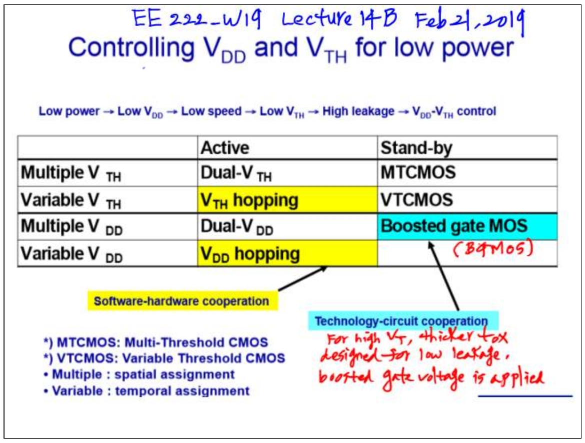


EE 222_W19 Lecture 14B Feb 21, 2019

Controlling V_{DD} and V_{TH} for low power

Low power → Low V_{DD} → Low speed → Low V_{TH} → High leakage → V_{DD}-V_{TH} control

	Active	Stand-by
Multiple V_{TH}	Dual-V_{TH}	MTCMOS
Variable V_{TH}	V_{TH} hopping	VTCMOS
Multiple V_{DD}	Dual-V_{DD}	Boosted gate MOS
Variable V_{DD}	V_{DD} hopping	(BgMOS)

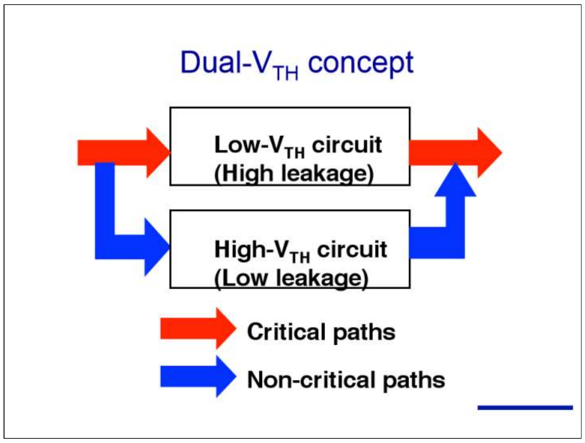
Software-hardware cooperation

Technology-circuit cooperation

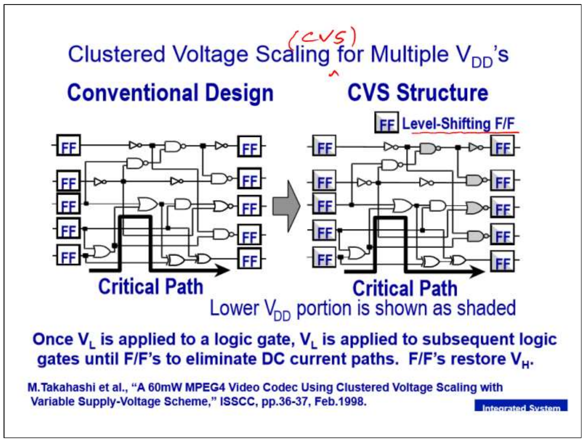
For high V_T, thicker tox designed for low leakage, boosted gate voltage is applied

*) MTCMOS: Multi-Threshold CMOS
*) VTCMOS: Variable Threshold CMOS

- Multiple : spatial assignment
- Variable : temporal assignment



Dual-V_{TH} concept



Clustered Voltage Scaling (CVS) for Multiple V_{DD}'s

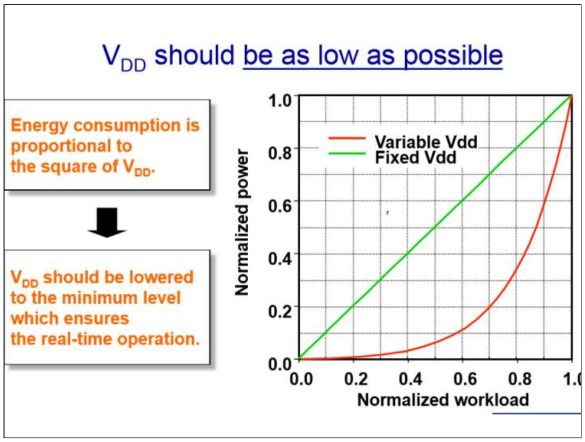
Conventional Design **CVS Structure**

Lower V_{DD} portion is shown as shaded

Once V_L is applied to a logic gate, V_L is applied to subsequent logic gates until F/F's to eliminate DC current paths. F/F's restore V_H.

M.Takahashi et al., "A 60mW MPEG4 Video Codec Using Clustered Voltage Scaling with Variable Supply-Voltage Scheme," ISSCC, pp.36-37, Feb.1998.

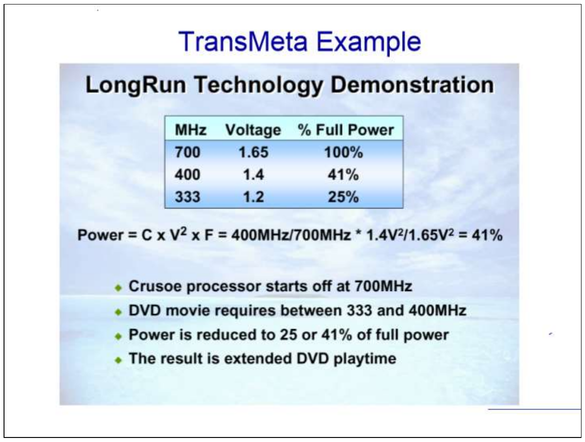
Integrated System



V_{DD} should be as low as possible

Energy consumption is proportional to the square of V_{DD}.

V_{DD} should be lowered to the minimum level which ensures the real-time operation.



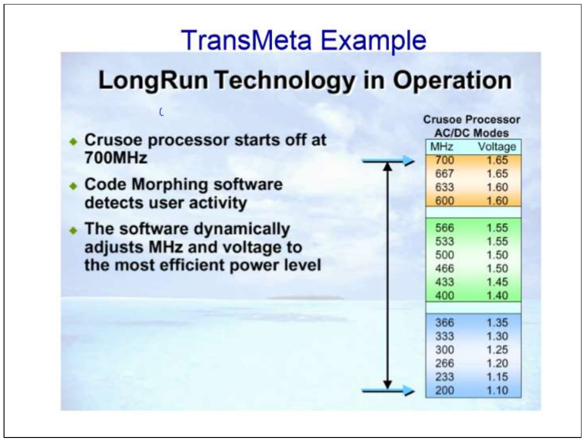
TransMeta Example

LongRun Technology Demonstration

MHz	Voltage	% Full Power
700	1.65	100%
400	1.4	41%
333	1.2	25%

Power = C x V² x F = 400MHz/700MHz * 1.4V²/1.65V² = 41%

- Crusoe processor starts off at 700MHz
- DVD movie requires between 333 and 400MHz
- Power is reduced to 25 or 41% of full power
- The result is extended DVD playtime



TransMeta Example

LongRun Technology in Operation

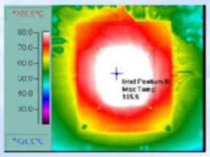
- Crusoe processor starts off at 700MHz
- Code Morphing software detects user activity
- The software dynamically adjusts MHz and voltage to the most efficient power level

Crusoe Processor AC/DC Modes

MHz	Voltage
700	1.65
667	1.65
633	1.60
600	1.60
566	1.55
533	1.55
500	1.50
466	1.50
433	1.45
400	1.40
366	1.35
333	1.30
300	1.25
266	1.20
233	1.15
200	1.10

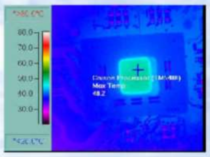
TransMeta Example

Processor Thermal Comparison



Pentium III
Playing DVD

105.5º C
221.9º F



Crusoe Processor
Playing DVD

48.2º C
118.8º F

Why is lowering VDD not enough ?

- Total P can be minimized by lower V
  - lower V are a natural result of smaller feature sizes
- But… transistor speeds decrease dramatically as V is reduced to close to "threshold voltage"
  - performance goals may not be met
  - $t_d = CV / k(V-V_t)^{\alpha}$ where α is between 1-2
- Why not lower this "threshold voltage"?
  - makes noise margin and I_{leak} worse!
- Need to do smarter voltage scaling!

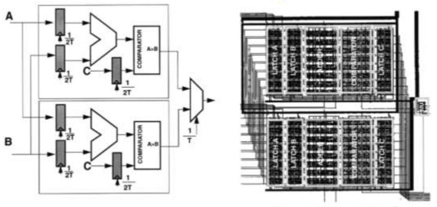
Reducing the Supply Voltage: Architectural Approach

- Operate at reduced voltage at lower speed
- Use architecture optimization to compensate for slower operation
  - e.g. concurrency, pipelining via compiler techniques
- Architecture bottlenecks limit voltage reduction
  - degradation of speed-up
  - interconnect overheads
- Similar idea for memory: slower and parallel

Trade-off AREA for lower POWER

C ($\propto$ Area) ↗ but V_{DD} ↘ decreases power

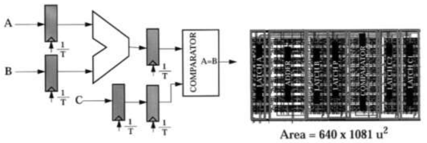
Parallel Datapath



Area = 1476 x 1219 μ^2

- The clock rate can be reduced by x2 with the same throughput: $f_{par} = f_{ref}/2$ = 20 MHz
- Total switched capacitance = $C_{par} = 2.15C_{ref}$
- $V_{par} = V_{ref}/1.7$ ↓
- $P_{par} = (2.15C_{ref})(V_{ref}/1.7)^2(f_{ref}/2) = 0.36P_{ref}$

Pipelined Datapath



Area = 640 x 1081 u^2

- $f_{pipe} = f_{ref}$
  $C_{pipe} = 1.1C_{ref}$
  $V_{pipe} = V_{ref}/1.7$
- Voltage can be dropped while maintaining the original throughput
- $P_{pipe} = C_{pipe}V_{pipe}^2 f_{pipe} = (1.1C_{ref})(V_{ref}/1.7)^2 f_{ref} = 0.37P_{ref}$

Datapath Architecture-Power Trade-off Summary

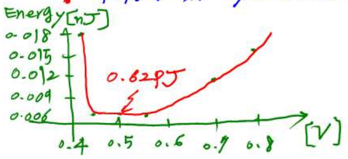
Datapath Architecture	Voltage	Area	Power
Original	5V	1	1
Pipelined	2.9V	1.3	0.37
Parallel	2.9V	3.4	0.34
Pipeline-Parallel	2.0V	3.7	0.18

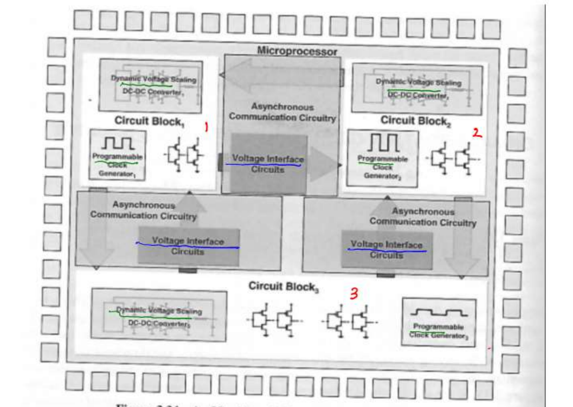
An Ultra Low Power SoC in 14 nm Trigate CMOS (ISSCC 2018 Intel Corp.) for IOT Applications

2.5 mm x 2.5 mm chip, 12M transistors

4 independent V_{DDs}

- 0.55 V for logic, ROM, register-file array
- 0.80 V for dense SRAM array
- 1.05 V for radio
- 1.70 V for I/O transceivers





Microprocessor
Dynamic Voltage Scaling DC-DC Converter₁
Circuit Block₁
1
Programmable Clock Generator₁
Asynchronous Communication Circuitry
Voltage Interface Circuits
Dynamic Voltage Scaling DC-DC Converter₂
Circuit Block₂
2
Programmable Clock Generator₂
Asynchronous Communication Circuitry
Voltage Interface Circuits
Asynchronous Communication Circuitry
Voltage Interface Circuits
Circuit Block₃
3
Dynamic Voltage Scaling DC-DC Converter₃
Programmable Clock Generator₃

Figure 3.34 An IC with multiple voltage and clock domains

MULTIPLE SUPPLY AND THRESHOLD VOLTAGE CMOS 79

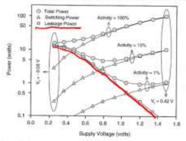


Figure 3.30 Power dissipation of a test circuit with varying supply voltage for three different activity factors assuming a 2 GHz clock frequency and a 100 nm CMOS technology [53]

As shown in Figure 3.31, for a target clock frequency of 2 GHz, the optimum supply voltage that minimizes the total power consumption of a dual supply and threshold voltage test circuit is 1.2 V. For this supply voltage and for the same clock frequency, the total power consumption of the dual supply and threshold voltage test circuit (V_{DD1} = 1.2 V,

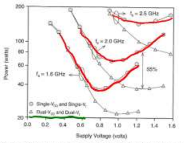
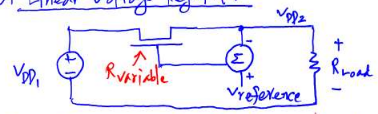
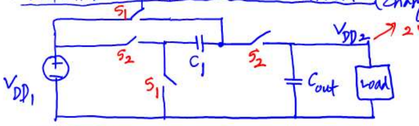


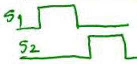
Figure 3.31 Power dissipation of dual V_{DD}/dual-V_t and standard single V_{DD}/single V_t test circuits with varying supply voltage for three different clock frequencies, assuming a 100 nm CMOS technology. For a dual V_{DD}/dual-V_t circuit, V_{DD2} and $|V_{t2}|$ are fixed at 0.5 V and 0 V, respectively. The supply voltage of the low activity circuits (V_{DD1}) is varied together with the threshold voltages ($|V_{t1}|$) while maintaining a target clock frequency [53]

A Linear Voltage Regulator



A Switched Capacitor DC-DC converter (charge Pump)



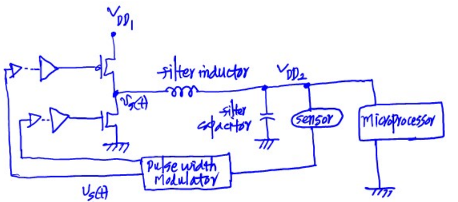


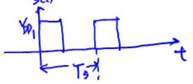
1) when S_1 is closed, C_1 charged to V_{DD_1}

when S_2 is closed & S_1 open,

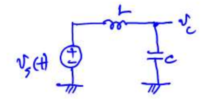
2) Cout is charged to $2V_{DD_1}$.

A Buck Converter





$$V_{DD_2} = \frac{1}{T_s}\int_0^{T_s} V_s(t)\,dt$$



$$v_s(t) = L\frac{di}{dt} + v_c \qquad (1)$$

$$C\frac{dv_c}{dt} = i \qquad (2)$$

$$C\frac{d^2v_c}{dt^2} = \frac{di}{dt} \qquad (3)$$

(3) → (1)

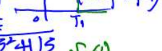
$$v_s(t) = LC\frac{d^2v_c}{dt^2} + v_c$$

$\downarrow \mathcal{L}$

$$V_s(s) = LC\left(s^2V_c(s) - s\,v_c(0) - v_c'(0)\right) + V_c(s)$$

$$V_s(s) = (LCs^2+1)V_c(s) - s\,v_c(0) - v_c'(0)$$

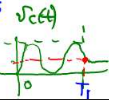
For $v_c(0)=0$, $v_c'(0)=0$, $v_s(t) = E\,u(t)$ (step input)

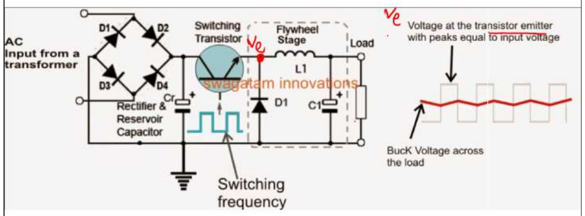


$$V_c(s) = \frac{1}{LCs^2+1}V_s(s) = \frac{E}{(LCs^2+1)s}$$

$$= \frac{\left(\frac{E}{LC}\right)}{\left(s^2+\frac{1}{LC}\right)s} = E\left[\frac{1}{s} + \frac{-s}{s^2+\frac{1}{LC}}\right]$$

$$v_c(t) = E\left[1 - \cos\frac{1}{\sqrt{LC}}t\right]$$





Combinational Vs. Sequential Circuits

Combinational (or Combinatorial) Networks/Circuits

- Circuit without storage
- Outputs depend only on its current inputs
- Examples: NAND gate, look-up table (LUT)

Vs.

Sequential Networks/Circuits

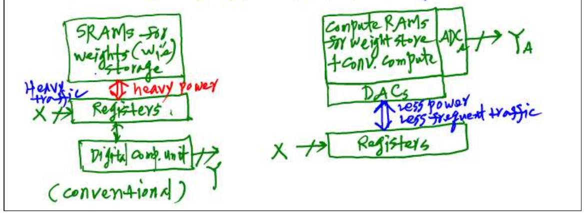
- Circuit with storage elements
- Outputs depend on present inputs and also on history of inputs
- Examples: RAM, finite state machine (FSM)
- Normally, combinational cells AND storage elements

Computation in Memory

(Also called Processing in Memory)

Ref. Convolution-RAM: An Energy Efficiet SRAM with Embedded Convolution Computation for Low Power CNN (convolution Neural Network) - based Machine Learning Applications

ISSCC 2018 A. Biswas, et al. MIT



$$Y = \sum_i W_i \times X_i = \sum_i \alpha\, w_i \times X_i = \frac{M}{N} \sum w_i \times X_i$$

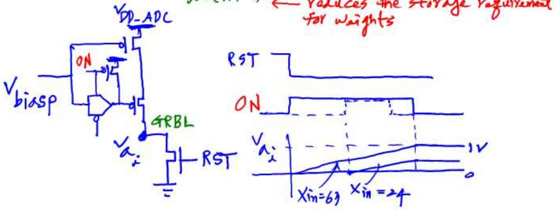
$w_i \in (+1, -1)$

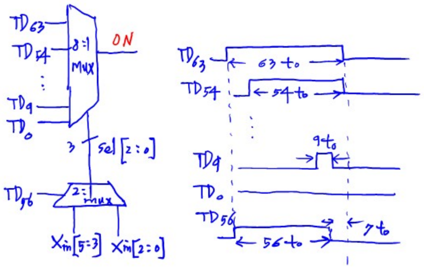
$$Y_A = \frac{1}{N} \sum W_i \times X_i = ADC\Big[\underbrace{\frac{1}{N} \sum w_i \times \overbrace{DAC(X_i)}^{V_{a_i}}}_{Y_{A\,average}}\Big]$$

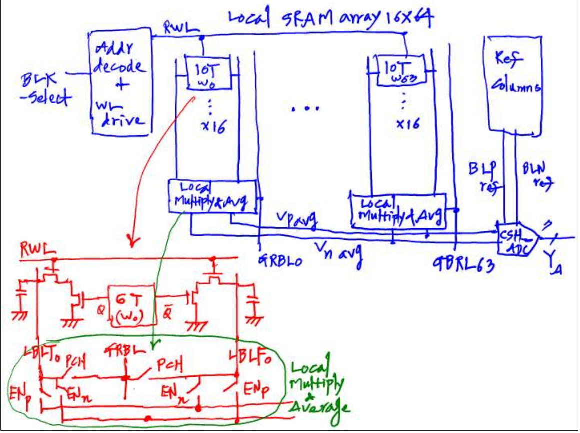
weight multiply & average

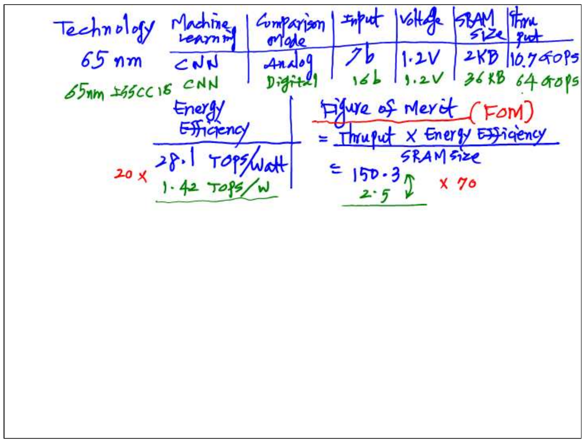
$$V_{P\,average} = \frac{1}{N} \sum_{i=0}^{N-1} |w_i| \times V_{a_i}$$

$w_i \in (+1, -1)$ ← reduces the storage requirement for weights









Technology	Machine Learning	Comparison mode	Input	Voltage	SRAM size	Thru put
65 nm	CNN	Analog	7b	1.2V	2KB	10.7 GOPS
65nm ISSCC 18	CNN	Digital	16b	1.2V	36KB	64 GOPS

Energy Efficiency
28.1 TOPS/Watt
1.42 TOPS/W

20 x

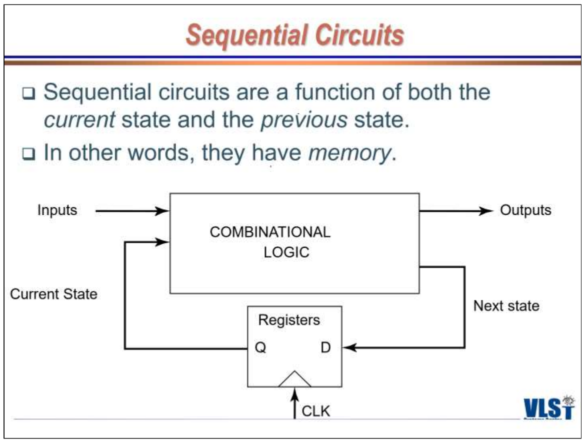
Figure of Merit (FOM)

$$= \frac{\text{Thruput} \times \text{Energy Efficiency}}{\text{SRAM size}}$$

≈ 150.3

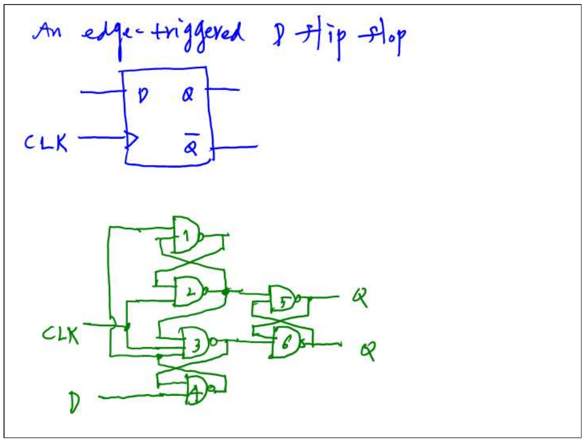
2.5

x 70

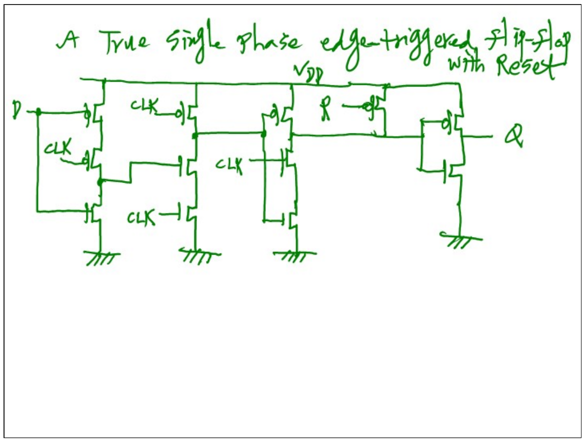


Sequential Circuits

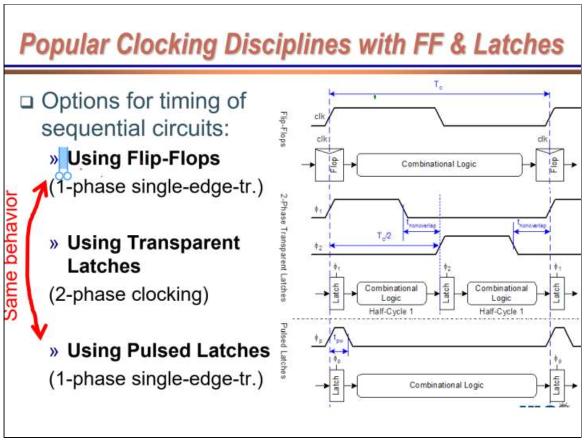
- ❑ Sequential circuits are a function of both the *current* state and the *previous* state.
- ❑ In other words, they have *memory*.



An edge-triggered D flip flop



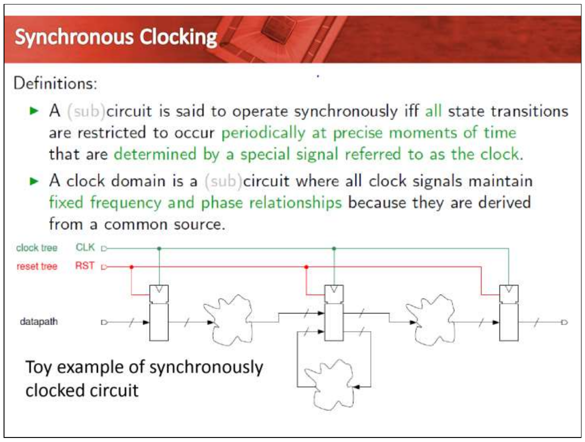
A True single phase edge-triggered flip-flop with Reset



Popular Clocking Disciplines with FF & Latches

- ❑ Options for timing of sequential circuits:
  - » **Using Flip-Flops** (1-phase single-edge-tr.)
  - » **Using Transparent Latches** (2-phase clocking)
  - » **Using Pulsed Latches** (1-phase single-edge-tr.)

Same behavior

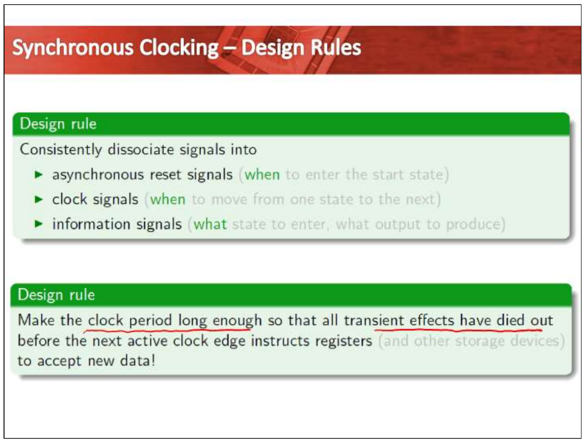


Synchronous Clocking

Definitions:

- A (sub)circuit is said to operate synchronously iff all state transitions are restricted to occur periodically at precise moments of time that are determined by a special signal referred to as the clock.
- A clock domain is a (sub)circuit where all clock signals maintain fixed frequency and phase relationships because they are derived from a common source.

Toy example of synchronously clocked circuit



Synchronous Clocking – Design Rules

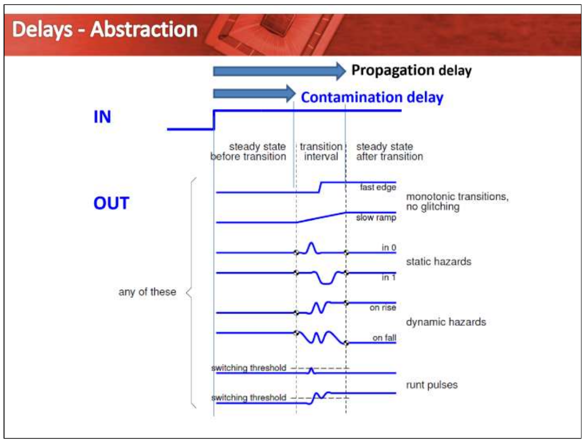
Design rule

Consistently dissociate signals into

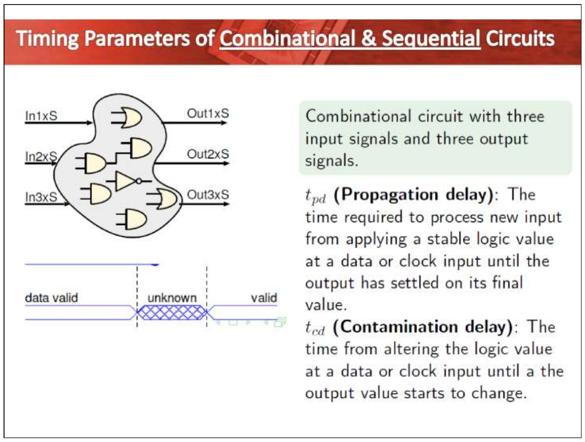
- asynchronous reset signals (when to enter the start state)
- clock signals (when to move from one state to the next)
- information signals (what state to enter, what output to produce)

Design rule

Make the clock period long enough so that all transient effects have died out before the next active clock edge instructs registers (and other storage devices) to accept new data!



Delays - Abstraction

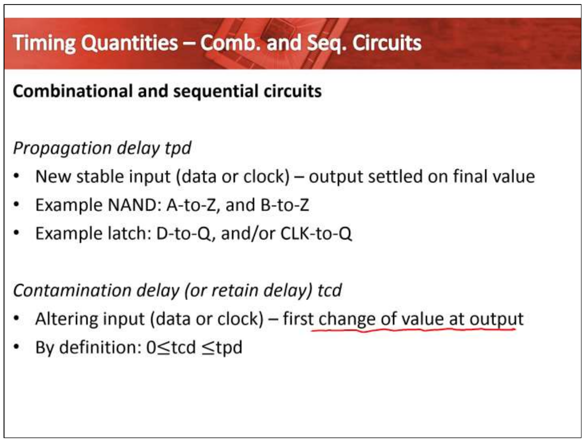


Timing Parameters of Combinational & Sequential Circuits

Combinational circuit with three input signals and three output signals.

t_{pd} **(Propagation delay)**: The time required to process new input from applying a stable logic value at a data or clock input until the output has settled on its final value.

t_{cd} **(Contamination delay)**: The time from altering the logic value at a data or clock input until a the output value starts to change.



Timing Quantities – Comb. and Seq. Circuits

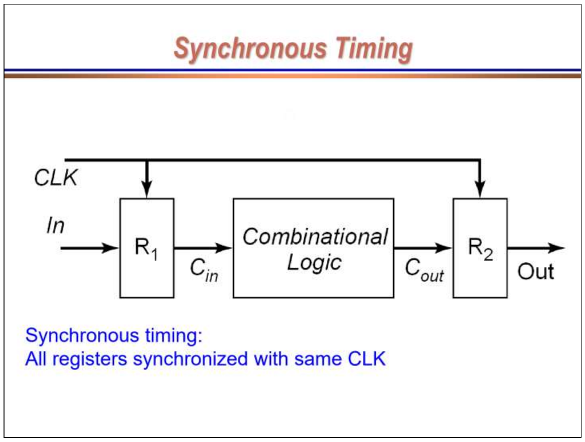
Combinational and sequential circuits

Propagation delay tpd

- New stable input (data or clock) – output settled on final value
- Example NAND: A-to-Z, and B-to-Z
- Example latch: D-to-Q, and/or CLK-to-Q

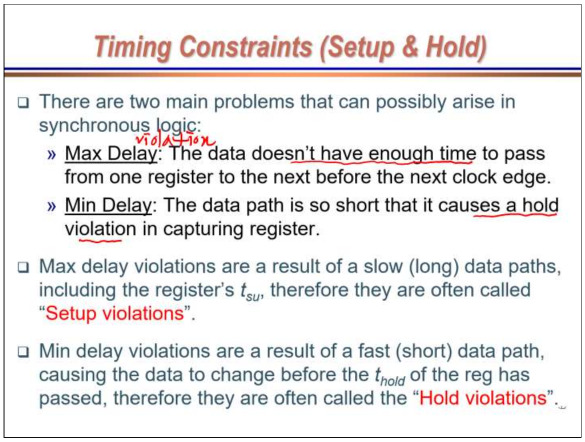
Contamination delay (or retain delay) tcd

- Altering input (data or clock) – first change of value at output
- By definition: 0≤tcd ≤tpd



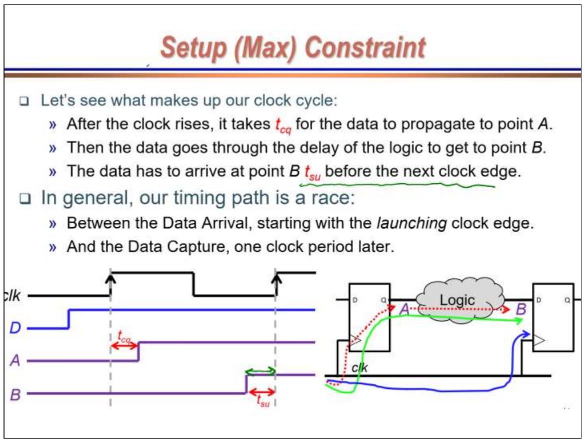
Synchronous Timing

Synchronous timing:
All registers synchronized with same CLK



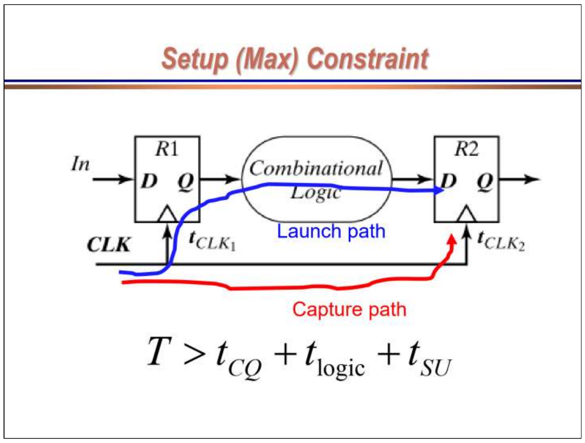
Timing Constraints (Setup & Hold)

- ❑ There are two main problems that can possibly arise in synchronous logic:
  - » Max Delay: violation The data doesn't have enough time to pass from one register to the next before the next clock edge.
  - » Min Delay: The data path is so short that it causes a hold violation in capturing register.
- ❑ Max delay violations are a result of a slow (long) data paths, including the register's t_{su}, therefore they are often called "Setup violations".
- ❑ Min delay violations are a result of a fast (short) data path, causing the data to change before the t_{hold} of the reg has passed, therefore they are often called the "Hold violations".



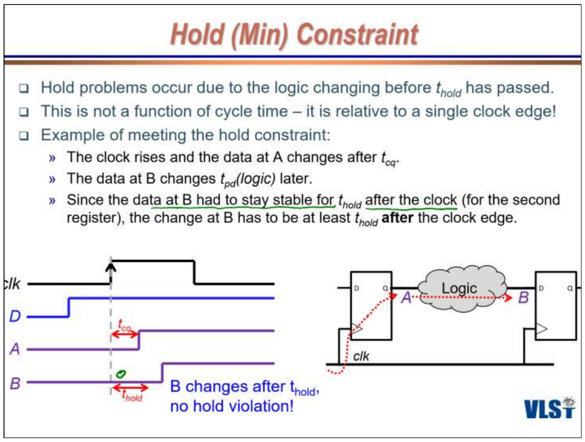
Setup (Max) Constraint

- Let's see what makes up our clock cycle:
  - » After the clock rises, it takes t_{cq} for the data to propagate to point *A*.
  - » Then the data goes through the delay of the logic to get to point *B*.
  - » The data has to arrive at point *B* t_{su} before the next clock edge.
- In general, our timing path is a race:
  - » Between the Data Arrival, starting with the *launching* clock edge.
  - » And the Data Capture, one clock period later.



Setup (Max) Constraint

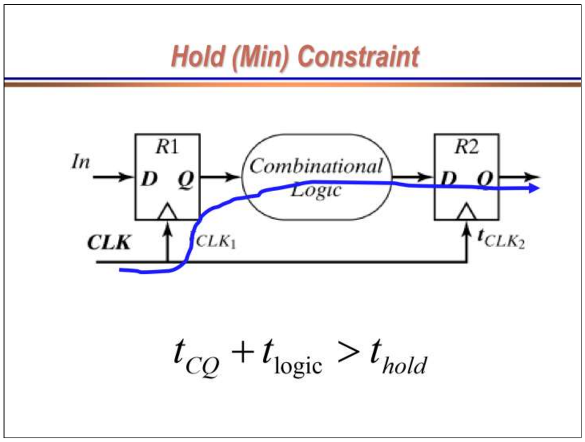
$$T > t_{CQ} + t_{\text{logic}} + t_{SU}$$



Hold (Min) Constraint

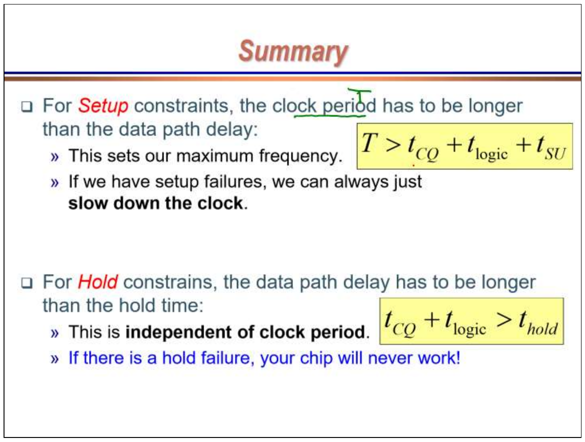
- Hold problems occur due to the logic changing before t_{hold} has passed.
- This is not a function of cycle time – it is relative to a single clock edge!
- Example of meeting the hold constraint:
  - » The clock rises and the data at A changes after t_{cq}.
  - » The data at B changes t_{pd}*(logic)* later.
  - » Since the data at B had to stay stable for t_{hold} after the clock (for the second register), the change at B has to be at least t_{hold} **after** the clock edge.

B changes after t_{hold}, no hold violation!



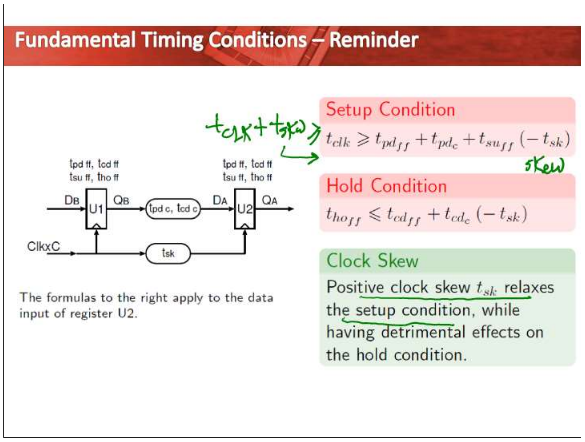
Hold (Min) Constraint

$$t_{CQ} + t_{\text{logic}} > t_{hold}$$



Summary

- For *Setup* constraints, the clock period has to be longer than the data path delay: T $\boxed{T > t_{CQ} + t_{\text{logic}} + t_{SU}}$
  - » This sets our maximum frequency.
  - » If we have setup failures, we can just **slow down the clock**.
- For *Hold* constrains, the data path delay has to be longer than the hold time: $\boxed{t_{CQ} + t_{\text{logic}} > t_{hold}}$
  - » This is **independent of clock period**.
  - » If there is a hold failure, your chip will never work!



Fundamental Timing Conditions – Reminder

The formulas to the right apply to the data input of register U2.

$t_{clk} + t_{skw} >$

Setup Condition

$$t_{clk} \geqslant t_{pd_{ff}} + t_{pd_c} + t_{su_{ff}} \, (-t_{sk})$$

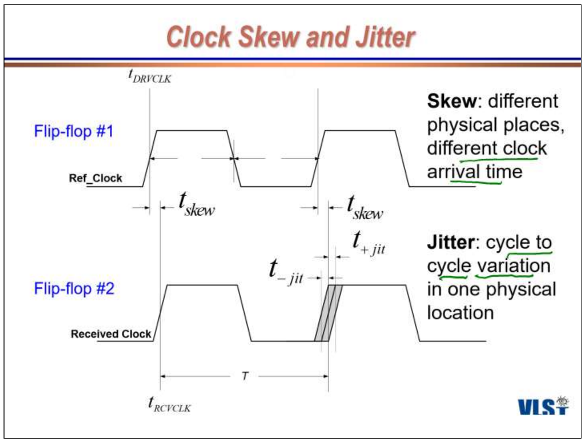
skew

Hold Condition

$$t_{ho_{ff}} \leqslant t_{cd_{ff}} + t_{cd_c} \, (-t_{sk})$$

Clock Skew

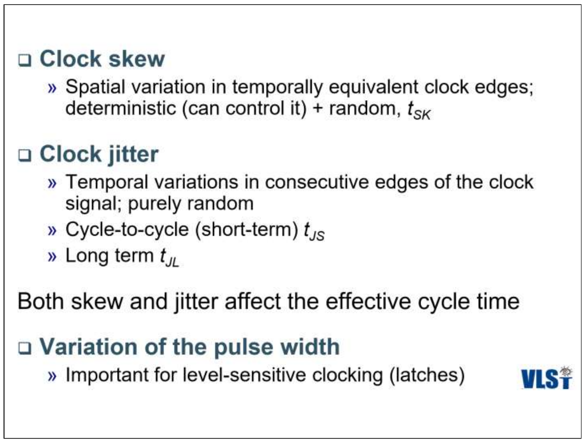
Positive clock skew t_{sk} relaxes the setup condition, while having detrimental effects on the hold condition.



Clock Skew and Jitter

Skew: different physical places, different clock arrival time

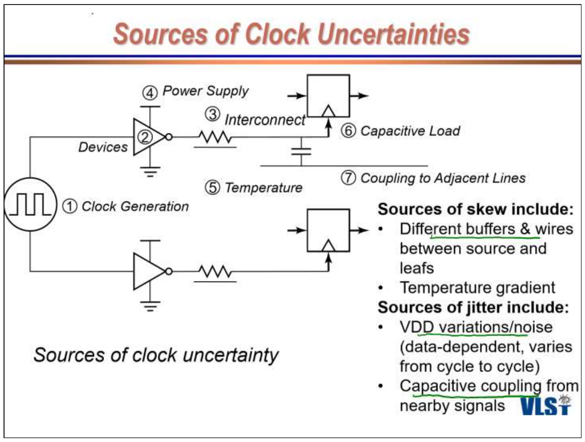
Jitter: cycle to cycle variation in one physical location



- ❑ **Clock skew**
  - » Spatial variation in temporally equivalent clock edges; deterministic (can control it) + random, t_{SK}
- ❑ **Clock jitter**
  - » Temporal variations in consecutive edges of the clock signal; purely random
  - » Cycle-to-cycle (short-term) t_{JS}
  - » Long term t_{JL}

Both skew and jitter affect the effective cycle time

- ❑ **Variation of the pulse width**
  - » Important for level-sensitive clocking (latches)



Sources of Clock Uncertainties

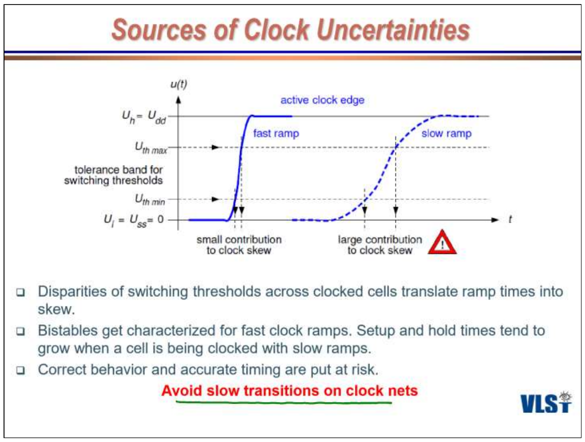
Sources of clock uncertainty

Sources of skew include:

- Different buffers & wires between source and leafs
- Temperature gradient

Sources of jitter include:

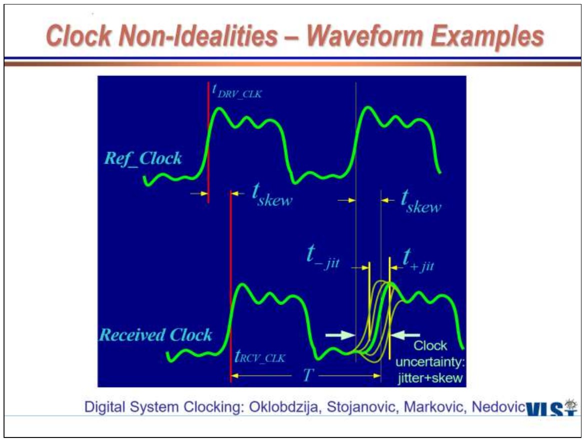
- VDD variations/noise (data-dependent, varies from cycle to cycle)
- Capacitive coupling from nearby signals



Sources of Clock Uncertainties

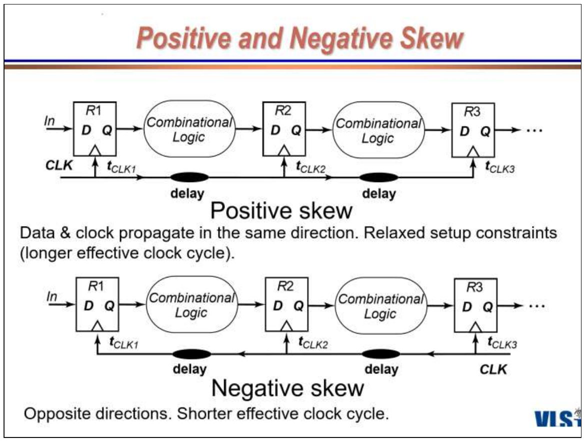
- ❑ Disparities of switching thresholds across clocked cells translate ramp times into skew.
- ❑ Bistables get characterized for fast clock ramps. Setup and hold times tend to grow when a cell is being clocked with slow ramps.
- ❑ Correct behavior and accurate timing are put at risk.

Avoid slow transitions on clock nets



Clock Non-Idealities – Waveform Examples

Digital System Clocking: Oklobdzija, Stojanovic, Markovic, Nedovic



Positive and Negative Skew

Positive skew

Data & clock propagate in the same direction. Relaxed setup constraints (longer effective clock cycle).

Negative skew

Opposite directions. Shorter effective clock cycle.

Setup (Max) Constraint

- The *Launch path* (still) consists of: $t_{launch} = t_{CQ} + t_{\text{logic}} + t_{SU} + t_{jitter}$
  - » $t_{cq}+t_{logic}+t_{setup}$
  - » But if jitter makes the launch clock later, we need to add it to the data path delay.
- The *Capture path* consists of: $t_{capture} = T + \delta - t_{jitter}$
  - » The clock period (*T*)
  - » Positive skew means the capture clock path is longer.
  - » If jitter makes the capture clock earlier, we need to subtract it.
- Our max constraint is: $t_{capture} \geq t_{launch}$
- So we get:

$$T \geq t_{CQ} + t_{\text{logic}} + t_{SU} + 2t_{jitter} - \delta$$

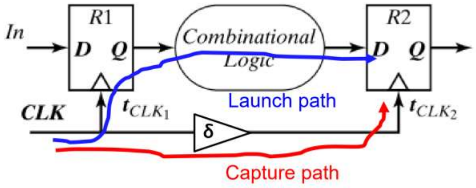
$+\delta$ Neg. skew

Pos. skew

Jitter: always worse T

Setup (Max) Constraint

- Data has to arrive before *next* active clock edge.



$$T + \delta > t_{CQ} + t_{\text{logic}} + t_{SU} + 2t_{jitter}$$

Hold (Min) Constraint

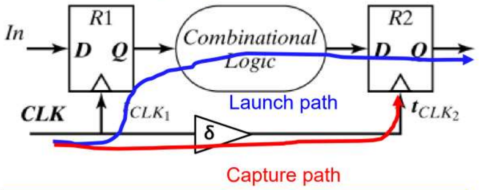
- The *Launch path* (still) consists of: $t_{launch} = t_{CQ} + t_{\text{logic}} - t_{jitter}$
  - » $t_{cq}+t_{logic}$
  - » But if jitter makes the launch clock *earlier*, we need to subtract it from the data path delay.
- The *Capture path* consists of: $t_{capture} = \delta + t_{jitter} + t_{hold}$
  - » Skew that makes the clock edge arrive at the capture register later than at the launch register.
  - » Actually, since it is a single clock edge, jitter should effect the capture clock the same as the launch clock.
  - » But as a worst case, we will add it as spacial jitter.
- Our min constraint is: $t_{launch} \geq t_{capture}$
- So we get:

$$t_{CQ} + t_{logic} \geq \delta + t_{hold} + 2t_{jitter}$$

Clock non-idealities: More difficult to meet hold constraint, need longer contamination delay

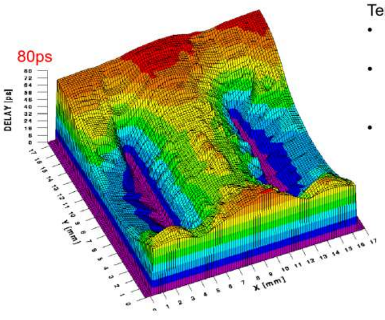
Hold (Min) Constraint

- Data has to arrive *after* same clock edge has arrived at capturing register



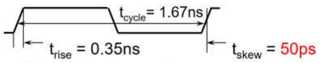
$$t_{CQ} + t_{\text{logic}} > t_{hold} + \delta + 2t_{jitter}$$

Clock Skew in Alpha Processor

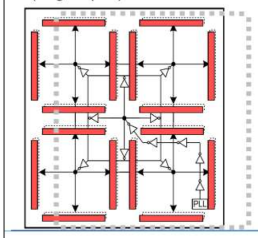


Terrible clock skew
- Zero skew by clock drivers
- Data path in middle has intermediate skew
- Peripherals for I/O has high skew

EV6 (Alpha 21264) Clocking



Global clock waveform
(target spec)

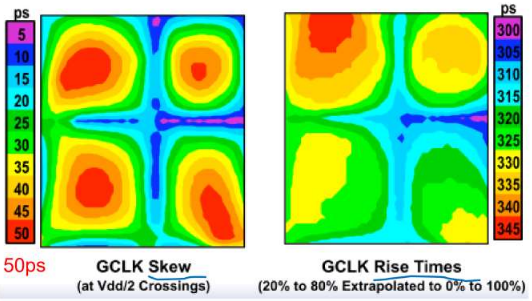


Next version of Alpha processor had less skew: target >50ps

Insert clock from PLL into center, then H-tree, and more local final clock buffers

Also, implemented more clock gating for power reduction

EV6 Clock Skew

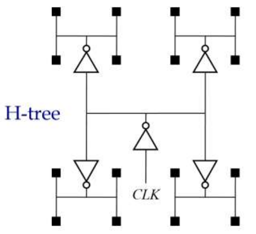


50ps

GCLK Skew
(at Vdd/2 Crossings)

GCLK Rise Times
(20% to 80% Extrapolated to 0% to 100%)

Clock Distribution – H-Tree



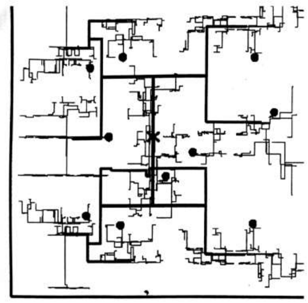
Clock tree minimizing skew

Clock insertion delay to every FF identical (ideally)

Sometimes: work with "**useful (intentional) skew**", e.g. if many FFs have hold violation, to avoid insertion of a large amount of buffers

Equal wire length/number of buffers to get to every location

More realistic H-tree



True H-tree only works for regular structures like FPGAs

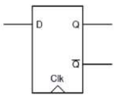
But can still have same delay on all paths from source to end points

[Restle98]

Dealing with Skew and Jitter

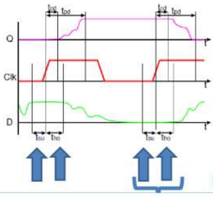
- ❑ Balance clock paths using regular distribution network, such as H-tree
- ❑ Use local clock GRIDS (increased cap and power)
- ❑ Route data and clock in opposite directions to improve hold at the expense of setup.
- ❑ Shield clock wires (minimize capacitive coupling)
- ❑ Use dummy metal density fillers for regular wires
- ❑ Use decoupling capacitors (for stable VDD)
- ❑ Time borrowing (or cycle stealing): long path borrows time from subsequent short path, accomplished using latches
- ❑ "Useful skew" to avoid expensive buffering for hold fix

Additional Timing Parameters of Sequential Circuits



Rising-edge-triggered D-type flip-flop

The value at the input D becomes available at the output Q at the rising edge of the clock



t_{su} **(Set-up time)**: The lapse of time before the active clock edge during which an input is required to assume a fixed logic value of either 0 or 1 at the input of a clocked circuit.

t_{ho} **(Hold time)**: The lapse of time after the active clock edge during which data are required to remain logically unchanged at the input of a clocked circuit.

Data call window: tsu + tho

Timing Quantities (Cont'd) – Seq. Circuits Only

Sequential circuits only

Setup time tsu

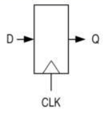
- Time lapse before active clock edge during which input of bistable element (flip-flop, latch, SRAM, ROM, …) needs to settle for correct sampling

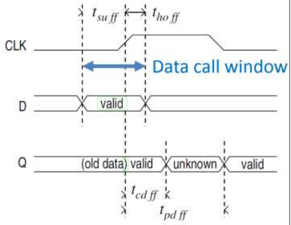
Hold time tho:

- Time lapse after active clock edge during which input data of bistable element needs to remain unchanged

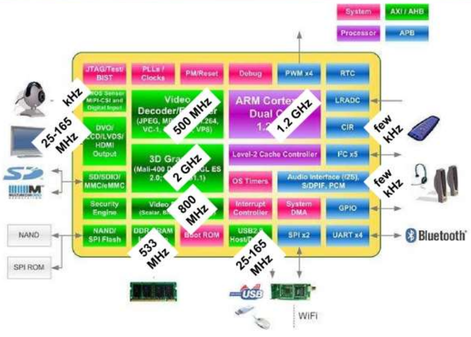
Data call window

- Sum of *tsu* and *tho*
- Typically centered around active (positive) clock edge



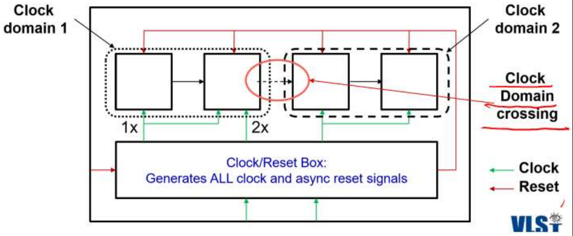


A Modern SoC has Many Clock Domains



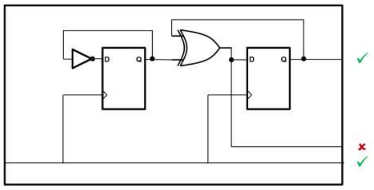
Designs with Multiple Clocks

- **Clock domain**: (Sub)circuits in which all clock signals maintain fixed frequency and phase relationship
  - A single clock domain may use multiple divided clocks
- **Clock box**: top-level entity that generates all clock signals



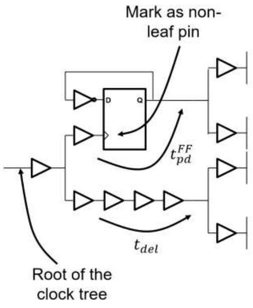
Clock Dividers

- Derive a slow clock from a fast clock by dividing it by an integer number (counter)
- Remember: all clocks (including divided clocks) need to be free of hzards and glitches
  - Must assume that any logic can prodice glitches
  - Generated (divided) clocks MUST come directly from a FF output



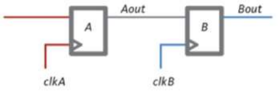
Clock Dividers in the Clock Tree

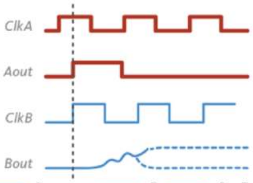
- Clock and derived clock must be phase aligned at the clock roots
- Clock divider has a delay (FF propagation delay)
  - Need to consider clock divider during clock-tree generation
- Modern clock tree synthesis tools can trace through clock dividers



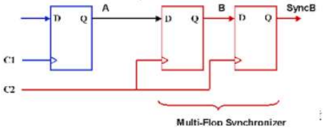
Crossing Clock Domains

- Arbitrary phase relationship
  - Timing violations will happen
  - Metastability



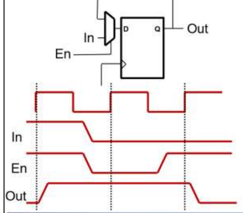


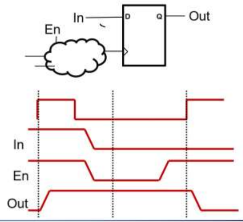
- Double sampling:
  - Provides an entire cycle for to resolve metastability



Clock Gating

- Flip-Flops with ENABLE: prevent FlipFlop(s) from capturing new data
  - Functional: preserve content over multiple cycles
  - Power savings: avoid activity in a block that is inactive

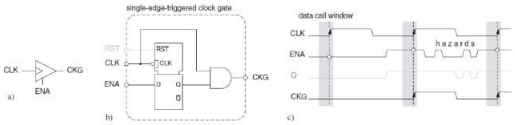




Clock Gating

- Advantages:
  - Saves a multiplexer per FlipFlop with Enable (Area, Delay, and Power advantage)
  - Avoids activity on the clock net leading to the FlipFlop
  - Avoids activity on the FlipFlops clock pin reducing internal power consumption
- Disadvantages:
  - Need for additional logic to suppress the clock while FlipFlop is disabled (area and power penalty)
  - Need to ensure that the clock signal is free of glitches

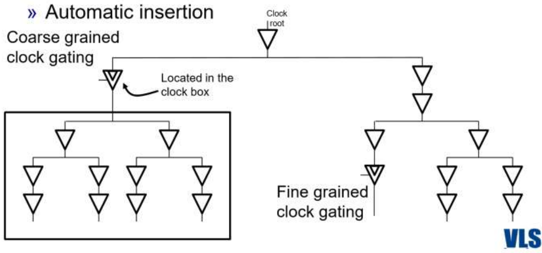
- A safe strategy to realize clock gating



  - Latch on the Enable signal shields glitches during the sensitive period of the clock
  - Implemented with individual cells: some timing constraints need to be observed
  - Often realized as single dedicated library cell

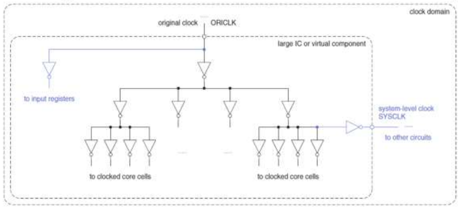
Clock Gating

- Coarse grained: disable entire blocks of the design
  - Inserted manually
- Fine grained: enable for small groups of FlipFlops
  - Automatic insertion



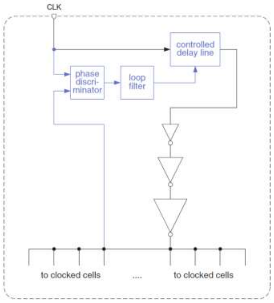
Improving IO Timing

- Chip provides a clock output that is aligned with the clock at the leaves of the FlipFlops
  - Output of the clock output pad is declared as clock leaf
  - Delayed clock is used as a reference for rest of system



Improving IO Timing

- Delay locked loop (DLL)
  - Generates a phase shifted clock such that the reference input is phase aligned with the input clock
  - Clock reference is taken from the leaves of the clock tree
- Internal clock at the leaves is aligned with clock input


CLK
controlled delay line
phase discri-minator
loop filter
to clocked cells
....
to clocked cells