of operation is important. As they are independent of internal clock pulse, they are operate quickly. so they are used in Quick response circuits.

- Used in the communication between two units having their own independent clocks.
- Used when we require the better external input handling.

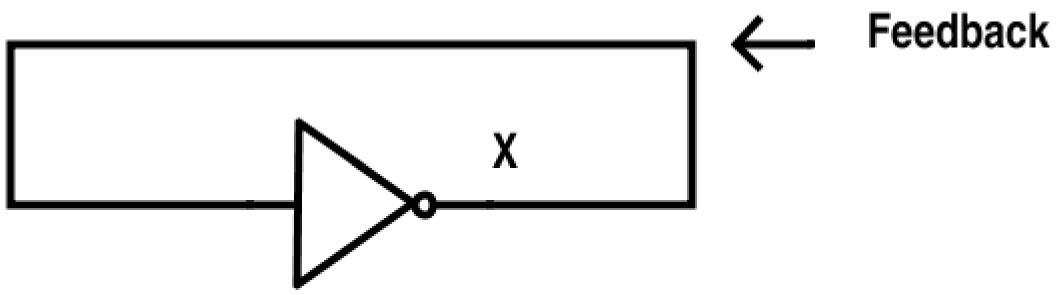
Limitations of Asynchronous Sequential Circuits

- Asynchronous sequential circuits are more difficult to design.
- Though they have a faster performance, their output is uncertain.

Back to top

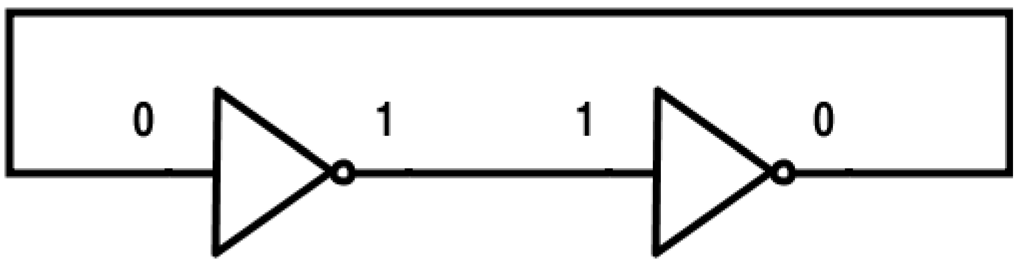
Feedback in Sequential Circuits

Combinational circuits do not require any feed back as the outputs are purely dependent on the present value of the input. But in case of sequential circuits, the outputs are dependent on past values of the input along with present values. In order to involve memory element like a flip – flop, feedback must be introduced in the circuit. For example, consider a simple feedback circuit as shown below.

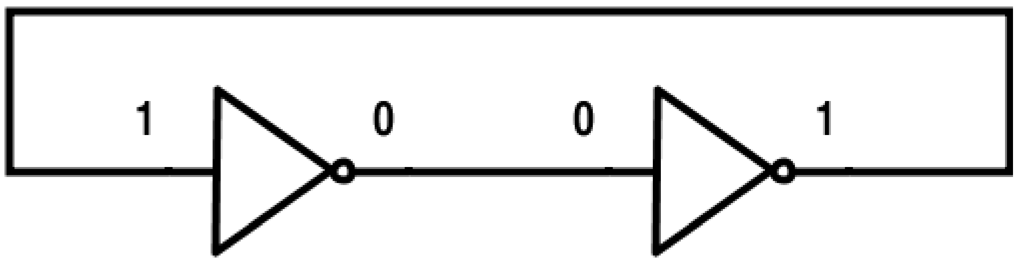


If 0 is the input to the inverter at an instance, this 0 will propagate and the output is 1. This 1 is fed back as input. This 1 will propagate and the output is 0. The process repeats and the result is a continuous oscillation of output between 0 and 1. There is no stable state in this scenario.

Now consider the following example of two inverters connected as shown.



Here two inverters are connected back to back with the output of the second inverter fed back to input of first inverter. If 0 is the input to first inverter at an instance, it propagates through the first inverter and the output is 1. This 1 is input to second inverter and propagates through it. The output of the second inverter is 0 which is fed back to the first inverter. But the input to first inverter is already 0 and hence no change occurs. The circuit is said to be in a stable circuit. Another stable state can be obtained when the input to the first inverter is 1.



Latches and Flip flops

Latch

Latch is a basic building element in sequential circuits. Latches do not have any clock signal , that is they are asynchronous sequential circuits.

• Latches are made up of static gates.

• Latch is a bistable multivibrator i.e. it has two stable states and can switch between these states.

• Latches will have a feedback path from the output. Thus they change their output at any instant using the previous and present states of the input signals.

• When enabled, the output of the latch is continuously affected by its input i.e. the output changes immediately when the input changes. When disabled, the state of the latch remains constant i.e. it remembers its previous value. A clock or enable signal is used as a control signal.

• Latches continuously check all the inputs and correspondingly it changes its output when enabled.

Example: S-R latch is an example for simple latch.

Example: S-R latch is an example for simple latch.

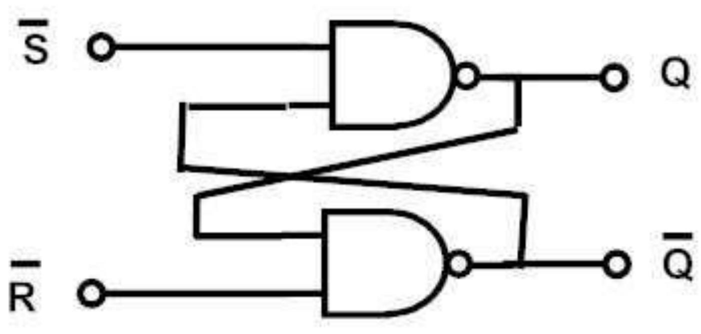


Figure 3.1.3: S-R Latch

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

Flip-Flop

The Flip-flop is also a building block of synchronous sequential circuits. It has two stable states. It can store one bit of information. Flip flops will have a clock signal. Their state changes depending on the clock pulse .These devices will is have two states and a feedback path .

- Flip-Flop is edge sensitive. They will change their state when the clock signal transition occurs from low to high or high to low.
- After the transition of clock signal from either 0 to 1 or 1 to 0 i.e. when the clock is at constant 0 or 1, the state remains unchanged even if the input changes.

Example: J-K Flip-Flop.

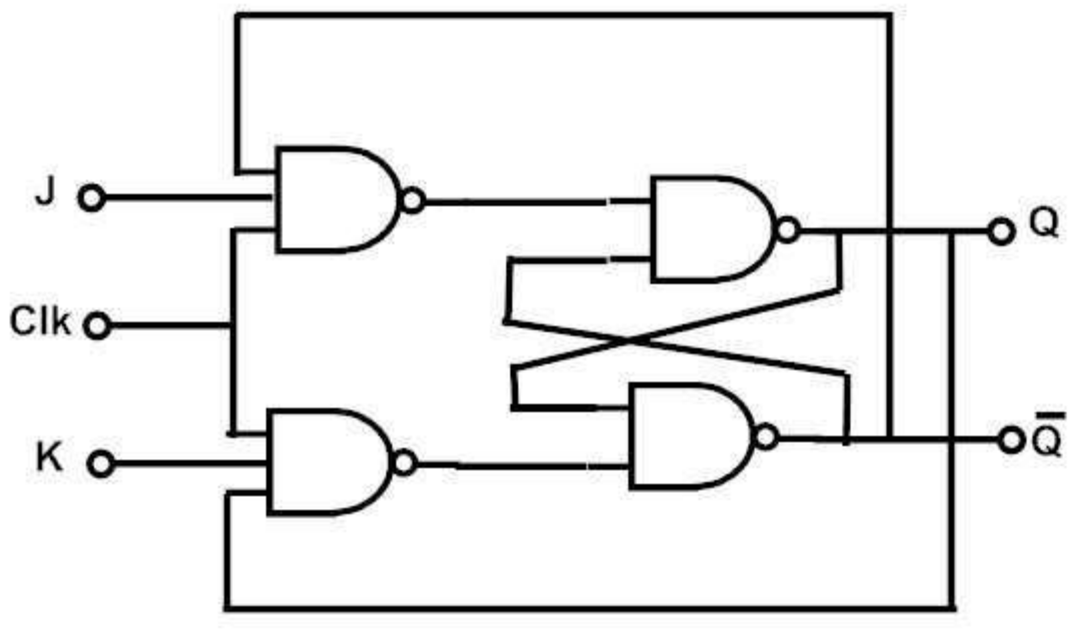


Figure 3.1.4: J-K Flip-Flop

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

NOTE: The only difference between latches and flip – flops is that a latch is level sensitive to control signal (clock or enable) while a flip – flop is edge sensitive to the control signal (usually clock).

Triggering

Definition

The change in output of a flip flop can be done by bringing a small change in the input signal. This small change can be done with the help of a clock pulse. This clock pulse is known as a Trigger pulse.

A flip – flop is said to be "Triggered", when a trigger pulse is applied to the input that brings changes in the output. Flip – flops are basic components in registers and counters, which store data in the form of multi – bit numbers. Number of flip – flops are connected to form a sequential circuit and all these flip – flops require trigger pulse. The number of trigger pulses applied to the input determines the number in a counter.

There are two type of triggering: Level Triggering and Edge Triggering

Level Triggering

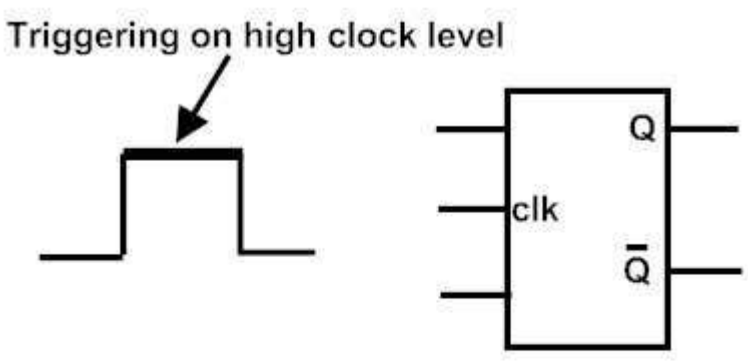
The triggering process in which the change in the output state is according to the active level of inputs is called "Level Triggering".

Level triggering is of two types, they are

1. High level triggering.
2. Low level triggering.

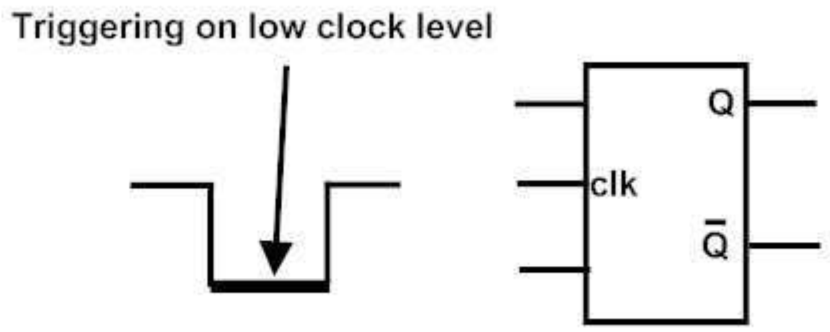
High Level Triggering

In High Level Triggering, the output of the flip – flop changes only when its enable input is at a high state i.e. logic high or logic 1. The symbolic representation of high level triggering is shown below.



Low Level Triggering

In Low Level Triggering, the output of the flip – flop changes only when its enable input is at a low state i.e. logic low or logic 0. The symbolic representation of low level triggering is shown below. A low level triggering is usually identified by the bubble on the clock input.



Edge Triggering

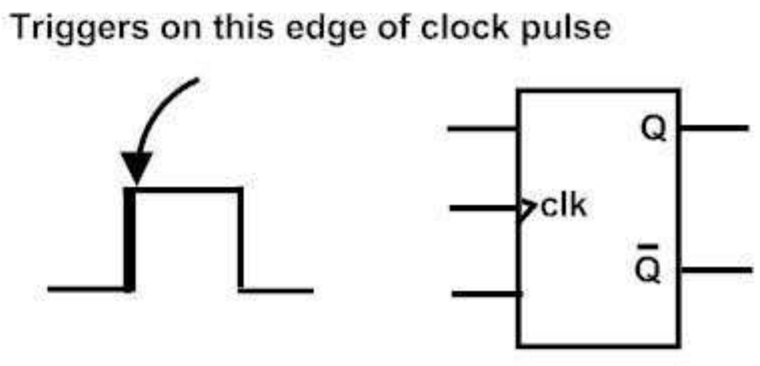
In Edge Triggering, the output changes only when the inputs are present at either of the transitions of the clock pulse i.e. either from low to high (0 to 1) or from high to low (1 to 0).

Edge triggering is of two types, they are

1. Positive edge triggering.
2. Negative edge triggering.

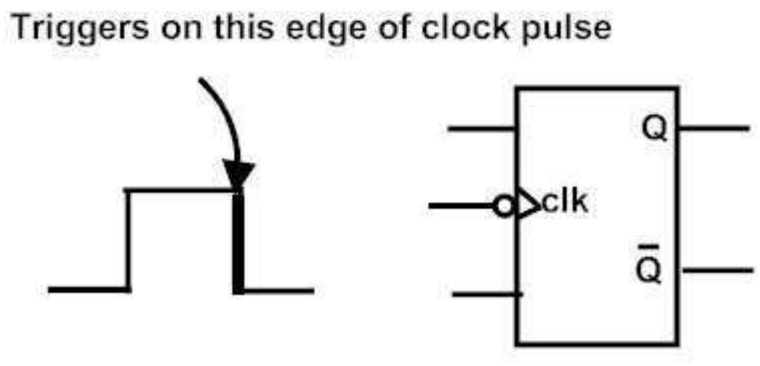
Positive Edge Triggering

In Positive Edge Triggering, the output changes only when the input is at the positive edge of the clock pulse input i.e. a transition from low to high (0 to 1). Positive Edge Triggering method is used when a flip-flop is required to respond at low to high level transition state. The symbolic representation of positive edge triggering is shown below.



Negative Edge Triggering

In Negative Edge Triggering, the output changes only when the input is at the negative edge of the clock pulse input i.e. a transition from high to low (1 to 0). Negative Edge Triggering method is used when a flip-flop is required to respond at high to low level transition state. The symbolic representation of negative edge triggering is shown below.



Edge Trigger Better Than Level Trigger

It is better to use edge triggering rather than level triggering. This is because level triggering might cause instability in the circuit for a particular case of a level triggered flip – flop, where the clock pulse is given to the input at the same time when the output of the flip – flop is changing. Feedback from the output to the input causes this instability. In order to avoid this instability, edge triggered flip – flops are used.

``

Static latchs

A latch is an essential component in the construction of an edge-triggered register. It is level-sensitive circuit that passes the D input to the Q output when the clock signal is high. This latch is said to be in transparent mode. When the clock is low, the input data sampled on the falling edge of the clock is held stable at the output for the entire phase, and the latch is in hold mode. A latch operating under the above conditions is a positive latch.

Static Latches Principle:

Static memories use positive feedback to create a bitable circuit — a circuit having two stable states that represent 0 and 1. The basic idea is shown in below figure which shows two inverters connected in case along with a voltage-transfer characteristic typical of such a circuit. Also plotted are the VTCs of the first inverter, that is o1 versus Vi1, and the second inverter (Vo2 versus Vo1).

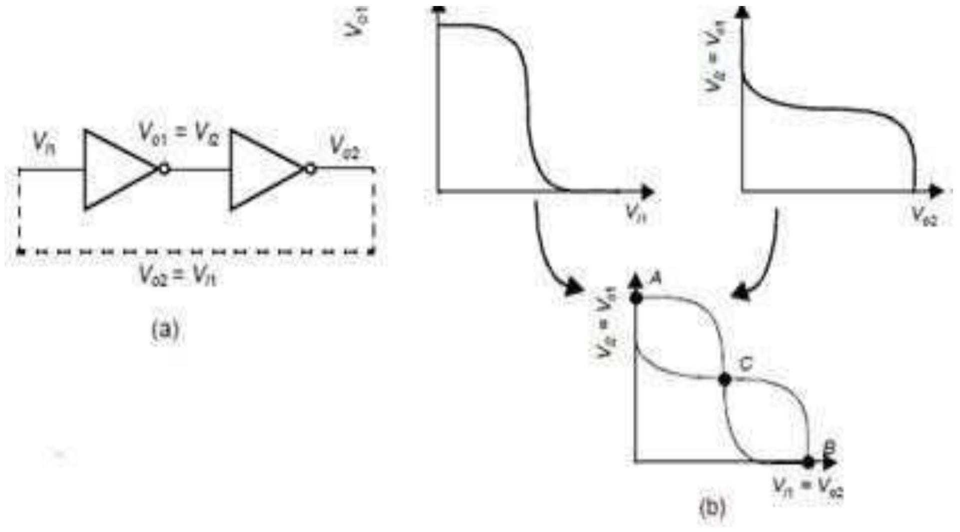


Fig 3.2.1: Example of H-tree clock distribution

[Source: Jan M. Rabaey, Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits: A Design perspective …]

The latter plot is rotated to accentuate thatVi2 = Vo1. Assume now that the

Output of the second inverter Vo2 is connected to the input of the first Vi1, as shown by the dotted lines in Figure 7.4a. The resulting circuit has only three possible operation points (A, B, and C), as demonstrated on the combined VTC.

Multiplexer Based Latches

There are many approaches for constructing latches. One very common technique involves the use of transmission gate multiplexers. Multiplexer based latches can provide similar functionality to the SR latch, but has the important added advantage that the sizing of devices only affects performance and is not critical to the functionality. Figure 7.11 shows an implementation of static positive and negative latches based on multiplexers.

For a negative latch, when the clock signal is low, the input 0 of the multiplexer is selected, and the D input is passed to the output. When the clock signal is high, the input 1 of the multiplexer, which connects to the output of the latch, is selected. The feedback holds the output stable while the clock signal is high. Similarly in the positive latch, the D input is selected when clock is high, and the output is held (using feedback) when clock is low.

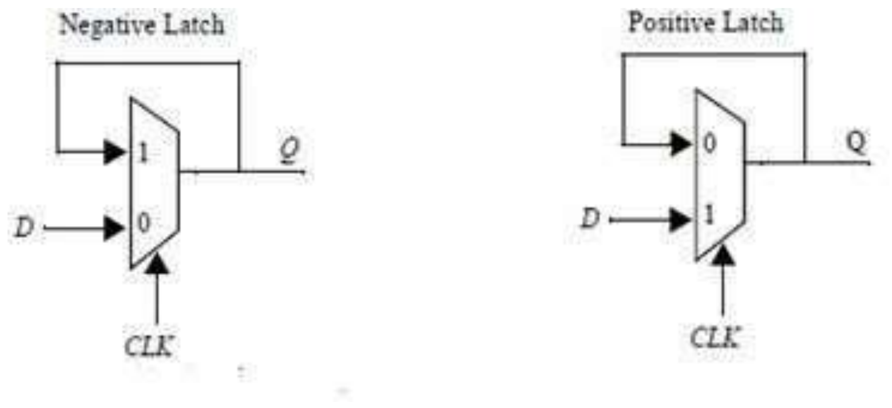


Fig 3.2.2: Negative and Positive latches based on multiplexer

[Source: Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits: A Design perspective …]

``

A transistor level implementation of a positive latch based on multiplexers is shown in figure. When CLK is high, the bottom transmission gate ison and the latch is transparent - that is, the D input is copied to the Q output. During this phase, the feedback loop is open since the top transmission gate is off.

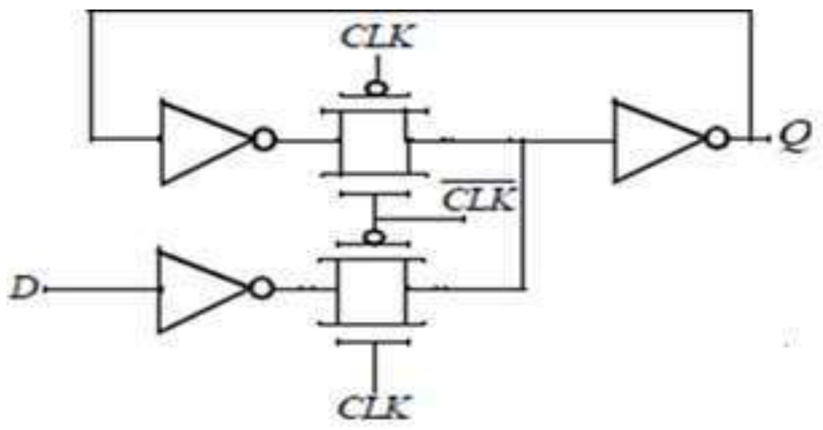


Fig 3.2.3: Transistor level implementation of a positive latch built using transmission gates

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

Unlike the SR FF, the feedback does not have to be overridden to write the memory and hence sizing of transistors is not critical for realizing correct functionality. The number of transistors that the clock touches is important since it has an activity factor of 1. This particular latch implementation is not particularly efficient from this metric as it presents a load of 4 transistors to the CLK signal.

It is possible to reduce the clock load to two transistors by using implement multiplexers using NMOS only pass transistor as shown in Figure 3.2.4 The advantage of this approach is the reduced clock load of only two NMOS devices. When CLK is high, the latch samples the D input, while a low clock-signal enables the feedback-loop, and puts the latch in the hold mode.

``

Low-Voltage Static Latches

The scaling of supply voltages is critical for low power operation. Unfortunately, certain latch structures don't function at reduced supply voltages. For example, without the scaling of device thresholds, NMOS only pass transistors don't scale well with supply voltage due to its inherent threshold drop. At very low power supply voltages, the input to the inverter cannot be raised above the switching threshold, resulting in incorrect evaluation. Even with the use of transmission gates, performance degrades significantly at reduced supply voltages.Scaling to low supply voltages hence requires the use of reduced threshold

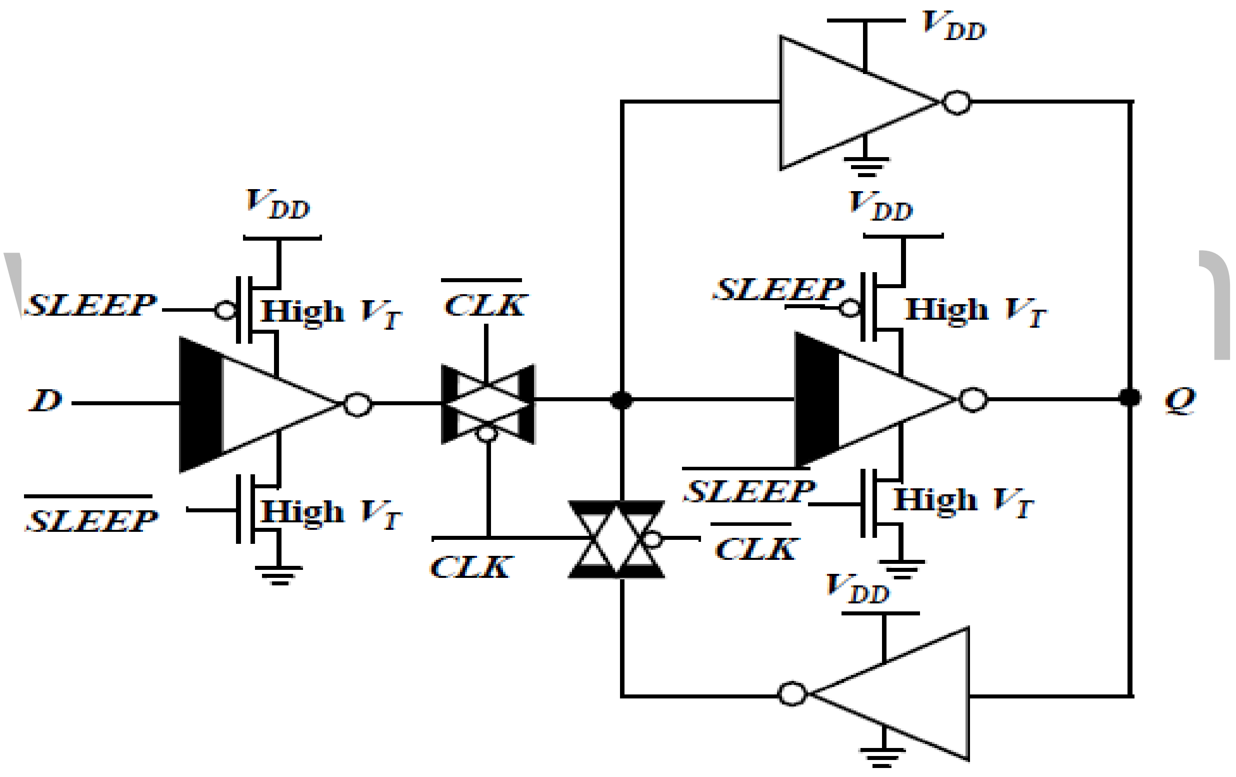


Fig 3.2.4: one solution for the leakage problem in low-voltage operation using MTCMOS

[Source : Jan M. Rabaey ,Anantha Chandrakasan, Borivoje. Nikolic, ‖Digital Integrated Circuits:A Design perspective …]

Devices. However, this has the negative effect of exponentially increasing

``

The sub- threshold leakage Power. Age energy is typically insignificant compared to the switching power. However, with the use of conditional clocks, it is possible that registers are idle for extended periods and the leakage energy expended by registers can be quite significant.

Many solutions are being explored to address the problem of high leakage during idle periods. One approach for this involves the use of Multiple Threshold devices as shown in above figure only the negative latch is shown here. The shaded inverters and transmission gates are implemented in low-threshold devices. The low threshold inverters are gated using high threshold devices to eliminate leakage.

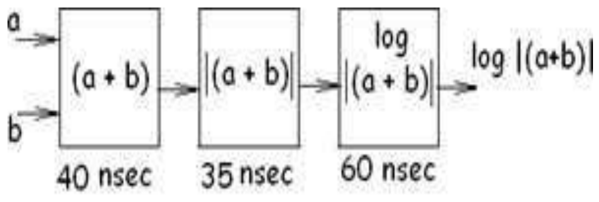
During normal mode of operation, the sleep devices are tuned on. When clock is low, the D input is sampled and propagates to the output. When clock is high, the latch is in the hold mode. The feedback transmission gate conducts and the cross-coupled feedback is enabled. Note there is an extra inverter, needed for storage of state when the latch is in the sleep state. During idle mode, the high threshold devices in series with the low threshold inverter are turned off (the SLEEP signal is high), eliminating leakage. It is assumed that clock is in the high state when the latch is in the sleep state. The feedback low-threshold transmission gate is turned on and the cross-coupled high-threshold device maintains the state of the latch.

TIMING ISSUES

Introduction:

While studying sequential circuits, we studied about Latches and Flip Flops. While Latches formed the heart of a Flip Flop, we have explored the use of Flip Flops in applications like counters, shift registers, sequence detectors, sequence generators and design of Finite State machines. Another important application of latches and flip flops is in pipelining combinational/algebraic operation. To understand what is pipelining consider the following example.

Let us take a simple calculation $C = \log(|a + b|)$ which has three operations to be performed viz. 1. Add a and b to get (nab), 2. Get magnitude of (Alba) and 3. Evaluate log | (a + b)|. Each operation would consume a finite period of time. Let us assume that each operation consumes 40 sec., 35 sec. And 60 sec. Respectively. The process can be represented pictorially



. Consider a situation when we need to carry this out for a set of 100 such pairs. In a normal course when we do it one by one it would take a total of 100 * 135 = 13,500 sec. We can however reduce this time by the realization that the whole process is a sequential process. Let the values to be evaluated be a_1 to a_{100} and the corresponding values to be added be b_1 to b_{100}.

Since the operations are sequential, we can first evaluate $(a_1 + b_1)$ while the value

$|(a_1 + b_1)|$ is being evaluated the unit evaluating the sum is dormant and

We can use it to evaluate $(a_2 + b_2)$ giving us both $|(a_1 + b_1)|$ and $(a_2 + b_2)$ at the end of another evaluation period.

Now at the end of the second evaluation period, in a similar fashion, we can initiate evaluation of log $|(a_1 + b_1)|$, $|(a_2 + b_2)|$ and $(a_3 + b_3)$ simultaneously and obtain the output log $|(a_1 + b_1)|$ at the end of the third evaluation period. To accommodate the operation with the longest delay of 60nsec. Each evaluation period will be of 60nsec. This assumes that before initiating each operation all operations should have been over. After the arrival of the first output data log $|(a_1 + b_1)|$.

The subsequent outputs log $|(a_2 + b_2)|$, log $|(a_3 + b_3)|$ will now start arriving at a gap of every 60 sec. In other words, the inputs to the system can be changed every 60 sec. i.e. all the 100 inputs can be applied in a time of 99*60 = 5940 sec. And the total time taken to evaluate 100 data will be 5940 + 180 = 6,120 sec. -- less than half of 13,500 sec needed earlier. This process of evaluation is called **Pipelining**. There is however a catch here, we need to ensure that the input to each block is held constant till the calculations are over, an impossible task if we stick to the configuration in Fig. 1.

A simple approach to ensure stability of input to each block is to insert a edge triggered flip flop register between each block with the registers clocked by a clock running at a period of at least 60 sec. with the new data being loaded every active clock edge. This is shown in Fig. 2.

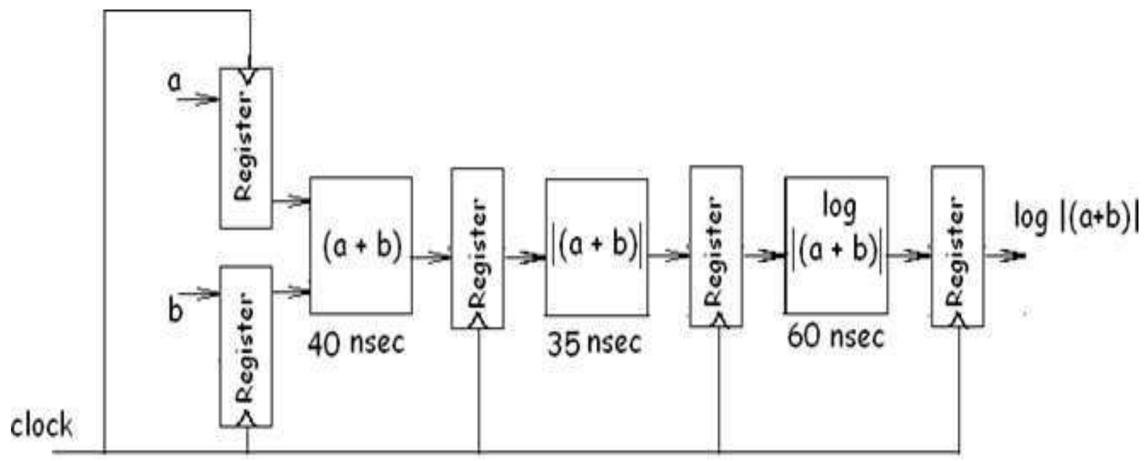


Fig 3.5.1: pipelining

[Source: Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits: Analysis & Design]

In the pipeline, we can use either an edge triggered D flip flop, a pulse triggered D flip flop or level triggered D latches. While using level triggered latches, the latches are clocked with clock and $\overline{clock}$ alternately. Let us now look in to pipeline system realized with three kinds of registers and the associated timing issues.

Timing Analysis of Pipeline Systems using Positive Edge Triggered Flip- Flops

The positive edge triggered flip-flops propagate the input value of the data to output node on the positive edge of the clock. A D flip-flop, as you may recall, has three terminals: Data input D_{in}, clock input CLK and data output Q_{uota}. When the clock input makes a low-high transition input D_{in} is propagated to output Q_{uota}. Let us briefly recollect some of the timings associated with flip flops and their importance.

1. **Cycle Time (T_{cycle})**

It is the amount of time elapsed between two successive positive edges of the clock. We denote this parameter by T_{cycle}.

2. **Clock-to-Out delay time (t$_{clock-out}$)**

It is the delay from the triggering edge of the clock to the new stable output on Q_{out} in both edge triggered and pulse triggered flip flops, assuming that D_{in} has been setup sufficiently early to triggering edge of the clock. This parameter is denoted as $t_{clock-out}$. In a pulse triggered flip flop, the triggering edge is defined as the trailing edge of the pulse in a positive pulse triggered flip flop and rising edge of the pulse n a negative pulse triggered flip flop.

3. **Data-stable-toout delay (t$_{data-out}$)**

It is the delay between the instant when the data input D_{in} stabilizes to the instant when the output Q_{out} stabilizes. As opposed to edge triggered flip flops wherein $t_{clock-out}$ is of interest to us, for level triggered flip flop $t_{data-out}$ is of interest. In the case of edge triggered flip flops, the output Q_{out} changes at the clock edge when the slave latch transmits the master output to the flip flop output while in the case of level and pulse triggered latches, at the active clock level input D_{in} is transferred to the output Q_{out}. In most cases, for a given technology, $t_{clock-out}$ is marginally larger than $t_{data-out}$.

4. **Setup Time (t$_{setup}$)**

It is defined as the minimum time between a Data (D_{in}) change and

The triggering edge of the clock, also referred to as data-to-clock delay, such that the output Q $_{out}$ will be guaranteed to change so as to become equal to the new value of D_{in}. We denote this parameter as s_{etup}. The setup time is defined such that, any change in Data at D_{in} arriving earlier than the Setup Time, s_{etup}, should not result in any changes in the clock-to-out delay time. In other words, data signal on D_{in} should occur sufficiently early so that making it appear any earlier would have no effect on when the output Q_{uota} changes.

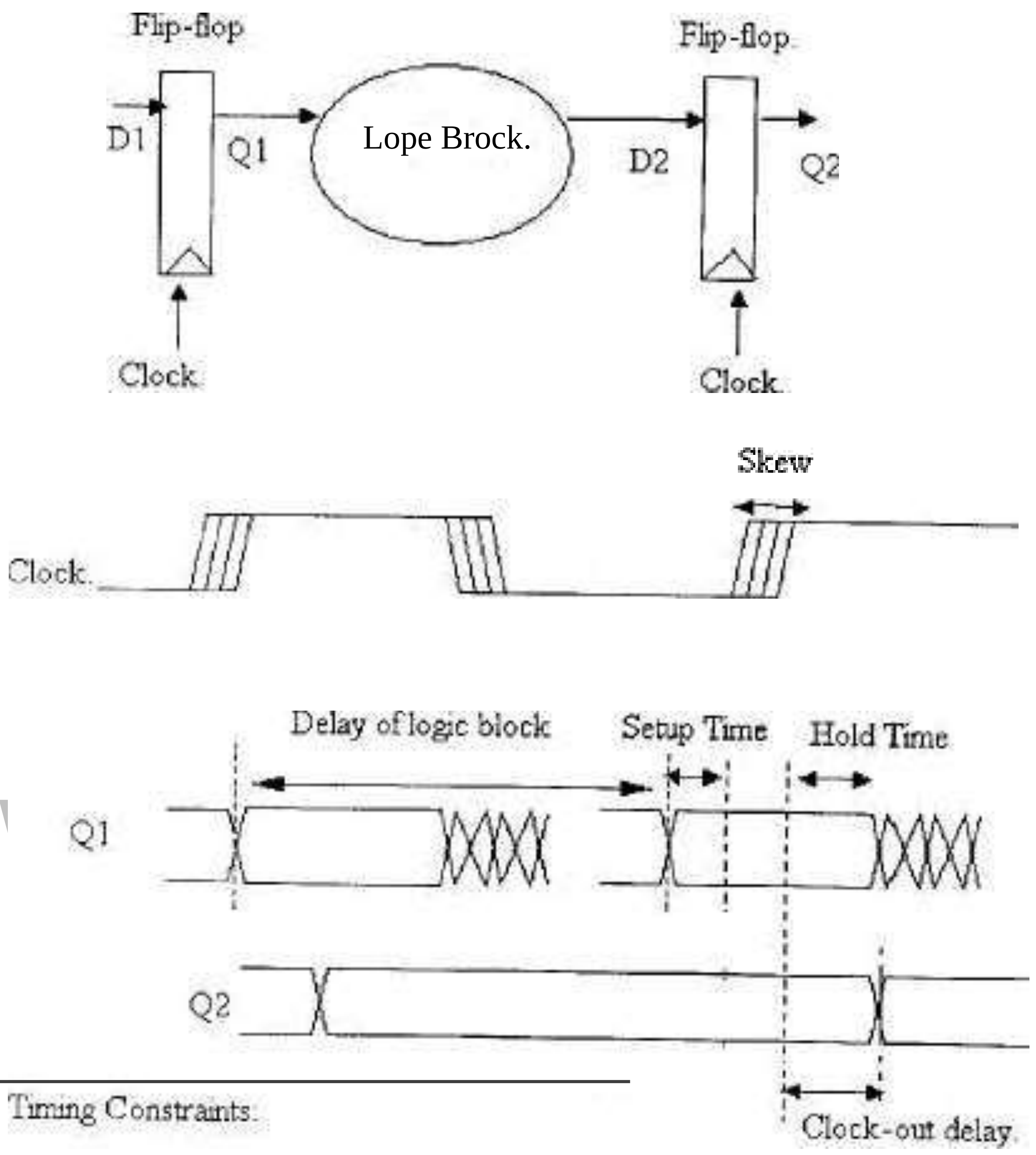
The variation of $c_{lock-out}$ delay with respect to setup time has been well studied. The studies show that, as the setup time is reduced, the $c_{lock-out}$ increases. The setup time cannot be reduced beyond a certain point after which new data is not latched into the flip-flop. They also defined the stable, transition and failure regions. The stable region is defined as the region of Data-to-Clock axis in which clock-to-out delay does not depend on the setup time. As setup time is reduced, clock-to-out delay starts to rise monotonically and ends in a failure. This region of data-to-clock axis is the transition region. The transition region is defined as the region of unstable Clock-to-out delay. Any change in data appearing in the failure region will not be latched into the flip-flop.

5. Hold time (t$_{hold}$)

Hold time is the minimum time that the data at D_{in} must be held constant after the triggering edge of the clock signal, assuming that the most recent data change occurred not later than the Setup time s_{etup}, such the output Q_{out} will remain stable. This is denoted as t_{old} and this value can be negative.

6. Clock Skew (t$_{skew}$)

Clock skew is the difference in arrival time of the active clock edge between two registers. One of the primary concerns in the design of a distribution network in a digital system is the difference in the arrival times of clock edge between two flip-flops. The goal of clock distribution is to minimize, manage or eliminate this term altogether. Clock skew is of importance between sequentially adjacent registers. The presence of clock skew can significantly limit the performance of a synchronous system, and even create race condition that causes incorrect data to be latched into flip-flop. Some of the main sources of clock skew can be attributed to differences in line lengths from clock source to the destination register;; differences in delays of any active buffers in the network, differences in interconnect parameters in the network. It has been predicted that it is hard to maintain the total clock skew less than about 150 pS for high performance microprocessors. But many active clock distribution schemes have been implemented to obtain clock skews as low as 15 to 20 pS. The values reported in literature sometimes differ in their definitions of skews and how they have been measured or estimated. Usually, local clock skew is defined as the clock-skew between any two points on the same clock-grid while global clock-skew is defined as clock-skew between any two points on the entire clock network. This would mean



Timing Constraints:

Max Delay Constraint:

$T_{cycle} > t_{ck\text{-}out} + t_{logic} + t_{setup} + t_{skew}$

Min Delay Constraint:

$t_{logic} > t_{hold} - t_{ck\text{-}out} - t_{skew}$

Fig 3.5.2: pipelining

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design]

That local clock skew can be significantly lowered while global clock would be hard to contain and probably would be as high as 150 pS even for best clock network designs.

Figure 3 illustrates the timing for a system built using edge-triggered flip- flops. The timing constraints for such a system are also shown in the figure. The Maximum delay constraint (indicated as Max. constraint) indicates that the clock cycle time should be greater than the delay of the logic block and the timing overheads introduced by flip-flops. This is called the Max. Constraint as this constraint defines the maximum logic delay (l_{ogic}) acceptable for a given clock, given flip flop timing parameters and clock skew. We reproduce the Max. Constraint here for convenience:

$$(1) \qquad T_{cycle} > t_{clock-out} + t_{logic}(\max) + t_{setup} + t_{skew}$$

In eqn. 1, the term l_{ogic} (max) refers to the maximum delay in the logic block. Let us consider the example of $\log(|a+b|)$ evaluation of (shown in Fig. 2) we have seen that it involves three operations of delay 35nsec. 40nsec. and 60nsec. Respectively. Let the $c_{lock-out}$, s_{etup}, s_{kew} and t_{old} for the registers used be respectively 15nsec. 10nsec. 10nsec. And 5nsec. With the maximum logic delay of the three logic operations is 60nsec. The C_{ycle} has to be greater than 95nsec. (15 + 60 + 10 + 10). The output

A natural question that arises in a reader's mind is the non-inclusion of t_{old} in the maximum delay constraint. This becomes obvious when we realize that t_{old} is subsumed in the combination of $c_{lock-out}$ and l_{ogic} as the hold process is concurrent with clock-out and combinational delay. The minimum delay constraint poses a limit on the minimum delay a logic block must have in order to avoid a data race between flip-flops. Data to

The flip-flop must be setup before the earliest clock might arrive;; yet we cannot guarantee that data will be valid until the clock-out delay after the latest clock. As we can see the amount of cycle time available for logic is reduced by setup time, skew and clock-out delay of the flip-flop. The Min delay constraint is expressed as

(2) $$t_{logic}(\min) > + t_{hold} - t_{clock-out} - t_{skew}$$

The Minimum logic delay that must be inserted to avoid a min delay failure is the sum of hold-time and skew minus the minimum clock-out delay. Also in these systems the logic delay is exactly known at the time cycles are partitioned, so some cycle have more logic and some have less logic. The clock cycle time should be long enough for the longest cycle to work correctly, which would imply that excess time in shorter cycles is wasted. Furthermore, for a pipeline which is of N stages, there is a total latency (a period when nothing is available at the output) of N clock cycles. In the case of the example considered, it will be 3 cycles and will be a minimum of 285 sec.

In summary, the total overhead of the flip-flop based systems is distributed in clock-out delay, setup time, clock skew and overhead due to imbalance logic.

2 Timing Analysis of Systems using level-triggered latched

Using level-triggered transparent latches can eliminate the clock-skew penalties of flip-flop systems. Such a system would operate with two-phase transparent latches. Transparent latches have same terminals as flip-flops;; data input D, clock input and data output Q. The latch is transparent when the clock input is high and the input is copied to the output. When the clock goes low, the previous value of data is retained.

Transparent latches are characterized by clock- out delay, data-out delay ($d_{ata\text{-}out}$) and hold time. The data-out delay is defined as the time taken for the o_{ut} to be stable after D_{in} gets to be stable. Some refer to data-out delay as set up time for latches. Again we assume s_{kew} of uncertainty in the arrival of clock input. Figure 4 illustrates the timing analysis of a two-phase system built using transparent latches. One latch is controlled by clock input while the other input is controlled by its complement. The latching operation can be initiated anywhere within the phase and beyond the skewed clock and hence eliminating clock skew from the overhead. We reproduce the Max. Constraint and Min. constraints here for convenience:

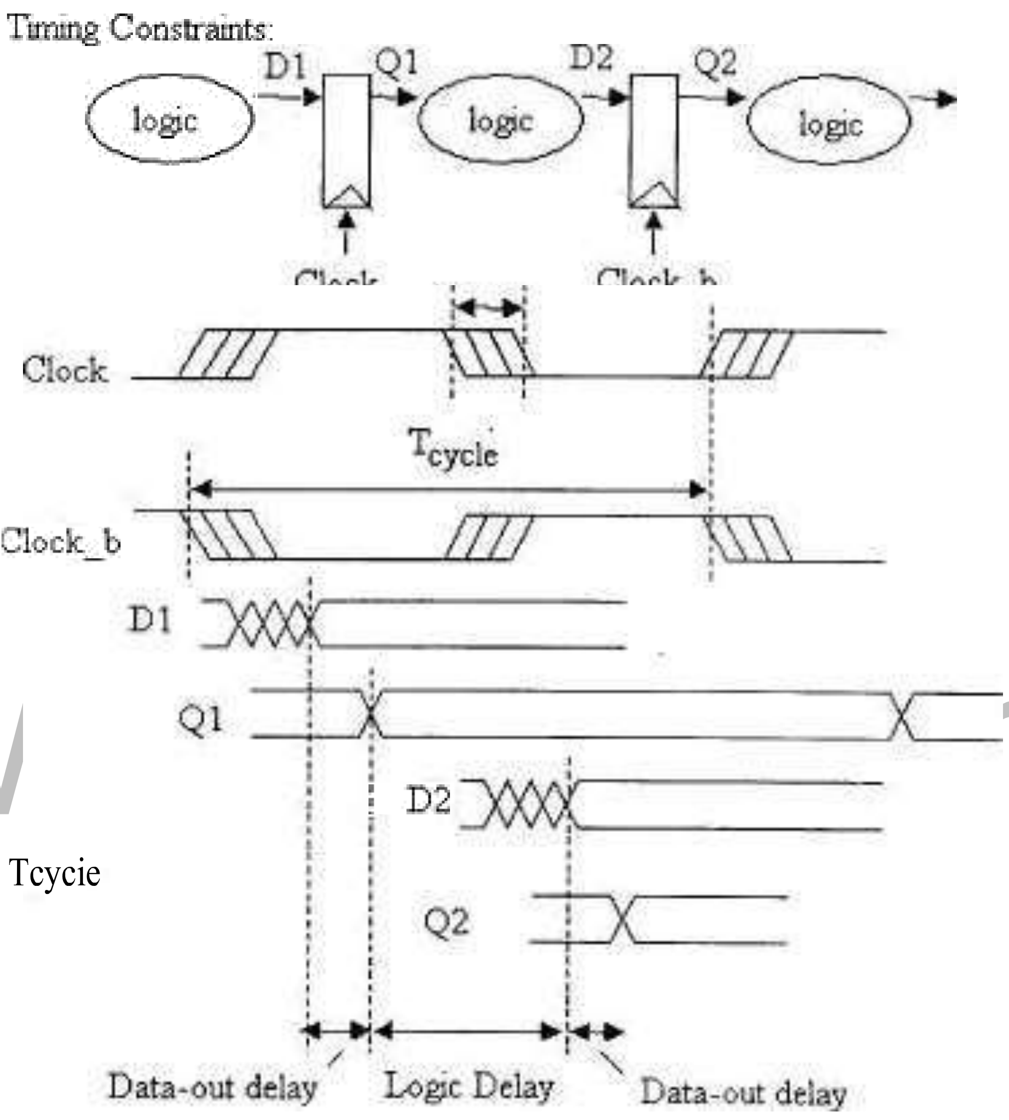
(3) $$T_{cycle} > 2 * t_{data-out} + t_{logic}(\max)$$

(4) $$t_{logic}(\min) > t_{hold} - 2 * t_{data-out} - t_{skew}$$

Level sensitive latches avoid the problem of imbalanced logic using time borrowing (also known as cycle stealing). As we can see from the figure, each latch can be placed anywhere in the wide ranges of locations in its half-cycle and still be transparent when the data arrives. This means that not all half-cycles need to have the same amount of logic. Some can have more and some can have less, such that data arrives at the latch later or earlier as long as latch is transparent. Hence, longer cycles may borrow time from shorter logic cycles if the pipelines are not perfectly balanced (not all logic delays are identical) so that latency is not determined by the total logic delay.

Let us consider the example of evaluation of $\log(|a+b|)$, redrawn in Fig. 5 by using level sensitive latches for registers. We have seen that it involves three

Latch 2



Tcycie

Latch1

Fig 3.5.3: Timing Analysis of Systems using level-triggered latched

Min delay constraint

logic hold data-out skew

[Source: Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits: Analysis & Design] operations with delays 35nsec., 40nsec. and 60nsec. respectively. Consider that each latch has $t_{clock\text{-}out}$ and t_{hold} respectively 15nsec., and 5nsec. and the clock

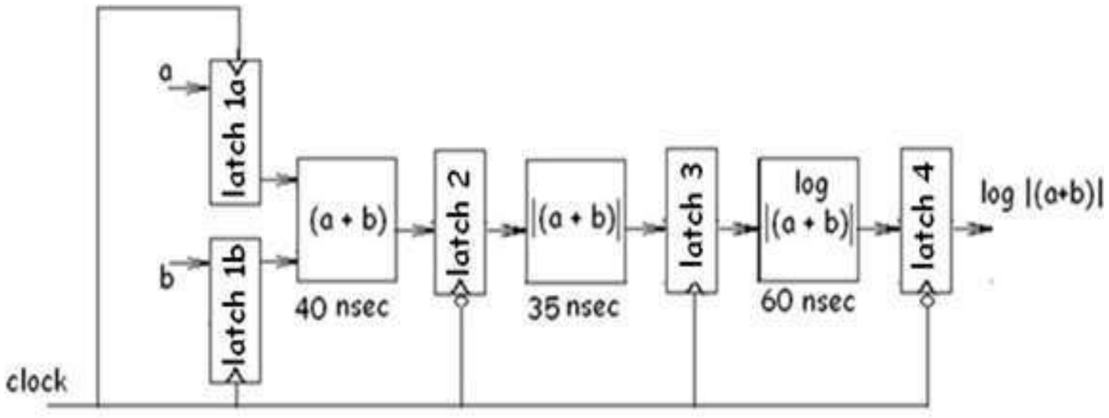


Fig 3.5.4 : Timing Analysis of Systems using level-triggered latched

[Source : Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits:Analysis & Design]

skew t_{skew} = 0 nsec.

In Fig. 6 we give an epoch tracing in the top portion and the clock diagram at the lower portion of the circuit. From the epoch tracing we find out that the operation is completed in 195 sec. and can be accommodated in 2 cycles. For convenience we have chosen the period to be 98 sec. In other words, we have a latency of only two cycles and with a clock period of 98 sec. and it works out to be 196 sec. It is left to the students to figure out that for the minimum logic delay defined by the expression in eqn.4, s_{kew} has no effect on the clock cycle time at all.

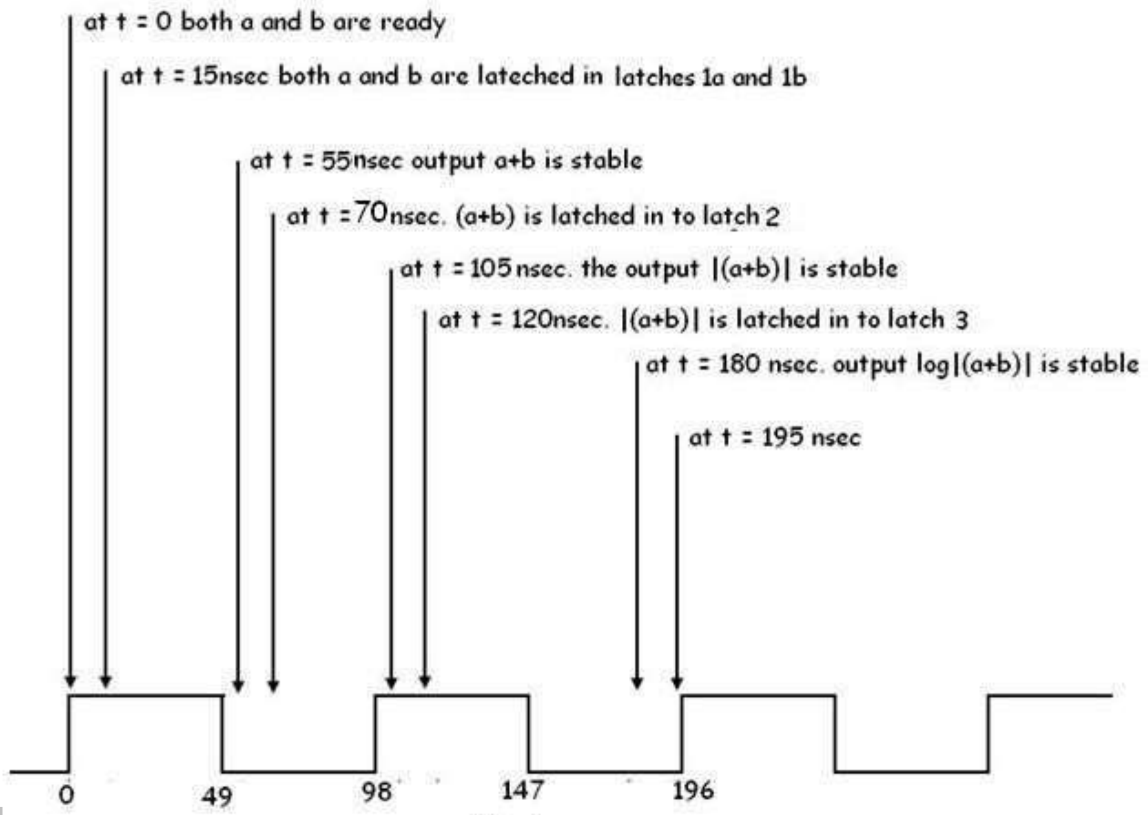

at t = 0 both a and b are ready
at t = 15nsec both a and b are latched in latches 1a and 1b
at t = 55nsec output a+b is stable
at t = 70nsec. (a+b) is latched in to latch 2
at t = 105nsec. the output |(a+b)| is stable
at t = 120nsec. |(a+b)| is latched in to latch 3
at t = 180 nsec. output log|(a+b)| is stable
at t = 195 nsec
0
49
98
147
196

Fig. 6

Has $c_{lock-out}$ and t_{old} respectively 15nsec. And 5nsec. And the clock

Fig 3.5.5: Timing Analysis of Systems output

[Source: Sung-Mo kang, Yusuf leblebici, Chulwoo Kim ―CMOS Digital Integrated Circuits: Analysis & Design]

In summary, systems constructed from transparent latches eliminate theoverhead of clock-skew and imbalance logic but still incur penalty introduced by data-out delays of the latches. Another major disadvantage with such a system would be distribution of two-phase clocks throughout the chip.

2 Timing Analysis of Systems using Positive Pulse Triggered Flip Flops:

A pulsed triggered flip flop behaves very similar to level triggered latches. The clock input to pulse triggered flip flop is a pulse with pulse width less than

50% of the clock cycle. Pulse triggered flip flops also have three terminals: data input D, clock (pulse) input and output Q. The latch is transparent during the pulse width and input is copied to the output terminal. In a positive pulse triggered flip flop, when the clock goes low, previous value of the data is retained. These flip flops are characterized by their pulse width, setup time, hold time and Data-to-out delays.

Figure 7 illustrates system timing for a system built with pulsed latches. The timing constraints are also shown in the figure. The data must arrive at least setup time before the earliest falling transition of the clock. We reproduce the Max. Constraint and Min. constraints here for convenience:

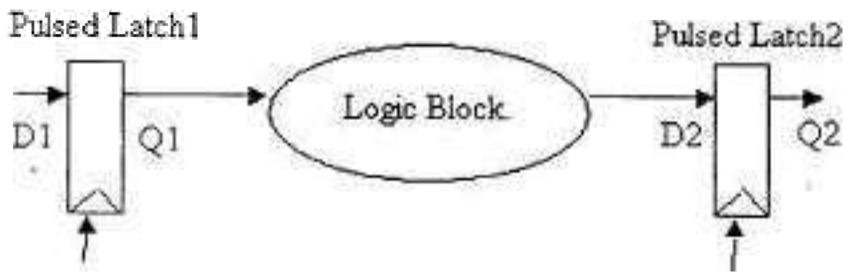
$$T_{cycle} > t_{clock-out} + t_{logic}(\max) + Max\{0, (t_{setup} + t_{skew} - t_{pulse-width})\} \quad (5)$$

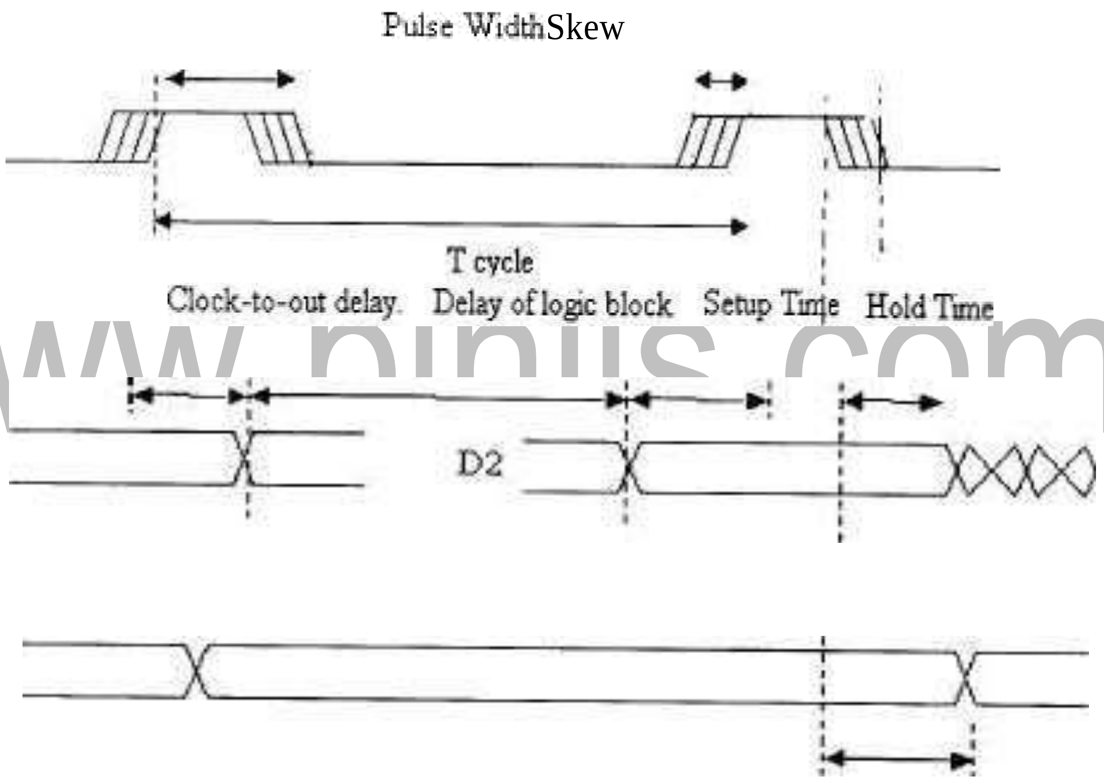
(6) $t_{logic}(\min) > t_{hold} - t_{clock-out} + t_{pulse-width} - t_{skew}$

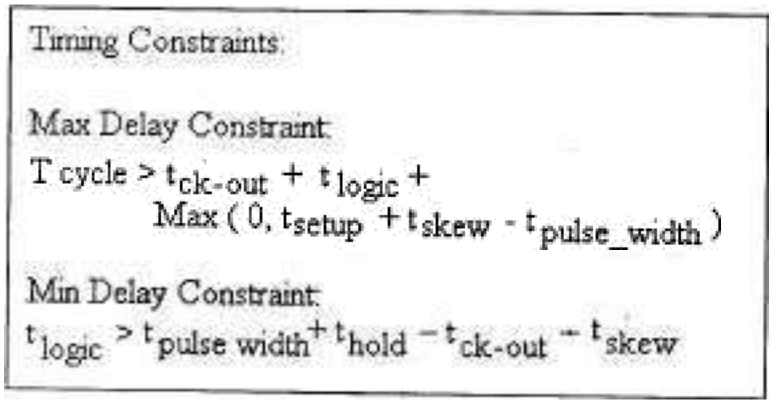
A wider pulse allows time borrowing where data from previous block can arrive late allowing it to borrow time from the subsequent stage. But the pulse width should also be short enough to avoid min delay failures arising when the delay of the logic block is shorter than the pulse width of the subsequent pulsed latch. It is difficult to distribute a clock pulse globally and usually a pulse is generated within the latch itself, which controls its operation. As we can see a pulsed latch allows time borrowing and provides better immunity against clock skews. The latency introduced by a pulsed latch is typically the data-to-out delay for data arriving in the pulse width.

Unlike most of the edge triggered flip-flops, pulsed triggered flip flops have a positive hold time requirement (which is close to the pulse width). Hence this parameter also needs to be considered for a pulsed latch. Clock skew usually eats into the available time for logic. Pulsed latches absorb clock skew while flip-flops introduce hard edges and provide no immunity against clock skew.

Let us now consider the same example as in the earlier two cases. The circuit diagram will be the same as Fig. 2 with the registers being realized using pulse







Timing Constraints:

Max Delay Constraint:

$$T_{cycle} > t_{ck\text{-}out} + t_{logic} + \text{Max}(0, t_{setup} + t_{skew} - t_{pulse\_width})$$

Min Delay Constraint:

$$t_{logic} > t_{pulse\ width} + t_{hold} - t_{ck\text{-}out} - t_{skew}$$

Fig 3.5.6: Timing Constraints

[Source: Sung-Mo kang, Yusuf leblebici, Chulwoo Kim —CMOS Digital Integrated Circuits: Analysis & Design]

Triggered flip flops. In this case let $c_{lock\text{-}out}$, s_{etup}, s_{kew} and t_{old} for the registers used be respectively 15nsec. 10nsec. 10nsec. And 15nsec. With the pulse width $p_{ulse\text{-}width}$ equal to 15 sec. using eqn. 5, we can evaluate the C_{ycle} to be 75 sec. and the minimum logic delay to be 25 sec. This configuration also gives a latency of three clock cycles which works out to be 225 sec.

In summary, comparing the three methods of pipelining, pulse triggered flip flop based pipelining gives the highest possible frequency of operation. In pulse triggered flip flop based pipeline system, we do not distribute pulses but generate it internally for each flip flop and distribute only the global clock for all the flip flops.