



School of Engineering & Technology  
Department of Electronics Engineering  
Pondicherry University  
Pondicherry-14

# UNIT I

# Low Power VLSI Design

By  
**Dr. R. Nakkeeran**  
Associate Professor

# Low Power Design – An Emerging Discipline

- Historical figure of merit for VLSI design – performance (circuit speed) and chip area (circuit density/cost)
- Power dissipation is now an important metric in VLSI design
  - No single major source for power savings across all design levels – Required a new way of THINKING!!!
  - Companies lack the basic power-conscious culture and designers need to be educated in this respect
- Overall Goal – To reduce power dissipations but maintaining adequate throughput rate

# Need for Low Power VLSI Chips

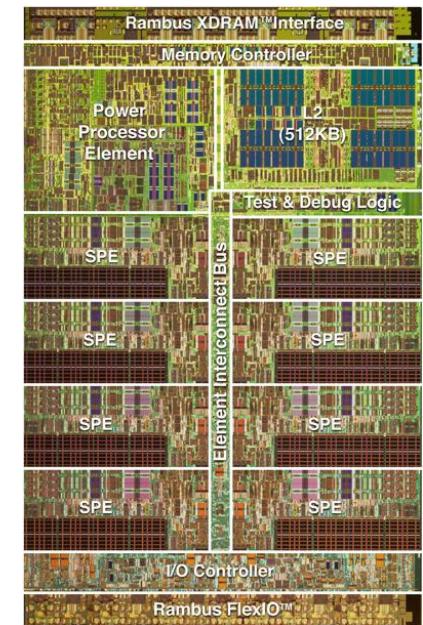
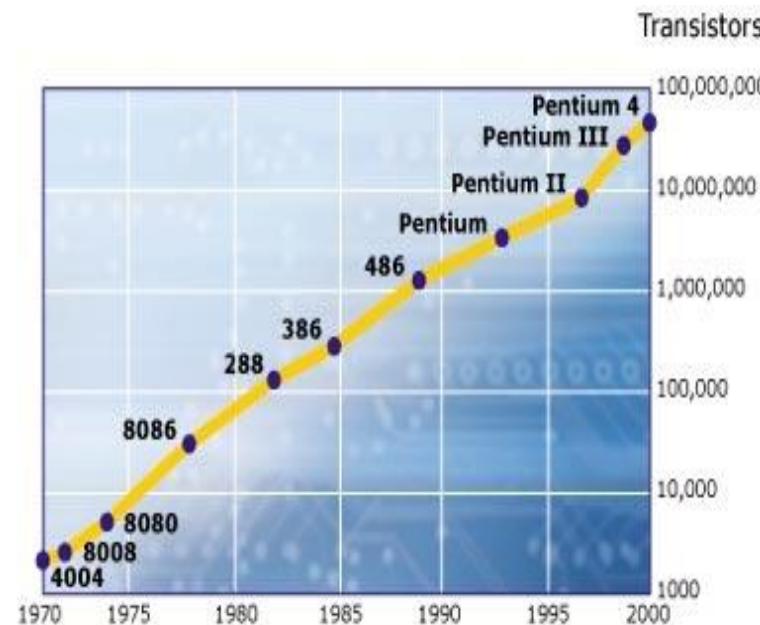
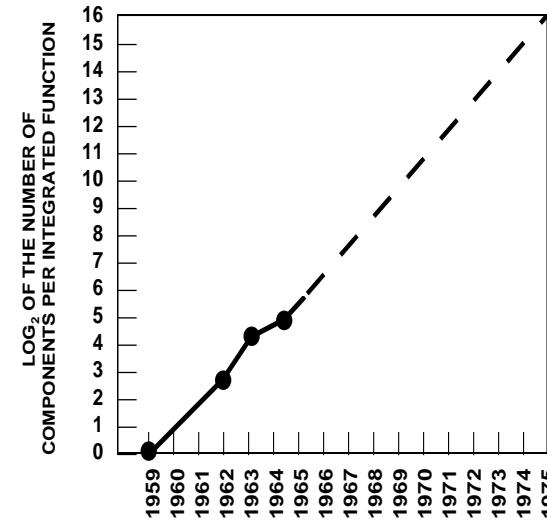
- Power dissipation **was** neglected due to
  - Low device density
  - Low operating frequency
- **Now** it is important issue due to
  - High device density
  - High operating frequency
  - Proliferation of portable consumer electronics
  - Concerns on Environments and energy sources

# Competitive Reasons – Low Power

- Battery Powered Systems – Extended Battery Life and reduce weight and size.
- High-Performance Systems
  - Cost
    - Package (chip carrier, heat sink, card slots, plenum, ...)
    - Power Systems (supplies, distribution, regulators, ...)
    - Fans (noise, power, reliability, area, ...)
    - Operating cost to customer – Energy Star issue.
  - Reliability
    - Failure rate increase by 4X for  $T_j$  @ 110C vs 70C
    - Mission critical operation at 100C
  - Size and Weight – Product footprint (office and deskspace)  
pp. 4

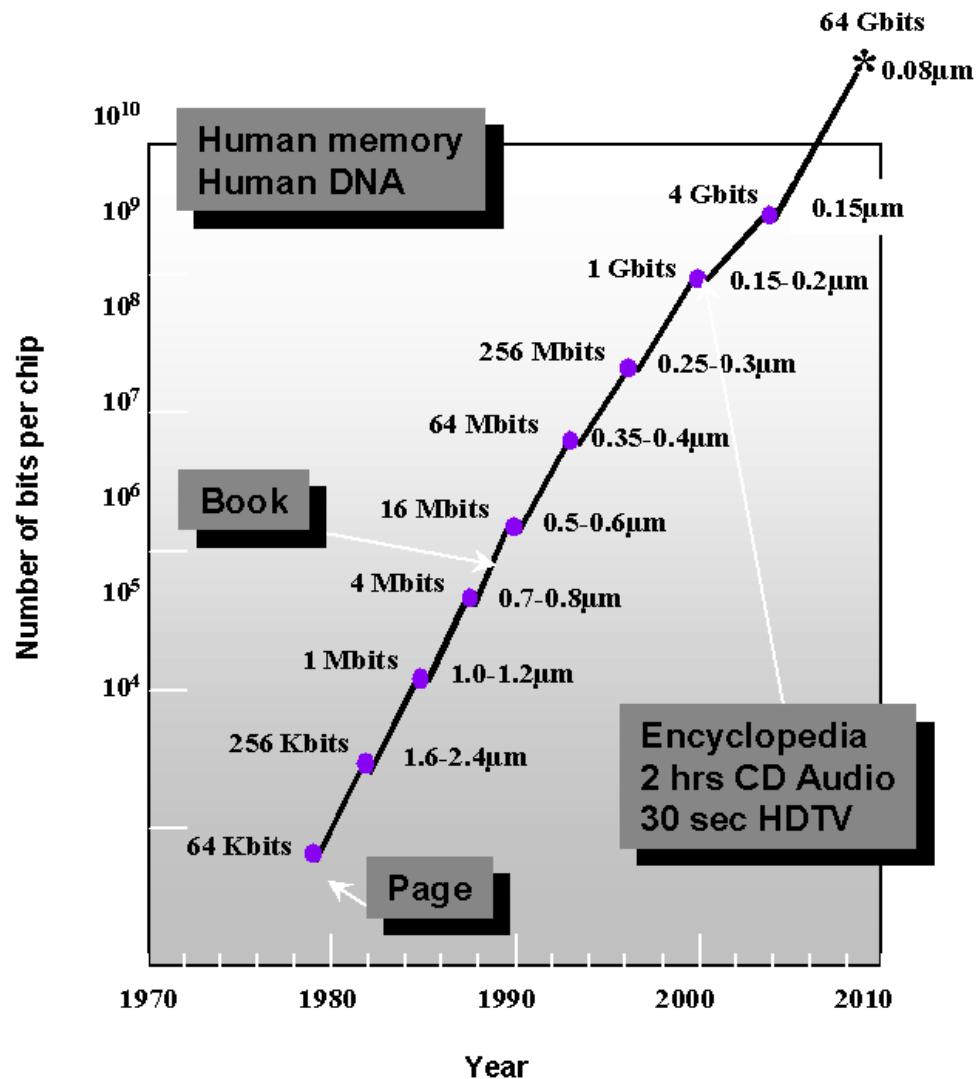
# Moore's Law

- In 1965, Gordon Moore noted that the number of transistors on a chip doubled every 18 to 24 months.
- He made a prediction that semiconductor technology will double its effectiveness every 18 months

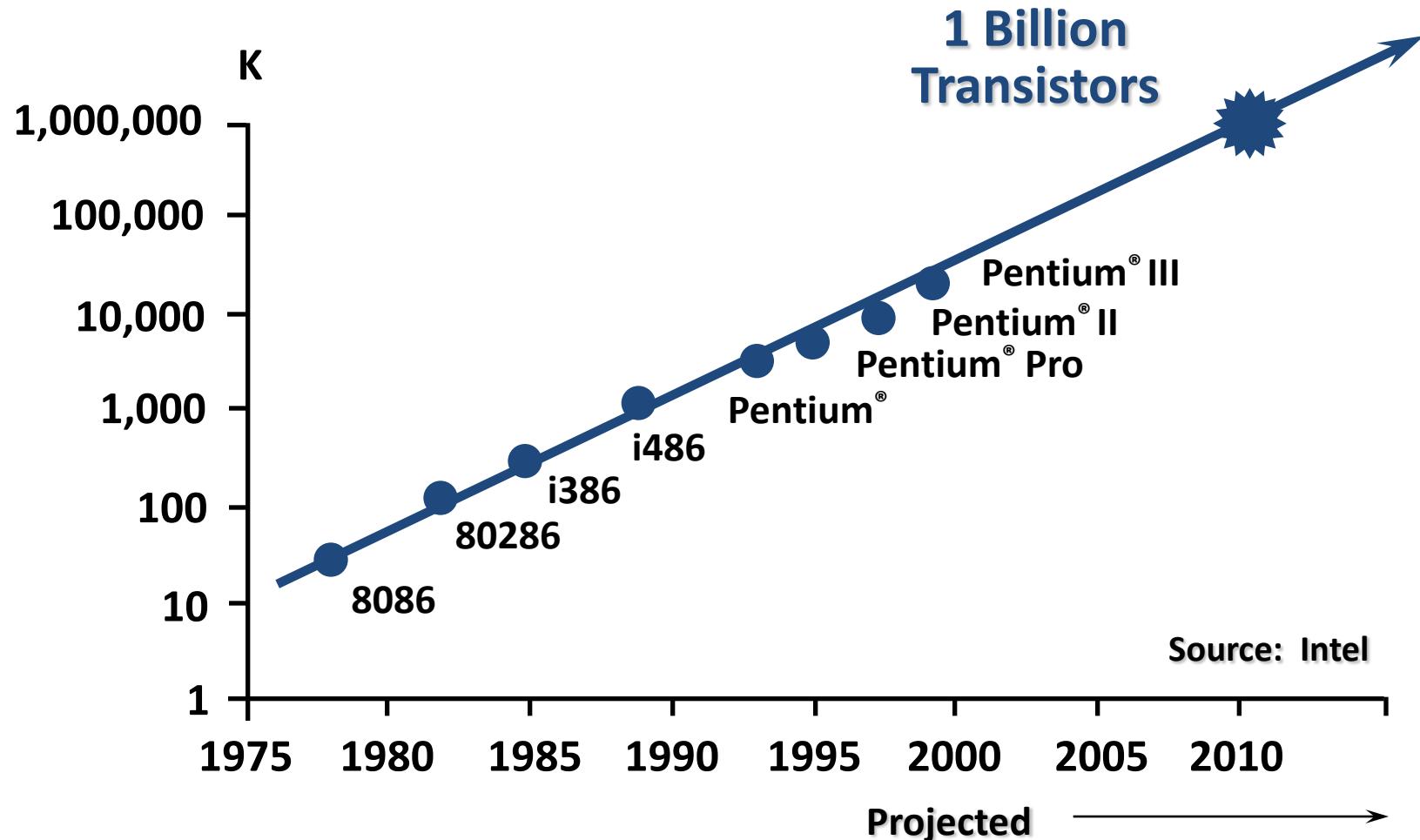


234M transistors in  
die size of 221 mm<sup>2</sup>

# Evolution in Complexity

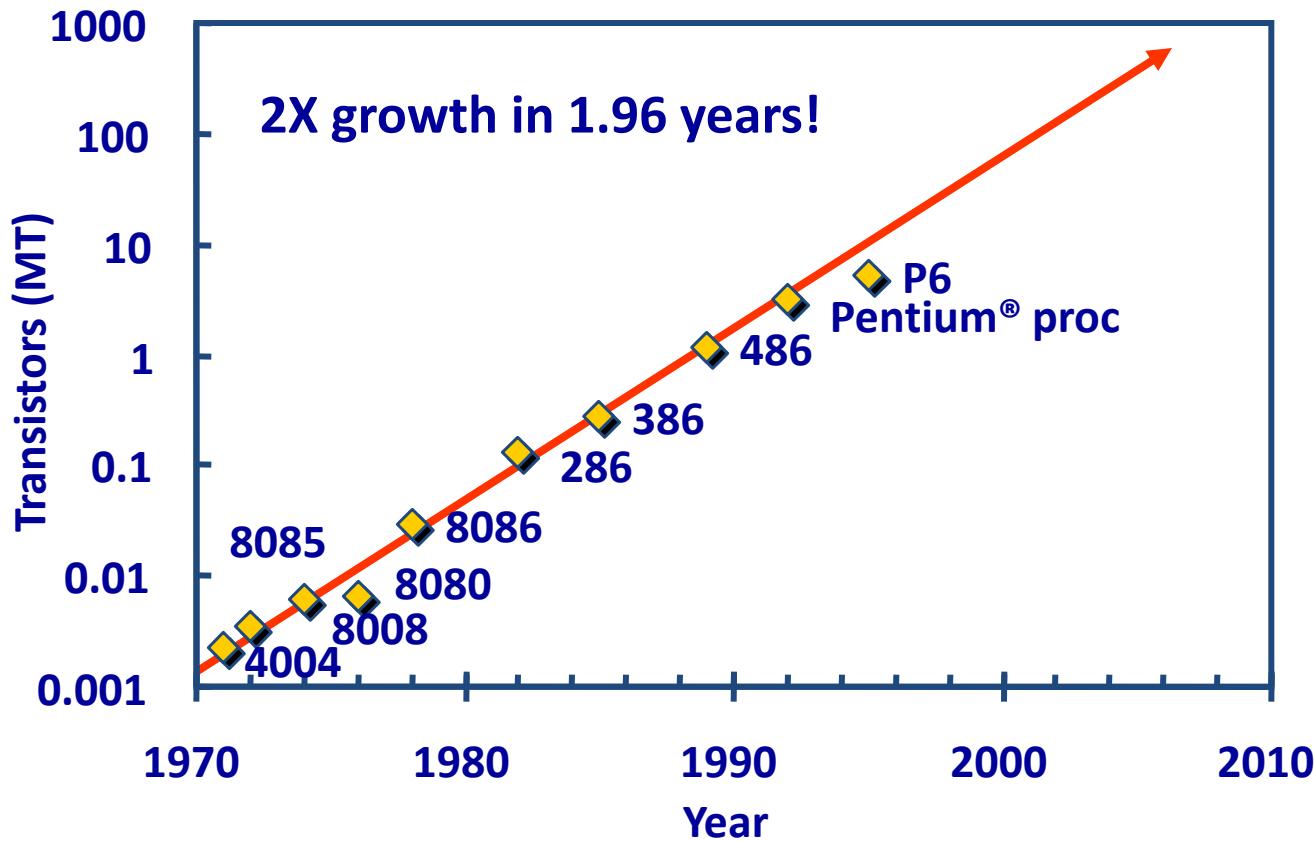


# Transistor Counts



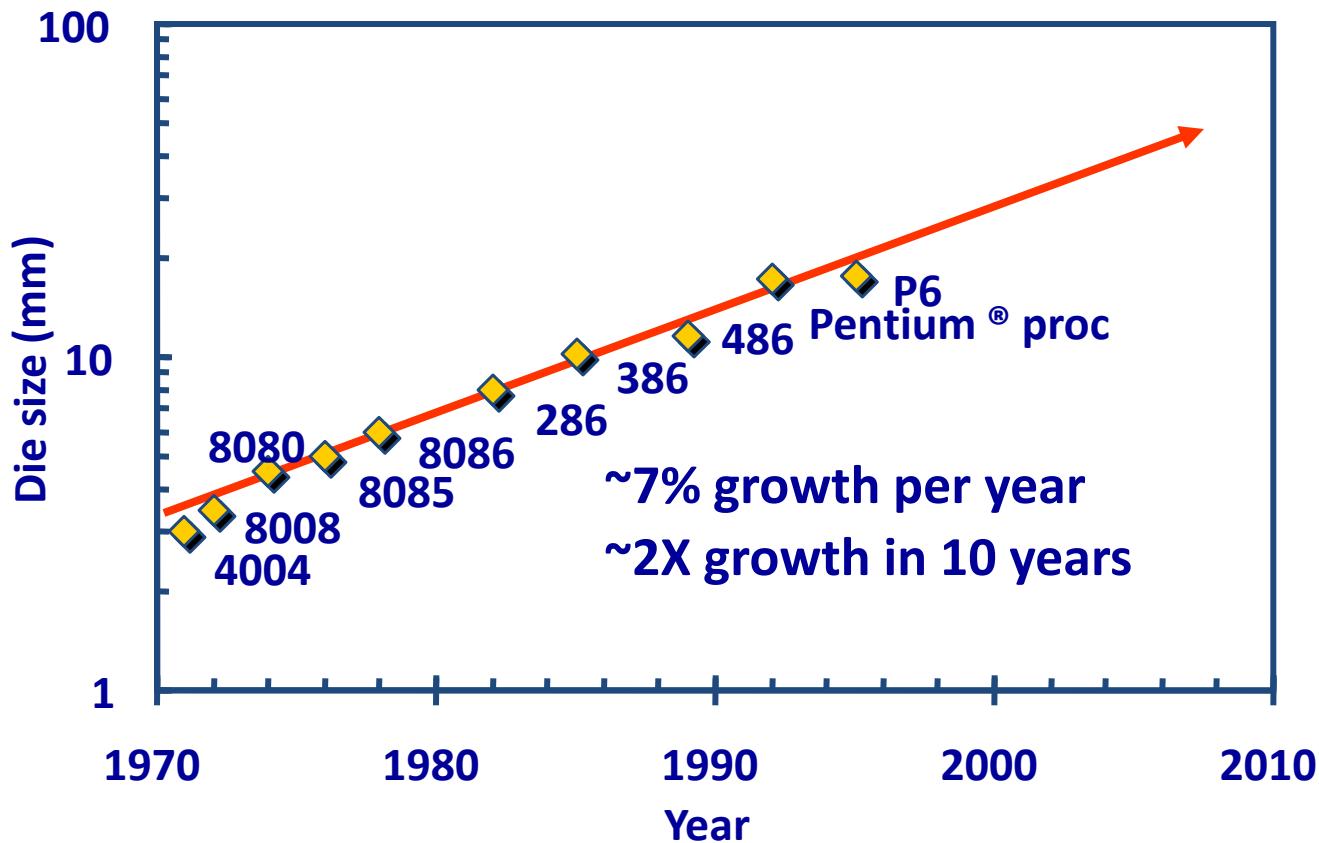
Courtesy, Intel

# Moore's law in Microprocessors



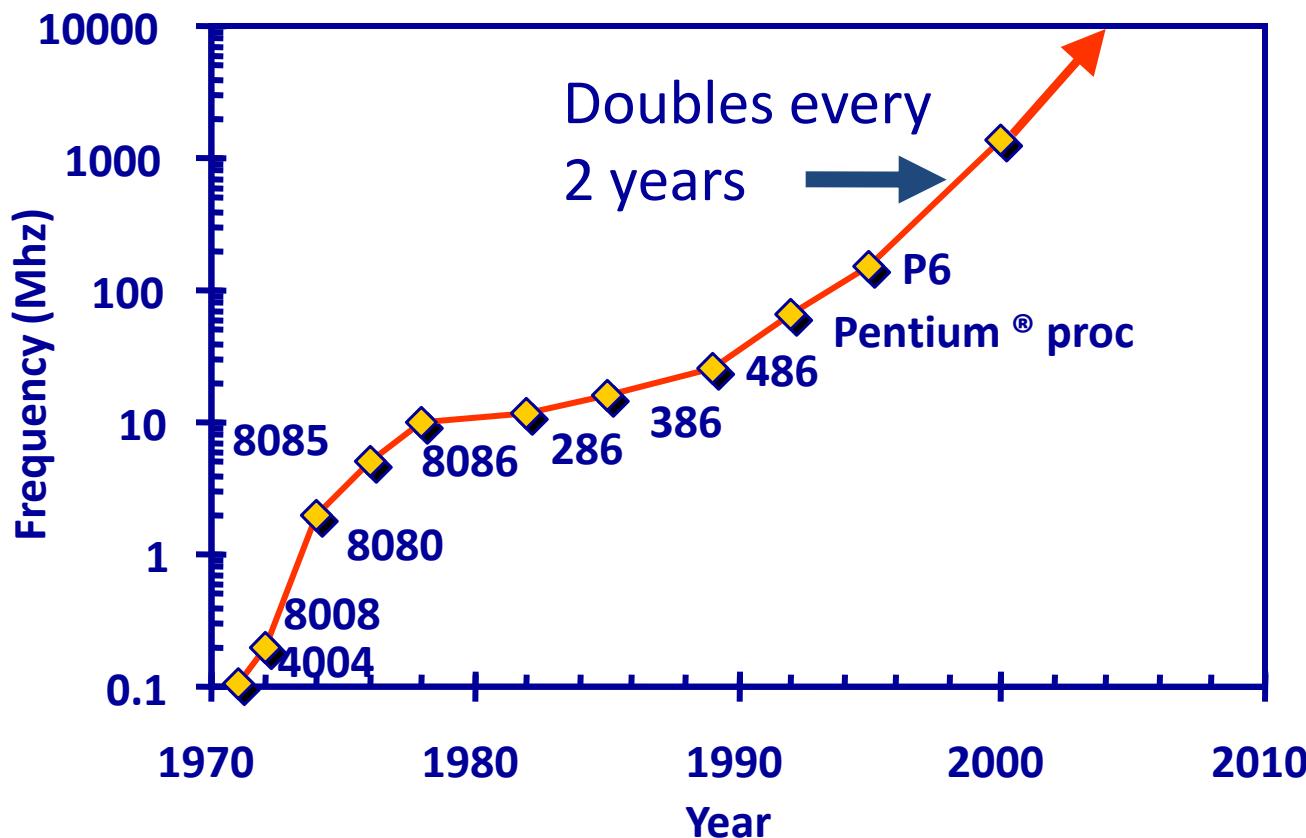
Transistors on Lead Microprocessors double every 2 years

# Die Size Growth



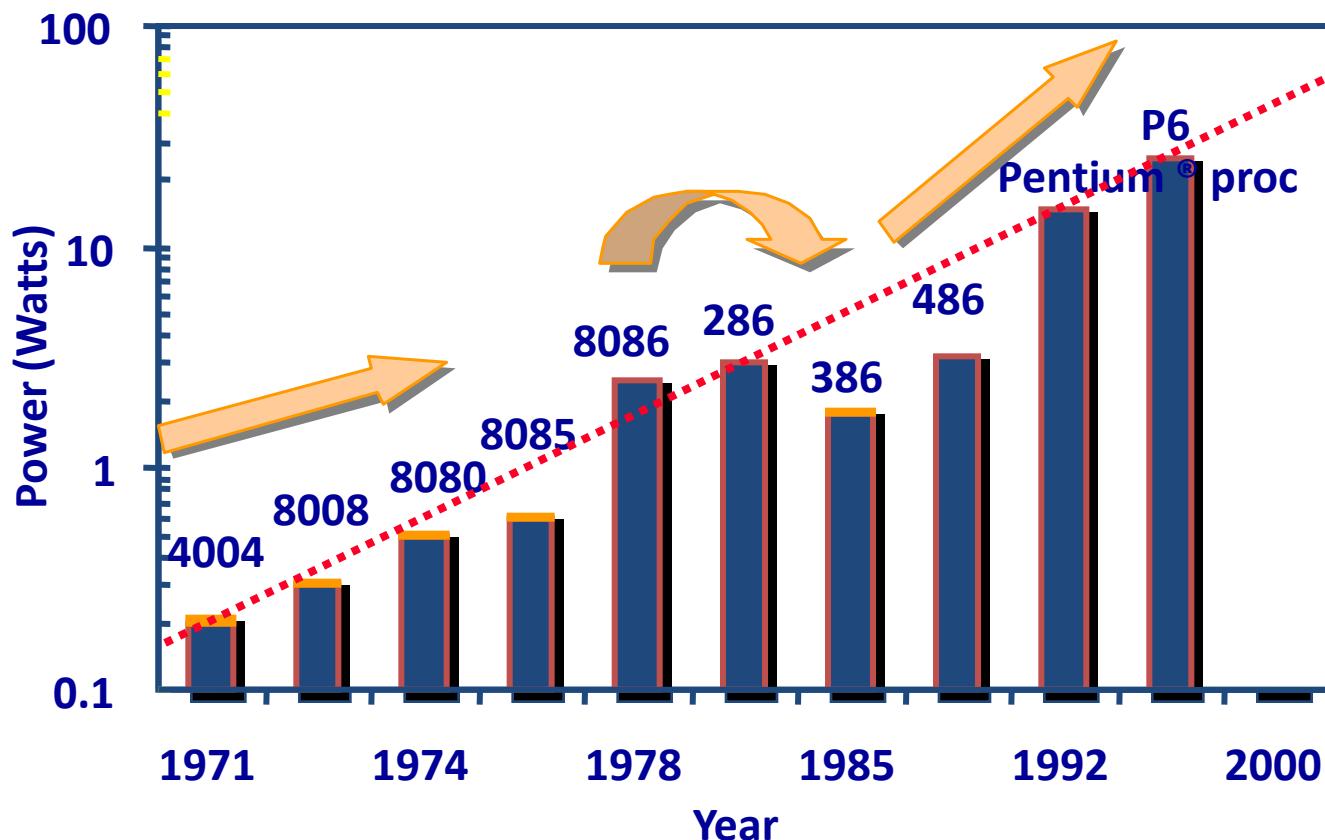
Die size grows by 14% to satisfy Moore's Law

# Frequency



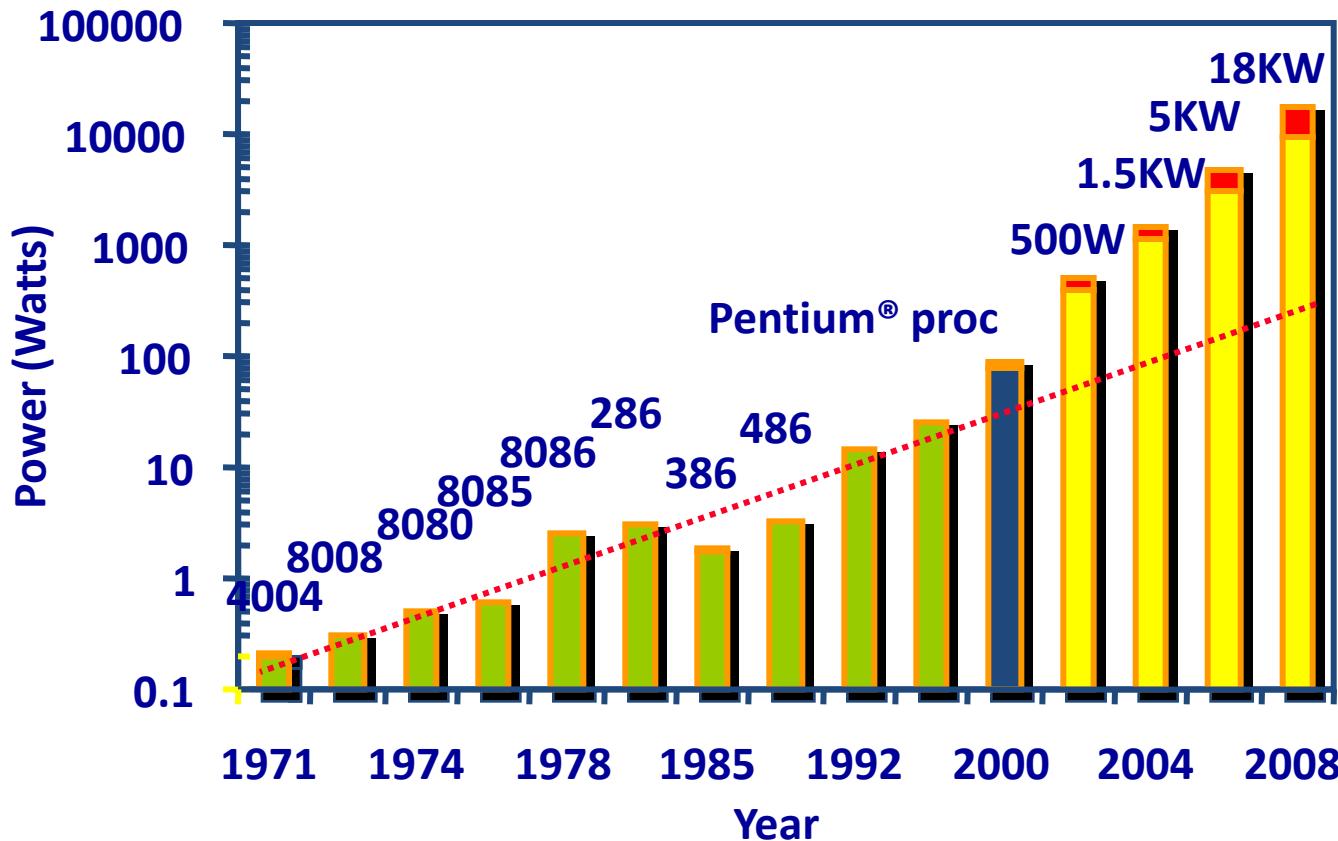
Lead Microprocessors frequency doubles every 2 years

# Power Dissipation



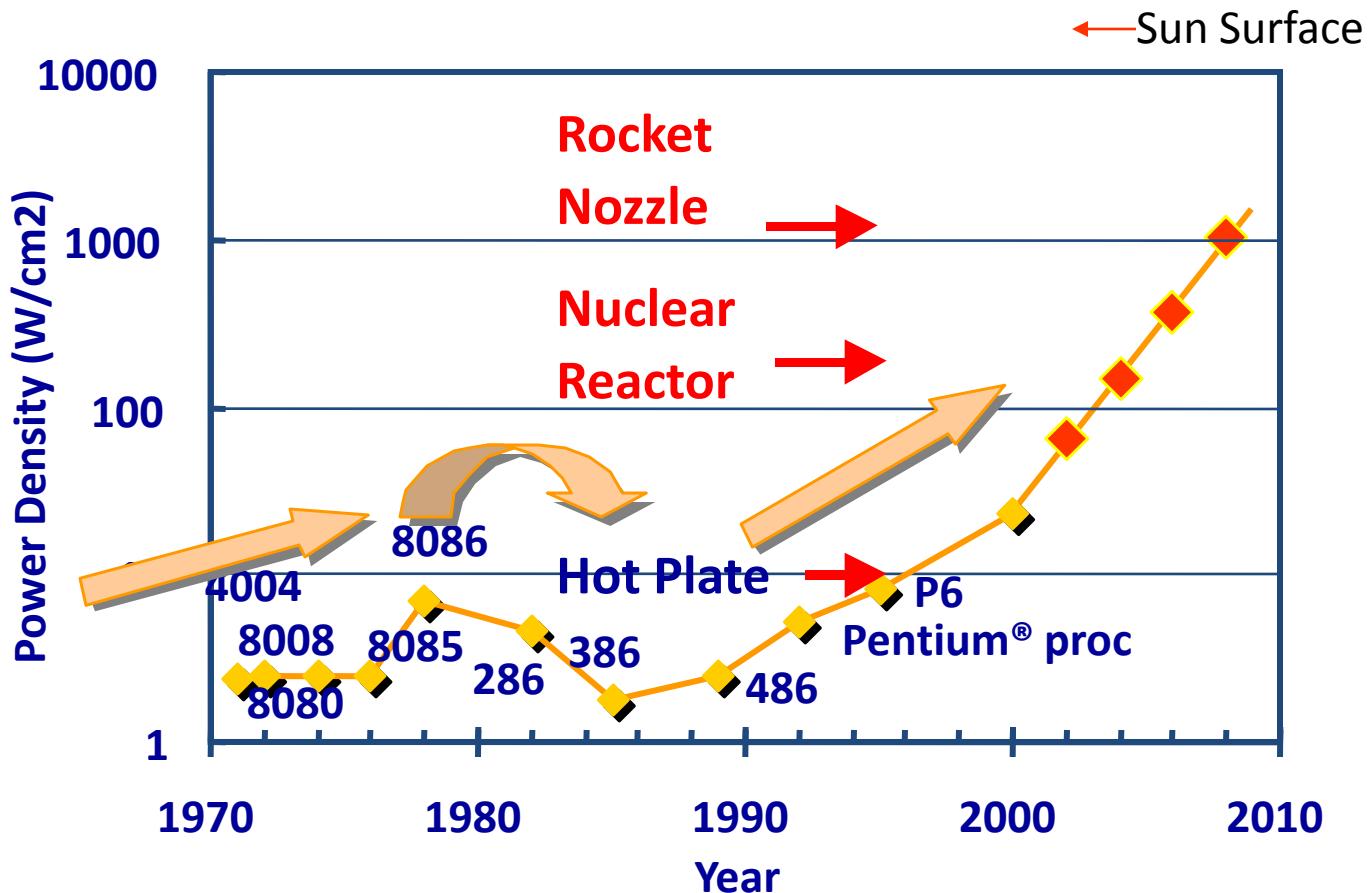
Lead Microprocessors power continues to increase

# Power will be a major problem



Power delivery and dissipation will be prohibitive

# Power density



Power density too high to keep junctions at low temp

# Not Only Microprocessors

Cell  
Phone

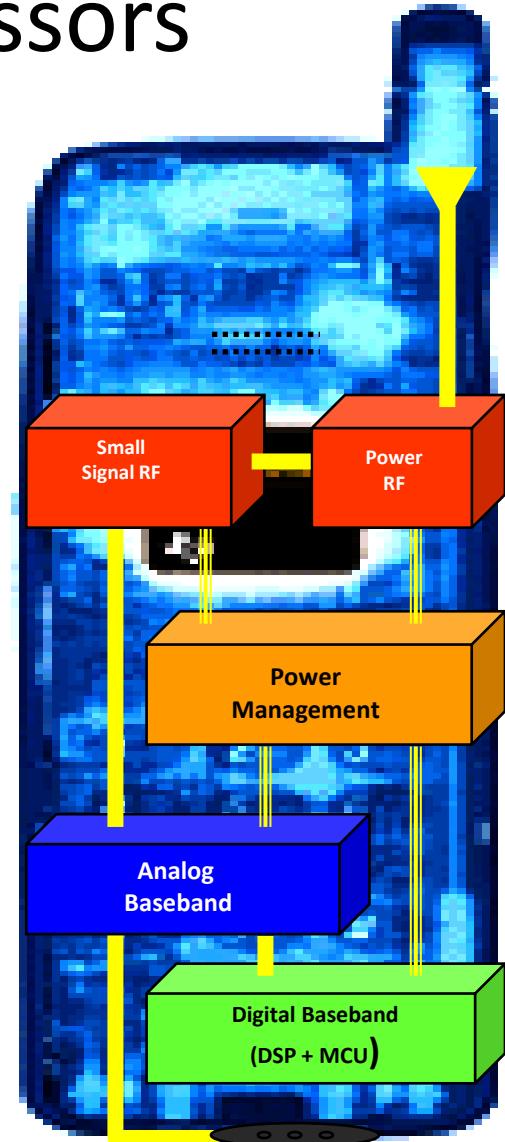


Digital Cellular Market  
(Phones Shipped)

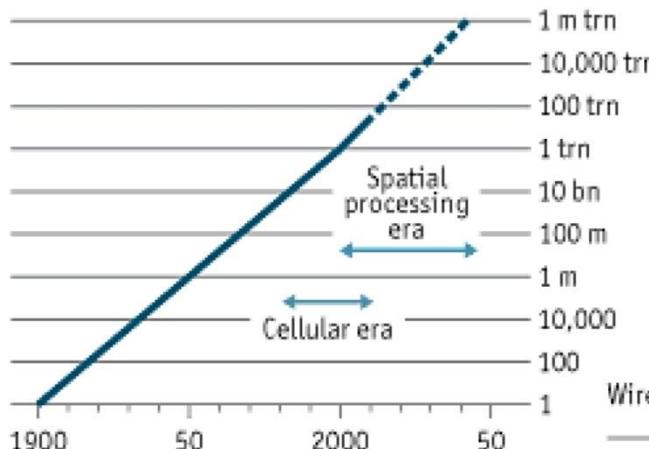
1996 1997 1998 1999 2000

Units 48M 86M 162M 260M 435M

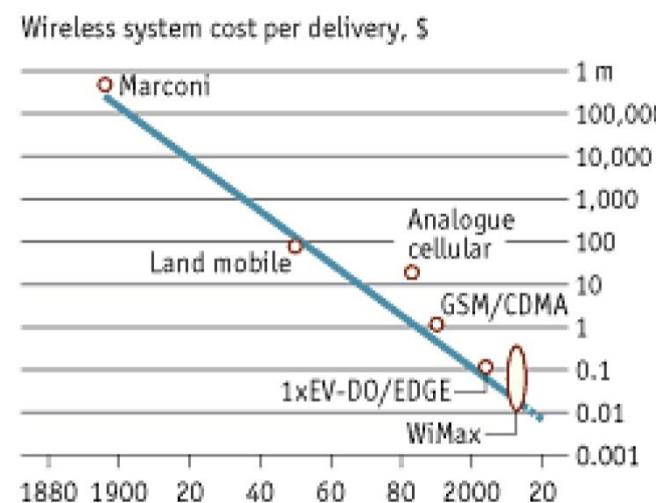
(data from Texas Instruments)



Increase in efficiency of wireless spectrum



"Spectral Efficiency":  
More bits/m<sup>3</sup>

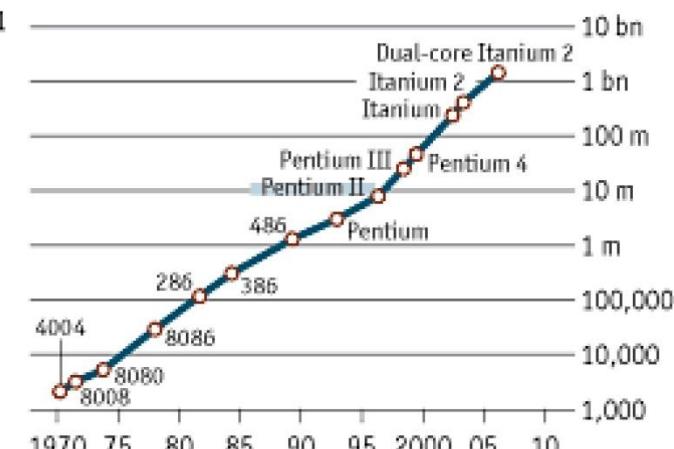


Rapidly declining  
system cost

# Important (Wireless) Technology Trends

Rapidly increasing  
transistor density

Moore's law, transistors in Intel chips



Sources: Martin Cooper (ArrayComm); Intel

# Important (Wireless) Technology Trends

## Speaking in tongues

Main two-way wireless technologies\*

	Data rate per second	Range	Cost†
Mobile WiMax	15Mb	5km	\$8 in 2008
3G cellular (HSDPA/LTE)	14Mb	10km	\$6
2G cellular (GSM/CDMA)	400k	35km	\$5
Wi-Fi	54Mb	50-100m	\$4
Bluetooth	700k	10m	\$1
Zigbee	250k	30m	\$4
UWB	~400Mb	5-10m	\$5
RFID	1-200k	0.01-10m	4 cents

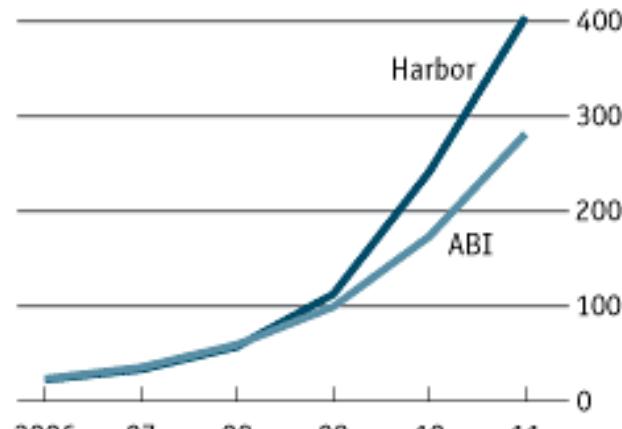
\*Typical performance; actual figures vary  
†Approx. device-chip cost at high volume

Sources: William Webb; Cambridge Consultants; OECD; Pyramid Research; Nokia; TI; CSR; Ember; Hitachi

Speed-Distance-Cost  
Tradeoffs

## On the right wavelength

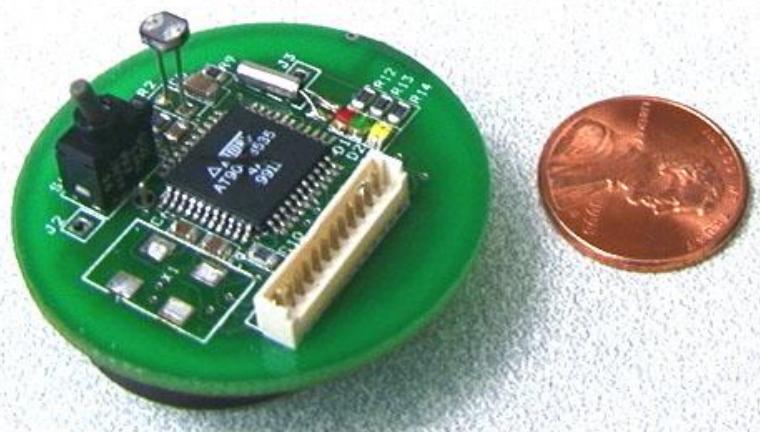
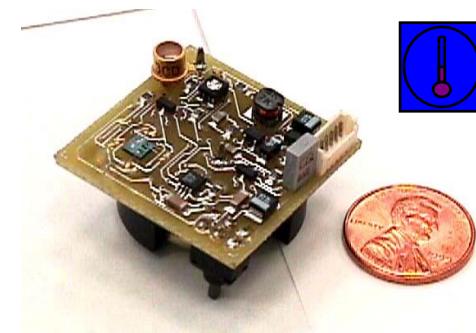
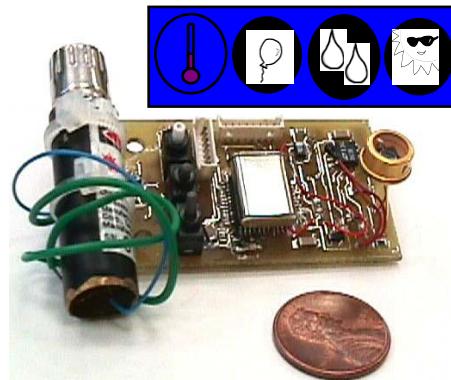
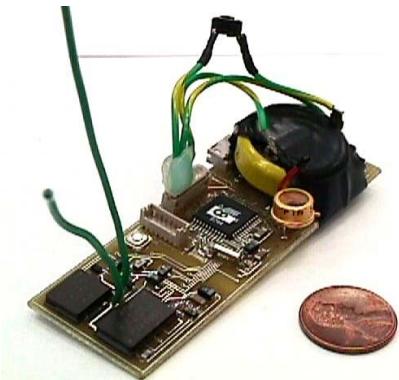
Wireless M2M devices and sensors, forecasts  
Shipments, m



Sources: Harbor Research; ABI Research

Rapid Growth: Machine-to-Machine Devices (mostly sensors)

# In the Physical World: Sensor Devices



# Low Power Design

- Source of power dissipation

- $P = P_{switching} + P_{short-circuit} + P_{leakage} + P_{static}$

- Definitions:

- Switching power               $P = CV^2f\alpha$
    - Short circuit power           $P = I_{sc}V$
    - Leakage power                 $P = I_{leakage}V$
    - Static power                  $P = I_{static}V$
    - $\alpha$  : switching activity factor

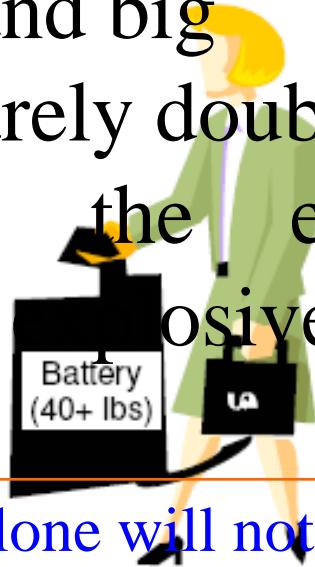
- Low power design would look at the trade-offs of the above issues

# Why Worry About Power?

- Portable devices:
  - Handhelds, laptops, phones, MP3 players, cameras, ... all need to run for extended periods on small batteries without recharging
  - Devices that need regular recharging or large heavy batteries will lose out to those that don't.
- Power consumption important even in “tethered” devices
  - System cost tracks power consumption:
    - Power supplies, distribution, heat removal
    - Power conservation, environmental concerns
- In 10 years, have gone from minimal consideration of power consumption to (designing with power consumption as a primary design constraint!)

# Battery

- Portable consumer electronics powered by battery
- Battery is heavy and big
- Energy density barely doubles in several years
- Safety concern: the energy density is approaching that of explosive chemicals



The battery technology alone will not solve the low power problem

# Basics

- Power supply provides energy for charging and discharging wires and transistor gates. The energy supplied is stored & then dissipated as heat.

$$P = dw / dt$$

*Power: Rate of work being done w.r.t time  
Rate of energy being used*

$$P = E / \Delta t$$

Unit: *Watts = Joules/seconds*

- If a differential amount of charge  $dq$  is given a differential increase in energy  $dw$ , the potential of the charge is increased by:
- By definition of current:  $I = dq / dt$        $V = dw / dq$

$$dw / dt = \frac{dw}{dq} \times \frac{dq}{dt} = P = V \times I$$

A very practical formulation!

$$w = \int_{-\infty}^t P dt \quad \text{Total energy}$$

# Basics

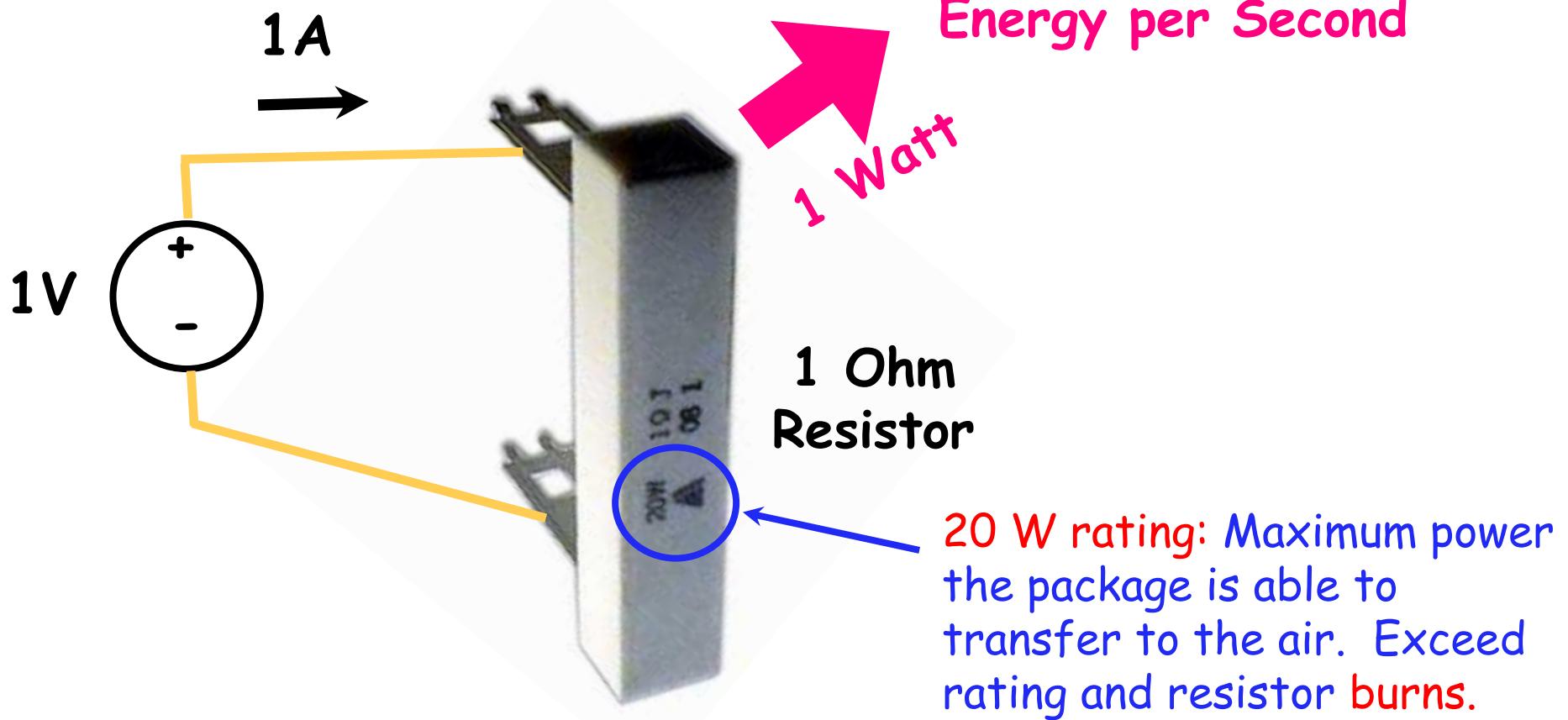
- **Warning!** In everyday language, the term “power” is used incorrectly in place of “energy”
- Power is **not** energy
- Power is **not** something you can run out of
- Power can **not** be lost or used up
- It is **not** a thing, it is merely a rate
- It can **not** be put into a battery any more than velocity can be put in the gas tank of a car

This is how electric tea pots work ...

Heats 1 gram of water  
0.24 degree C

0.24 Calories per Second

1 Joule of Heat  
Energy per Second



# Cooling an iPod nano ...



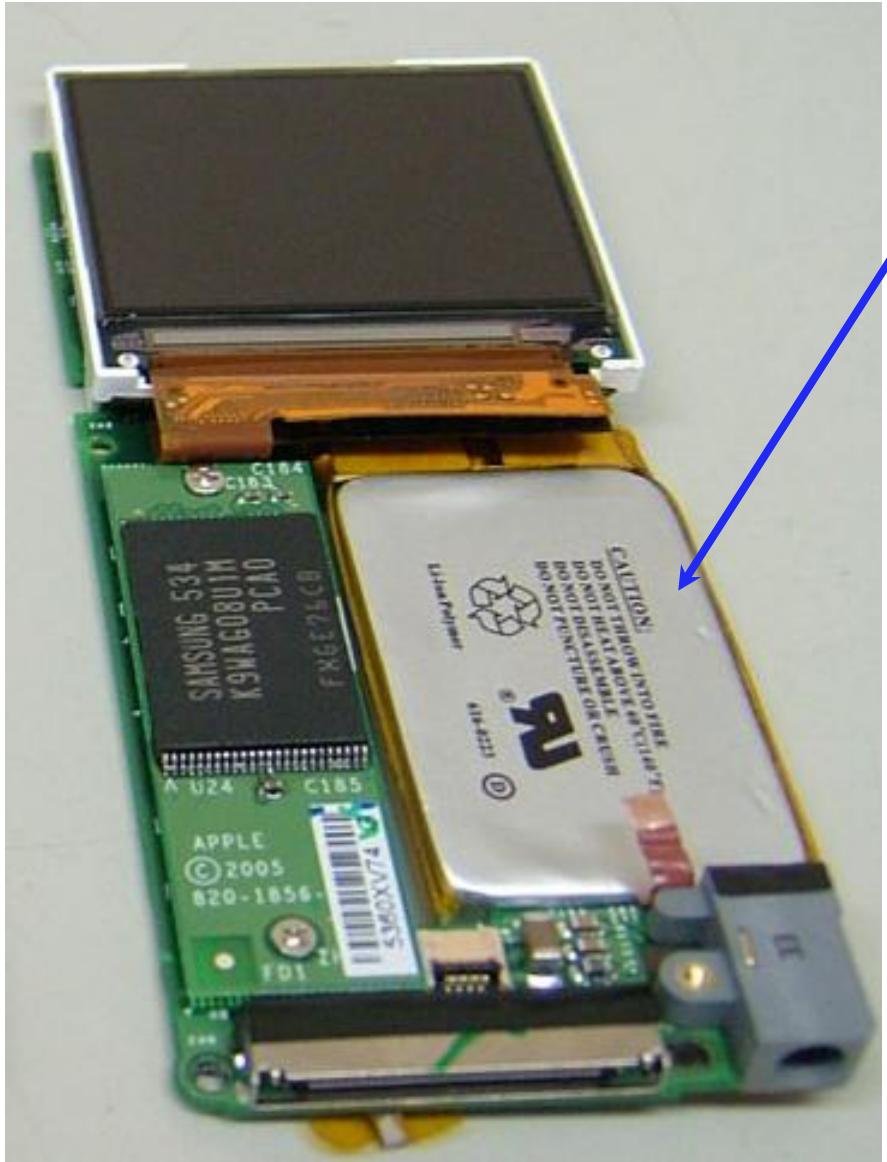
Like a resistor, iPod relies on passive transfer of heat from case to the air

Why? Users don't want fans in their pocket ...

To stay "cool to the touch" via passive cooling,  
**power budget of 5 W**

If iPod nano used 5W all the time, its battery would last 15 minutes ...

# Powering an iPod nano (2005 edition)



Battery has 1.2 W-hour rating: Can supply 1.2 W of power for 1 hour

$$1.2 \text{ W} / 5 \text{ W} = 15 \text{ minutes}$$

More W-hours require bigger battery and thus bigger "form factor" -- it wouldn't be "nano" anymore!

Real specs for iPod nano :  
14 hours for music,  
4 hours for slide shows

85 mW for music

300 mW for slides

# How Do We Measure and Compare Power Consumption?

- One popular metric for microprocessors is: MIPS/watt
  - MIPS, millions of instructions per second
    - Typical modern value?
  - Watt, standard unit of power consumption
    - Typical value for modern processor?
  - MIPS/watt reflects tradeoff between performance and power
  - Increasing performance requires increasing power
  - Problem with “MIPS/watt”
    - MIPS/watt values are typically not independent of MIPS
      - Techniques exist to achieve very high MIPS/watt values, but at very low absolute MIPS (used in watches)
    - Metric only relevant for comparing processors with a similar performance
  - One solution,  $\text{MIPS}^2/\text{watt}$ . Puts more weight on performance

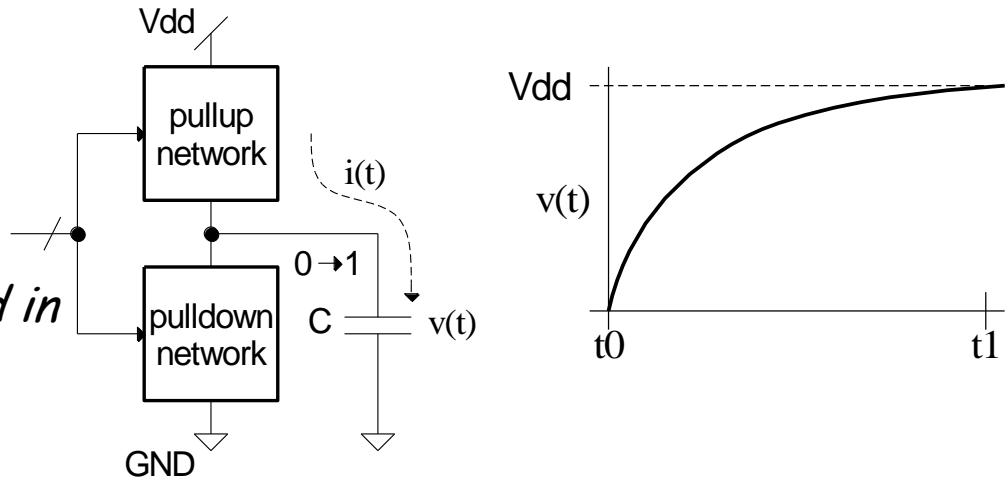
# Metrics

- How does MIPS/watt relate to energy?
- Average power consumption = energy / time
  - MIPS/watt = instructions/sec / joules/sec = instructions/joule
  - Equivalent metric (reciprocal) is energy per operation (E/op)
- E/op is more general - applies to more than processors
  - also, usually more relevant, as batteries life is limited by total energy draw
  - This metric gives us a measure to use to compare two alternative implementations of a particular function

# Power in CMOS

Switching Energy:  
energy used to  
switch a node

*Calculate energy dissipated in  
pullup:*



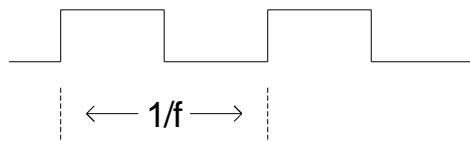
$$\begin{aligned}
 E_{sw} &= \int_{t_0}^{t_1} P(t) dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot i(t) dt = \int_{t_0}^{t_1} (V_{dd} - v) \cdot c (dv/dt) dt = \\
 &= cV_{dd} \int_{t_0}^{t_1} dv - c \int_{t_0}^{t_1} v \cdot dv = cV_{dd}^2 - 1/2cV_{dd}^2 = \boxed{1/2 cV_{dd}^2}
 \end{aligned}$$

Energy supplied      Energy stored      Energy dissipated

An equal amount of energy is dissipated on pulldown

# Switching Power

- Gate power consumption:
  - Assume a gate output is switching its output at a rate of:



$\alpha \cdot f$   
*activity factor*      *clock rate*  
(probability of switching on  
any particular clock period)

$$P_{avg} = E/\Delta t = \text{switching rate} \cdot E_{sw}$$

Therefore:

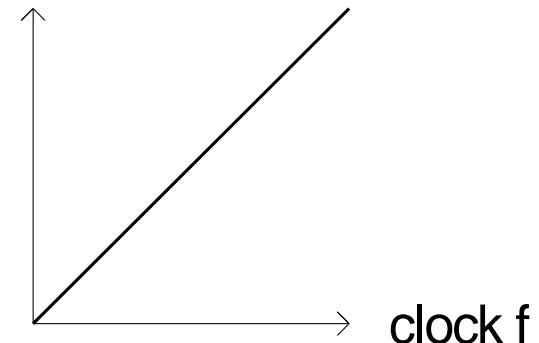
$$P_{avg} = \alpha \cdot f \cdot 1/2 c V_{dd}^2$$

P<sub>avg</sub>

⌘ Chip/circuit power consumption:

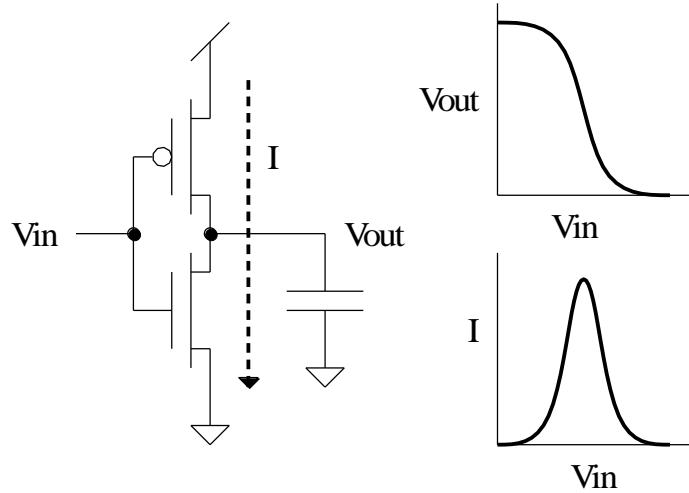
$$P_{avg} = n \cdot \alpha_{avg} \cdot f \cdot 1/2 c_{avg} V_{dd}^2$$

*number of nodes (or gates)*



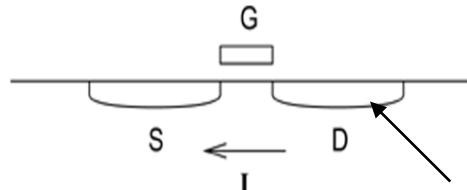
# Other Sources of Energy Consumption

- “Short Circuit” Current:

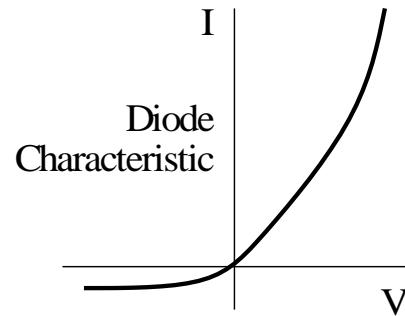


10-20% of total chip power

## ⌘ Junction Diode Leakage :



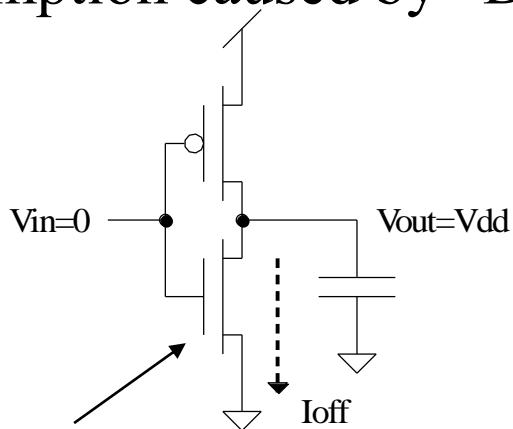
Transistor drain regions  
“leak” charge to substrate.



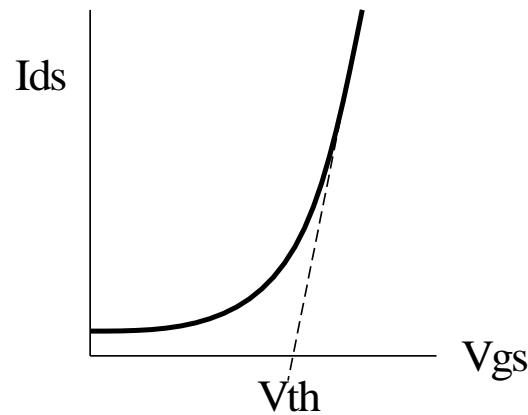
~1nWatt/gate  
few mWatts/chip

# Other Sources of Energy Consumption

- Consumption caused by “DC leakage current” ( $I_{ds}$  leakage):



Transistor s/d conductance  
never turns off all the way



*Low voltage* processes much worse

- This source of power consumption is becoming increasingly significant as process technology scales down
- For 90nm chips around 10-20% of total power consumption
- Estimates put it at up to 50% for 65nm

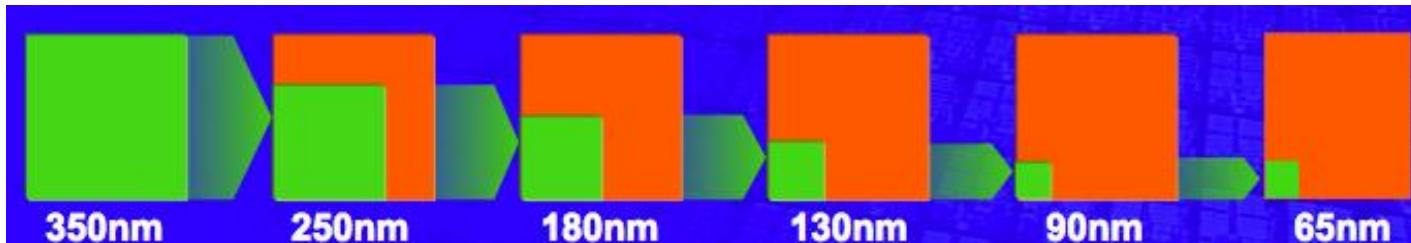
# Controlling Energy Consumption: *What Control Do You Have as a Designer?*

- Largest contributing component to CMOS power consumption is switching power:

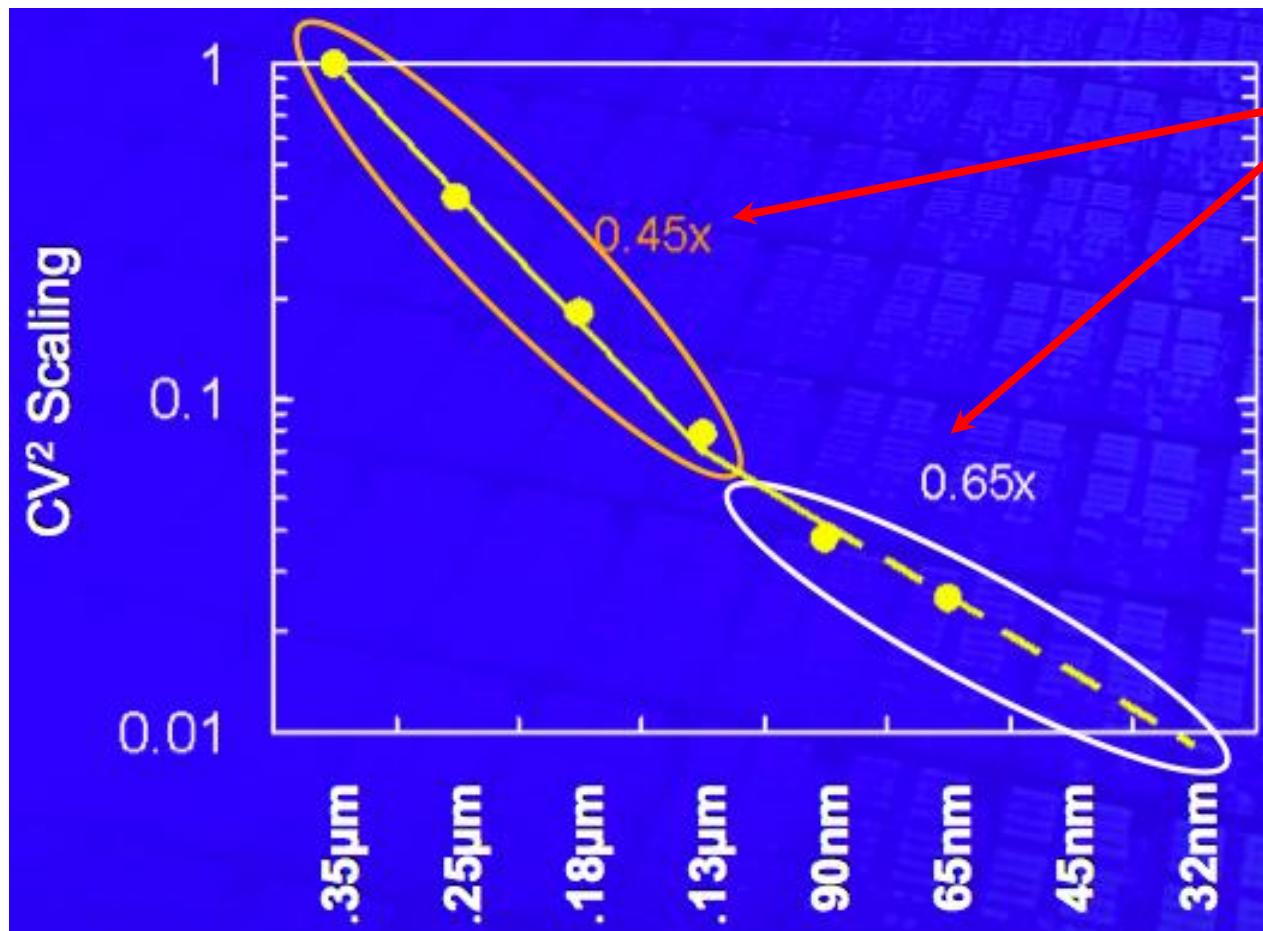
$$P_{avg} = n \cdot \alpha_{avg} \cdot f \cdot 1/2 c_{avg} V_{dd}^2$$

- Factors influencing power consumption:
  - n: total number of nodes in circuit
  - $\alpha$ : activity factor (probability of each node switching)
  - f: clock frequency (does this effect energy consumption?)
  - $V_{dd}$ : power supply voltage
- What control do you have over each factor?
- How does each effect the total Energy?

# Scaling Switching Energy per Gate



Moore's Law  
at work ...



Due to  
reduced V and  
C (length and  
width of Cs  
decrease, but  
plate distance  
gets smaller)

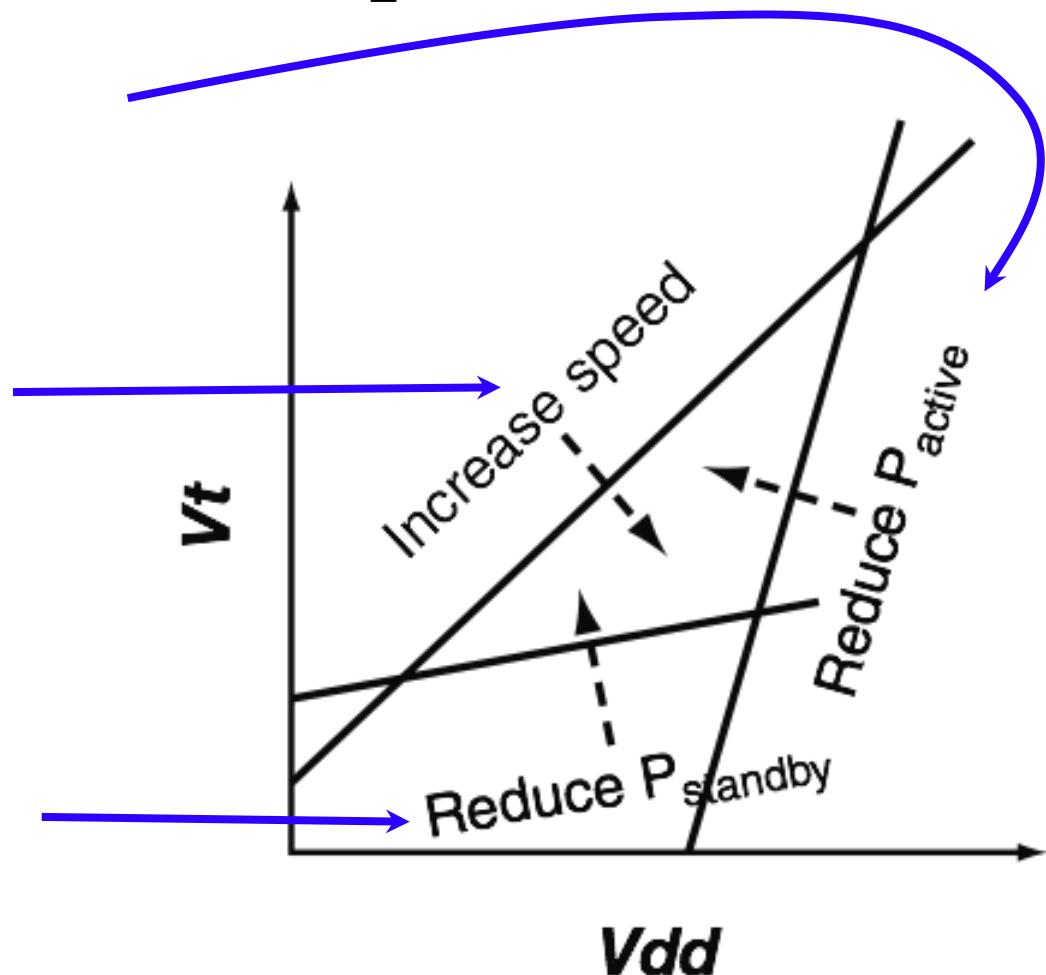
Recent slope  
reduced  
because V is  
scaled less  
aggressively

# Device Engineers Trade Speed and Power

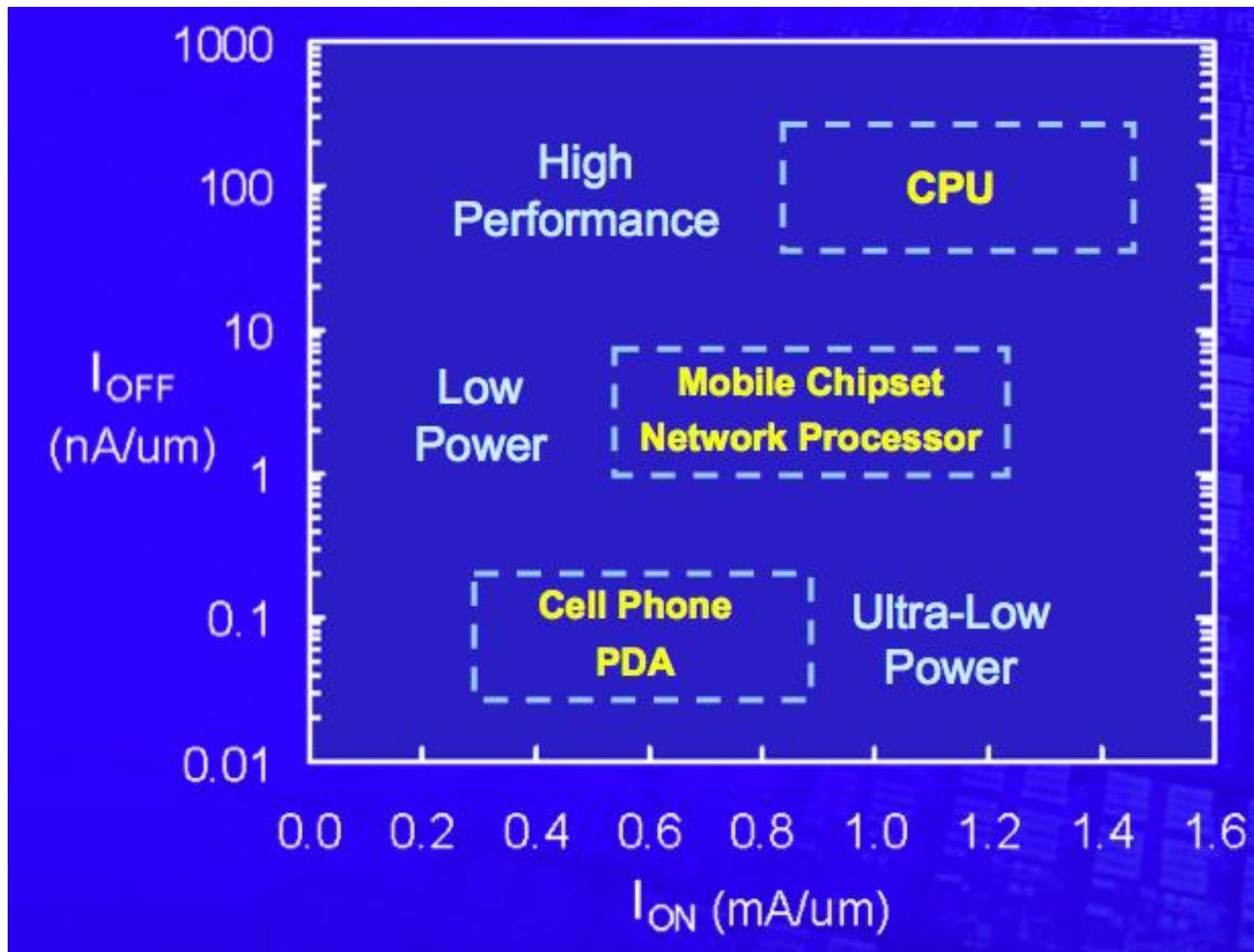
We can reduce  $CV^2$  ( $P_{\text{active}}$ )  
by lowering  $V_{dd}$

We can increase speed  
by raising  $V_{dd}$  and  
lowering  $V_t$

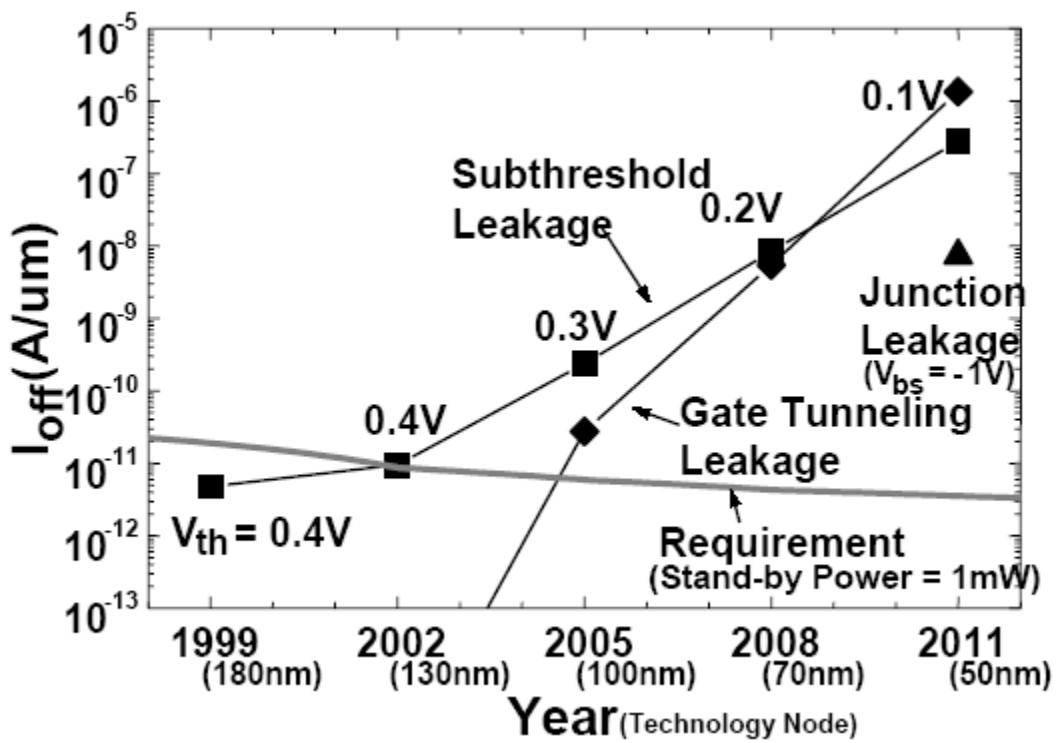
We can reduce leakage  
( $P_{\text{standby}}$ ) by raising  $V_t$



# Customize processes for product types



# Leaky Transistors



# 3 Strategies for Low Power

## 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	

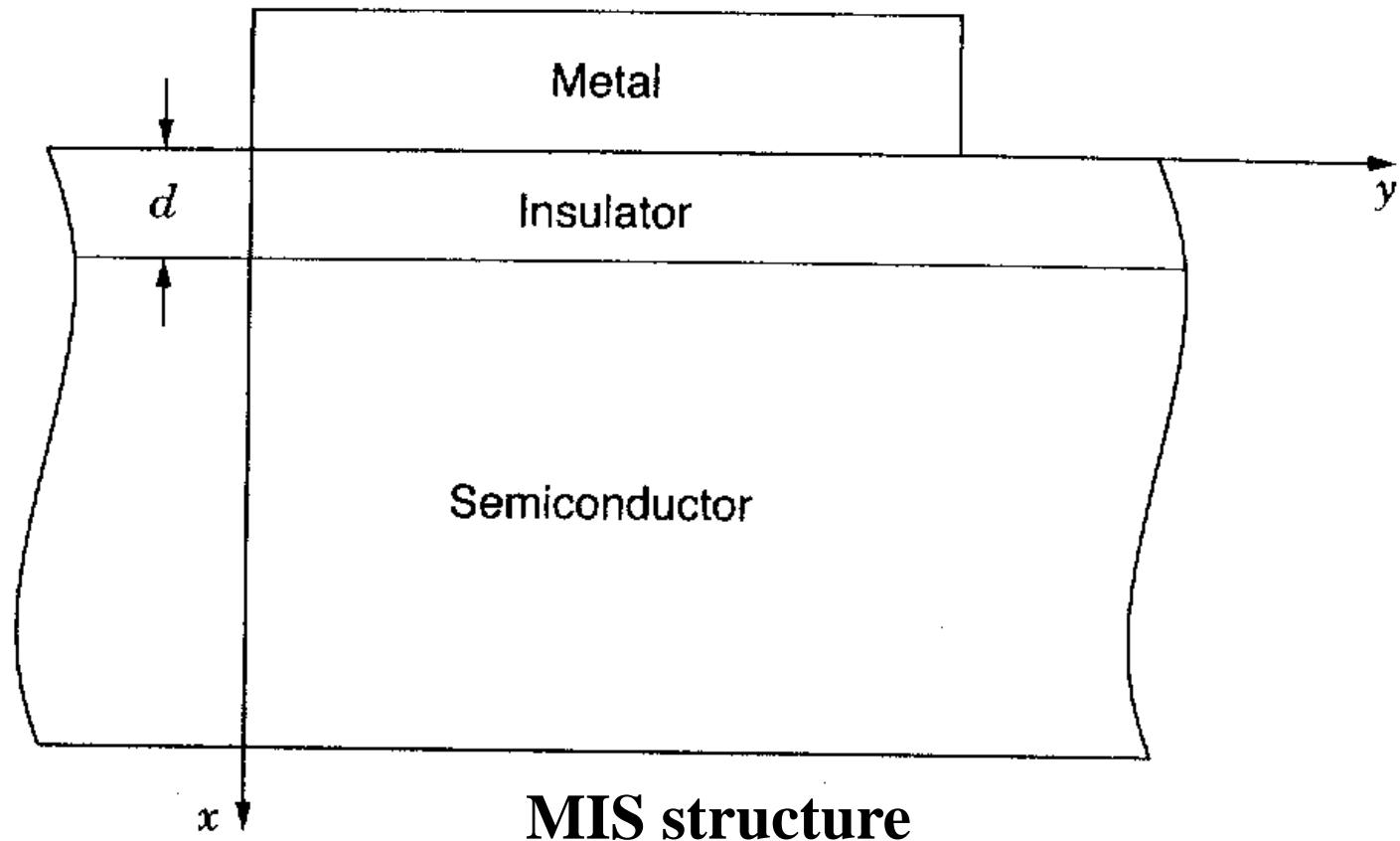
Software-hardware cooperation

Technology-circuit cooperation

# Reliability and Cooling Costs

- High power dissipation → high temperature → malfunction
- High performance microprocessors: ~50 Watts (a hand-held soldering iron)
- Packaging cost and cooling cost: fans
- Power supply rails: high transient current (e.g. 3A)

# Physics of Power Dissipation in CMOS FET Devices



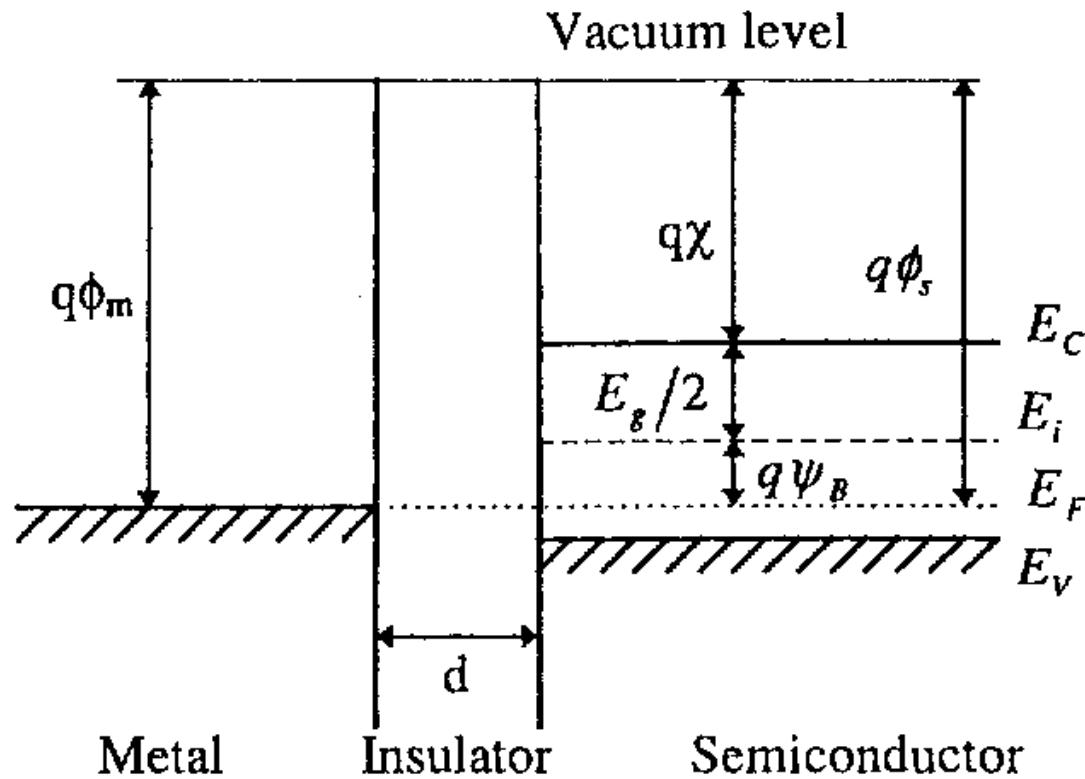
# Physics of Power Dissipation in CMOS FET Devices

- For an ideal MIS diode, the energy difference  $\psi_{ms}$  between the metal work function  $\psi_m$  and the semiconductor work function  $\psi_s$  is zero
- $\psi_{ms} \equiv \psi_m - (\chi + E_g/2q + \Psi_B) = 0$ 
  - where  $\chi$  is the semiconductor electron affinity (from conduction band to vacuum level),  
 $E_g$  the band gap (from valence band to conduction band),  
 $\Psi_B$  the potential barrier between the metal and the insulator, and  
 $\Psi_B$  the potential difference between the Fermi level  $E_F$  and the intrinsic Fermi level  $E_i$

# Fermi-Dirac Function

- $f_{FD}(E) = 1 / (1 + \exp((E - E_F) / kT))$
- The Fermi-Dirac distribution function gives the probability that a certain energy state will be occupied by an electron
- As in a gas, the electrons in a solid are in constant motion and consequently changing their energy and momentum

# P-type



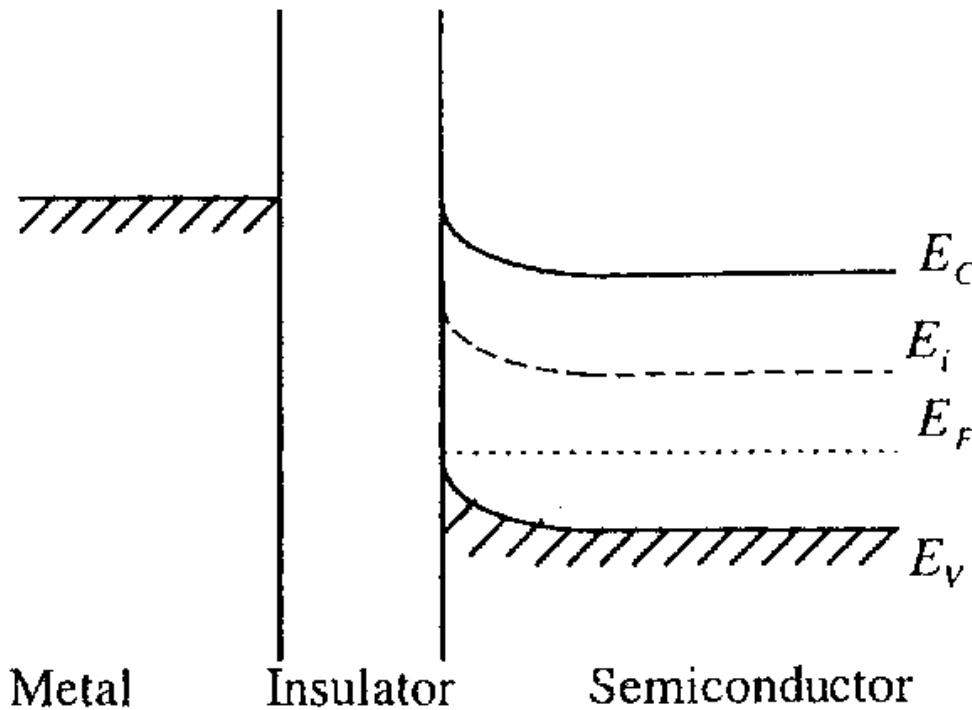
**Energy bands in an unbiased MIS diode**

# CMOS Gate Power equations

- $P = C_L V_{DD}^2 f_{0 \rightarrow 1} + t_{sc} V_{DD} I_{peak} f_{0 \rightarrow 1} + V_{DD} I_{leakage}$
- Dynamic term  $C_L V_{DD}^2 f_{0 \rightarrow 1}$
- Short-circuit term  $t_{sc} V_{DD} I_{peak} f_{0 \rightarrow 1}$
- Leakage term  $V_{DD} I_{leakage}$
- The Maxwell-Boltzmann statistics relates the equilibrium hole concentration to the intrinsic Fermi level:  
$$p_0 = n_i \exp((E_i - E_F)/kT)$$

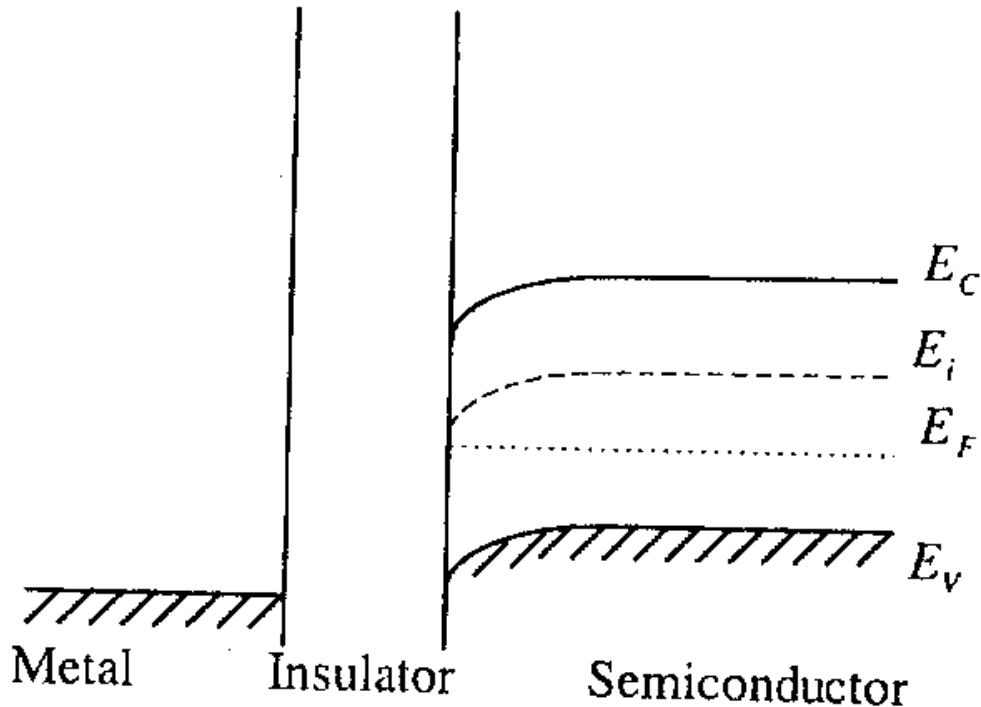
# P substrate

- The Fermi level  $E_F$  in the semiconductor is now  $-qV$  below the Fermi level in the metal gate



**Energy bands when a negative bias is applied**

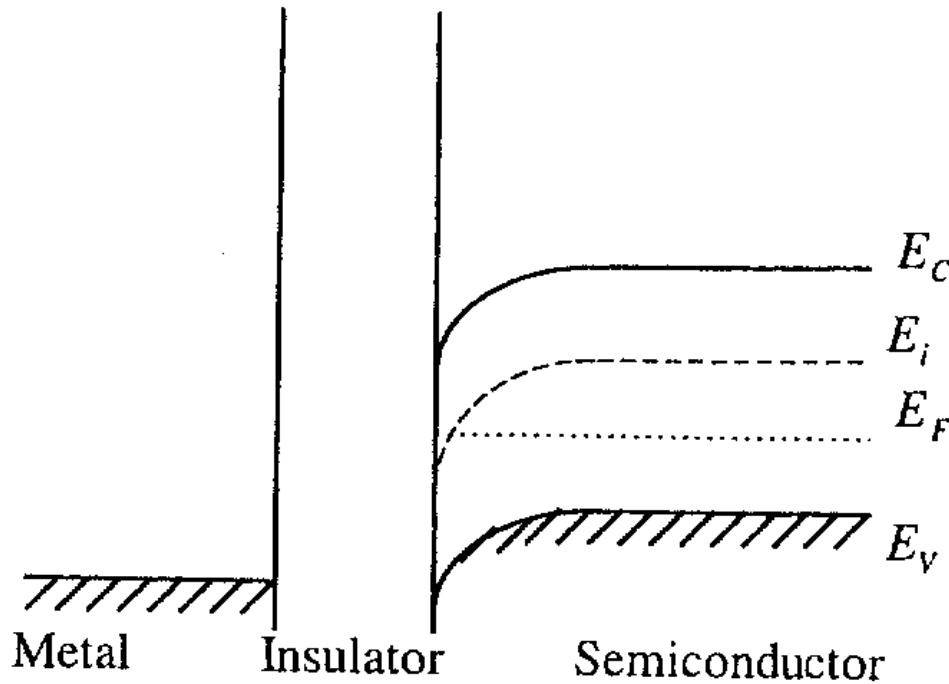
# P Substrate



**Energy bands when a small positive bias is applied**

- If the applied voltage is increased sufficiently, the bands bend far enough that level  $E_i$  at the surface crosses over to the other side of level  $E_F$
- This is brought about by the tendency of carriers to occupy states with the lowest total energy
- In the present condition of inversion the level  $E_i$  bends to be closer to level  $E_c$  and electrons outnumber holes at the surface

- $E_i$  at the surface now is below  $E_F$  by an amount of energy equal to  $2 \Psi_B$ , where  $\Psi_B$  is the potential difference between the Fermi level  $E_F$  and the intrinsic Fermi level  $E_i$  in the bulk.



### Energy bands in strong inversion

- The value of  $V$  necessary to reach the onset of strong inversion is called the threshold voltage

# Surface Space Charge Region and the Threshold Voltage

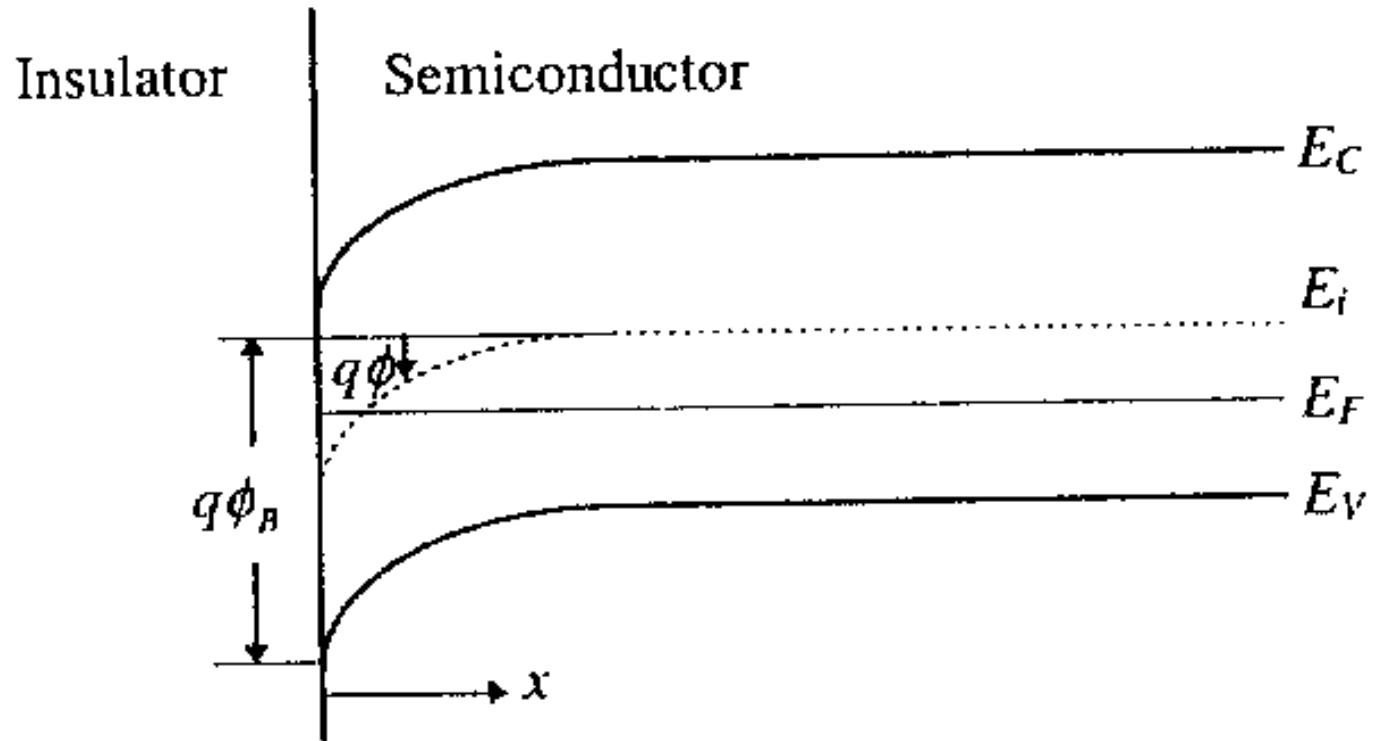
- Poisson equation
- $\nabla \cdot D = \rho(x, y, z)$

Where  $D$  is the electric displacement vector, is equal to  $\epsilon_s E$  under low-frequency or static conditions

$\epsilon_s$  is the permittivity of Si

$E$  the electric field vector and

$\rho(x, y, z)$  the total electric charge density



**Energy bands at the insulator semiconductor surface**

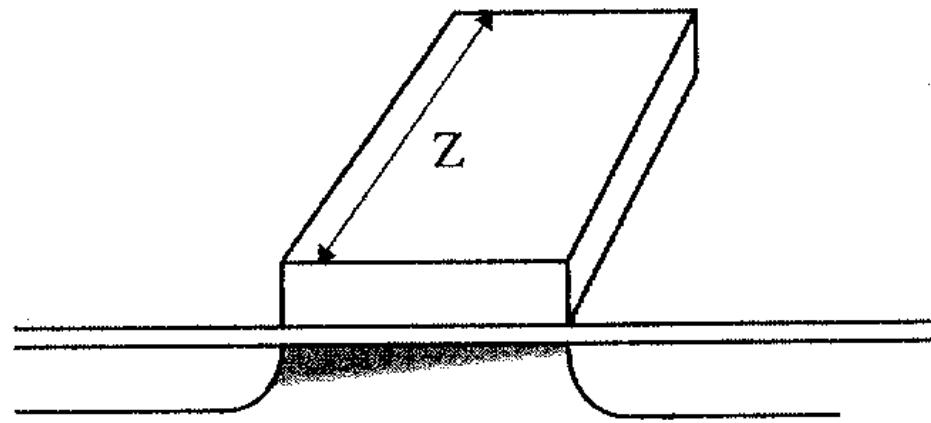
# Threshold voltage

- $V_T = (2d/\epsilon_i) * (q \epsilon_s N_A \psi_B (1 - e^{-2\beta\psi_B}))^{0.5} + 2\psi_B$

The total voltage needed to offset the effect of nonzero work function difference and the presence of the charges is referred to as the flat-band voltage  $V_{FB}$

$$V_{FB} = \psi_{ms} - Q_T * d / \epsilon_i$$

- $V_T = (2d/\epsilon_i) * (q \epsilon_s N_A \psi_B (1 - e^{-2\beta\psi_B}))^{0.5} + 2\psi_B + V_{FB}$
- $V_T$  decreases when  $L$  (length) is decreased, varies with  $Z$  (width), and decreases when the drain-source voltage  $V_{DS}$  is increased.



**Variation of minority concentration in the channel of a MOSFET biased in weak inversion**

# Threshold voltage

- **Drain-induced barrier lowering (DIBL)** is the basis for a number of more complex models of the threshold voltage shift
- It refers to the decrease in threshold voltage due to the depletion region charges in the potential barrier between the source and the channel at the semiconductor surface
- A recent model adopt a quasi two-dimensional approach to solving the two-dimensional Poisson equation
- $dE_x/dx$  at each point  $(x, y)$  can be replaced with the average of its value at  $(0, y)$  and at  $(W, y)$

## Short channel effect

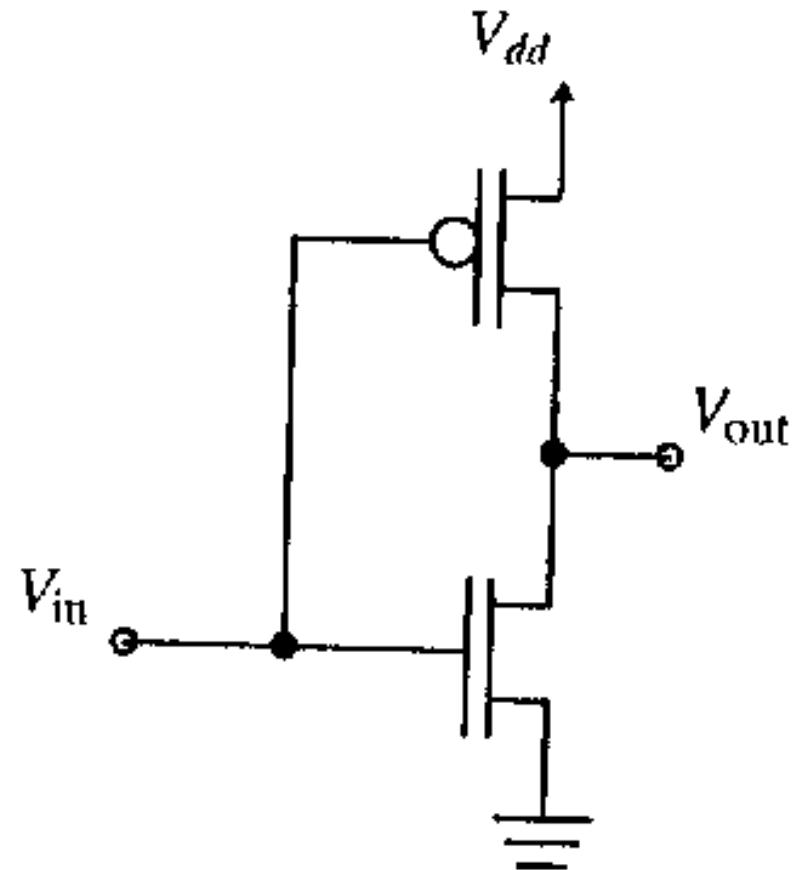
- The minimum value of the surface potential increases with decreasing channel length and increasing  $V_{DS}$

# Subsurface Drain-Induced Barrier Lowering (Punchthrough)

- The punchthrough voltage  $V_{PT}$  defined as the value of  $V_{DS}$  at which  $I_{D, st}$  reaches some specific magnitude with  $V_{GS} = 0$
- The parameter  $V_{PT}$  can be roughly approximated as the value of  $V_{DS}$  for which the sum of the widths of the source and the drain depletion regions becomes equal to  $L$

# Power Dissipation in CMOS

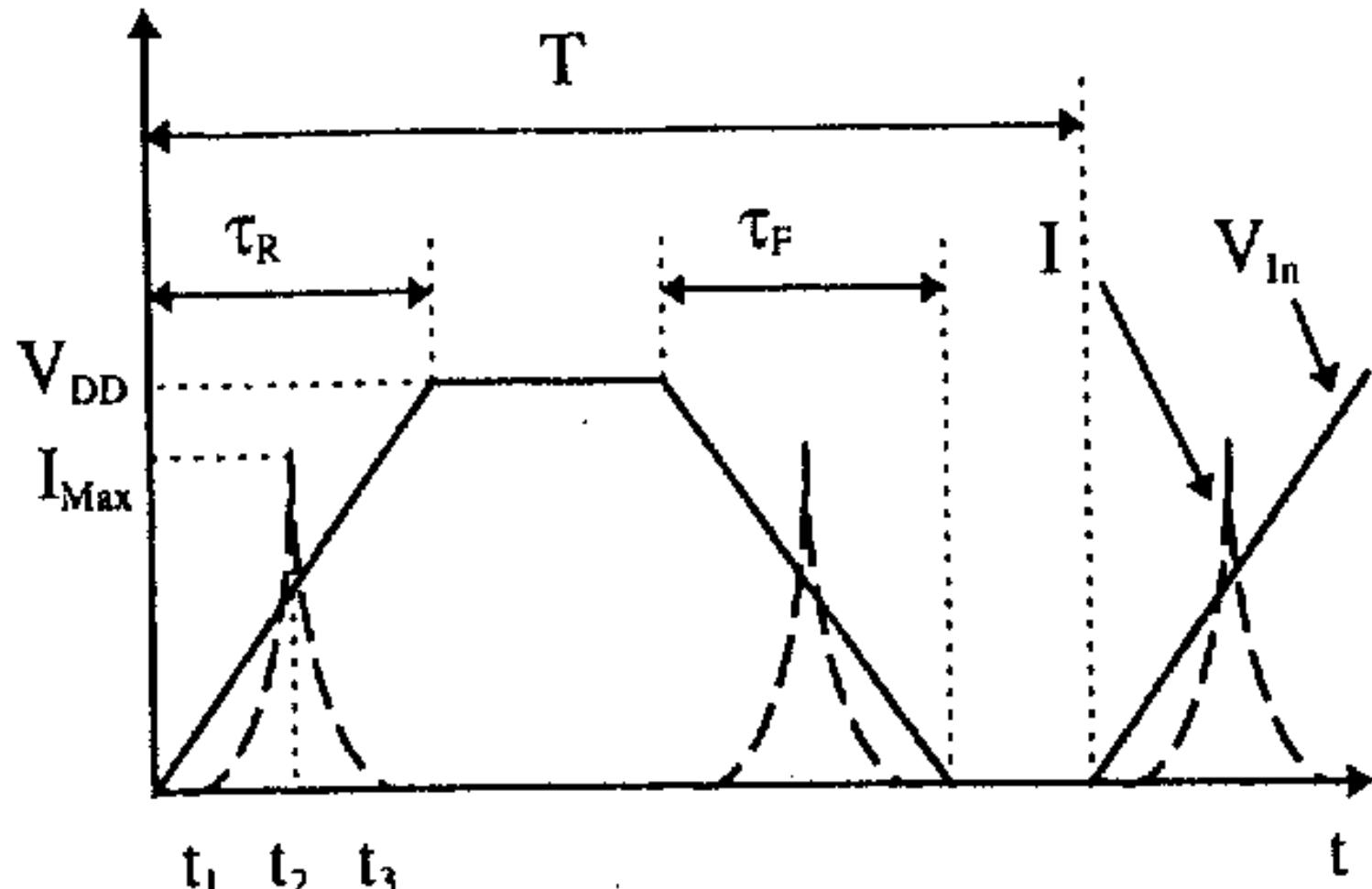
- The first ICs ever fabricated used a PMOS process. This is due to the simplicity of fabrication of a p-channel enhancement mode MOS field-effect transistor (PMOST) with threshold voltage  $V_{Tp} < 0$
- The charge mobility factor caused the move to the NMOS process
- Then change to CMOS because of the power dissipation problem
- This advantage of CMOS over NMOS has proven to be important enough that the shortcomings of CMOS are overlooked
- The CMOS process is more complex than the NMOS, the CMOS requires use of guard-rings to get around the latch-up problem, and CMOS circuits require more transistors than the equivalent NMOS circuits



**CMOS inverter**

# Short-Circuit Dissipation

- The short-circuit dissipation of the gate varies with the output load and the input signal slope
- The short-circuit dissipation decreases linearly (roughly) in both absolute terms and a fraction of the total dissipation as the output load is increased to a critical value and then it will increase again rapidly



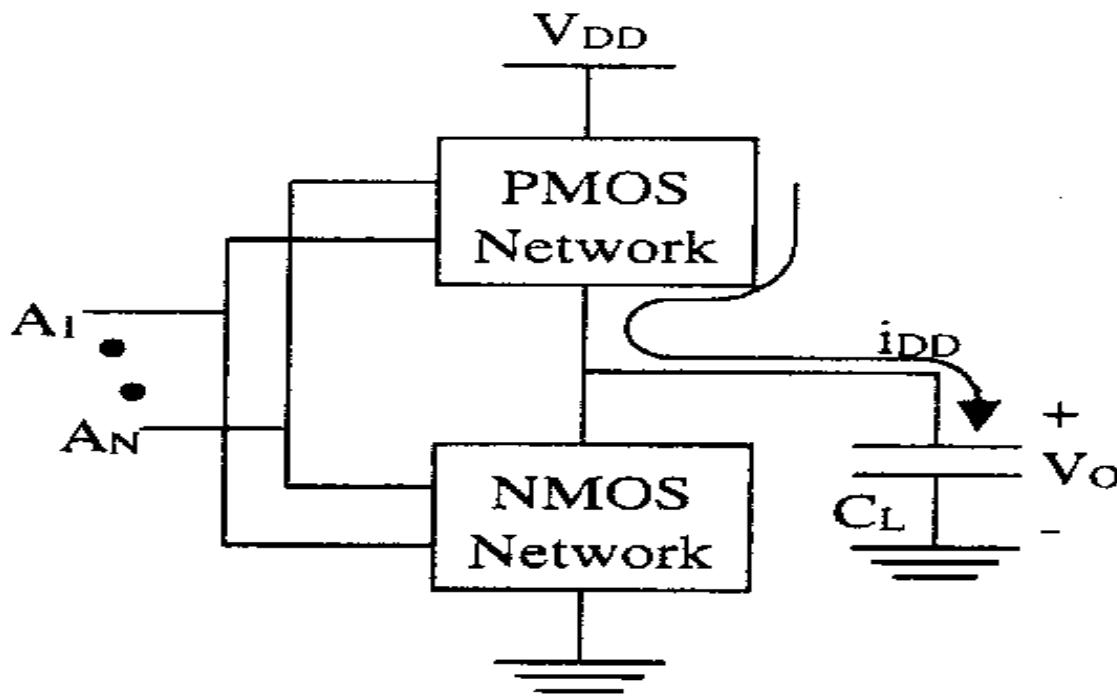
## Short Circuit Current

- For simplicity a symmetrical inverter (i.e.,  $\beta_N = \beta_p$  and  $V_{Tn} = -V_{tp}$ ;) and a symmetrical input signal (rise time = fall time) are considered.
- $I = \beta/2(V_{in} - V_T)^2$  for  $0 \leq I \leq I_{max}$
- $I_{mean} = 1/T \int_0^T I(t) dt$   
 $= 2 * 2/T \int_{t1}^{t2} \beta/2 (V_{in}(t) - V_T)^2 dt$
- Assuming the rising and falling portions of the input voltage waveform to be linear ramps,
- $V_{in}(t) = t * V_{DD}/\tau$
- $I_{mean} = 2 * 2/T \int_{(Vt/V_{DD})}^{\tau} \beta/2 (t * V_T/\tau - V_T)^2 dt$
- Let  $\theta = (V_T/\tau)t - V_T$

- $I_{\text{mean}} = -2\beta/T \int_{(V_t/V_{DD})}^{\tau/2} \theta d\theta$
- $I_{\text{mean}} = 1/12 * \beta/V_{DD} (V_{DD} - V_T)^3 \tau/T$
- The short-circuit power dissipation of an unloaded inverter is
- $P_{SC} = \beta/12(V_{DD} - V_T)^3 \tau/T$
- If the inverter is lightly loaded, causing output rise and fall times that are relatively shorter than the input rise and fall times, the short-circuit dissipation increases to become comparable to dynamic dissipation
- To minimize dissipation, an inverter should be designed in such a way so that the input rise and fall times are about equal to the output rise and fall times

# Dynamic Dissipation

- Assuming that the input  $V_{in}$  is a square wave having a period  $T$  and that the rise and fall times of the input are much less than the repetition period, the dynamic dissipation is given by
- $P_D = C_L V_{DD}^2/T$



## Energy per transition

- When  $V = V_{DD}$ ,  $E_{0 \rightarrow 1} = C_L V_{DD}^2$ .
- When energy stored in a capacitor with capacitance  $C_L$  and voltage  $V_{DD}$  across its plates is  $C_L V_{DD}^2/2$ , the rest of the energy, another  $C_L V_{DD}^2/2$ , is converted into heat

# Principles of Low-Power Design

- Using the lowest possible supply voltage
- Using the smallest geometry, highest frequency devices but operating them at the lowest possible frequency
- Using parallelism and pipelining to lower required frequency of operation
- Power management by disconnecting the power source when the system is idle
- Designing systems to have lowest requirements on subsystem performance for the given user level functionality

# Fundamental Limits

- The limit from **thermodynamic** principles results from the need to have, at any node with an equivalent resistor  $R$  to the ground, the signal power  $P_s$  exceed the available noise power  $P_{\text{avail}}$
- The **quantum** theoretic limit on low power comes from the Heisenberg uncertainty principle. In order to be able to measure the effect of a switching transition of duration  $\Delta t$ , it must involve an energy greater than  $h/\Delta t$ :
- $P \geq h/(\Delta t)^2$  where  $h$  is the Planck's constant

- Finally the fundamental limit based on **electromagnetic** theory results in the velocity of propagation of a high-speed pulse on an interconnect to be always less than the speed of light in free space,  $c_0$ :
- $L/\tau \leq c_0$  where  $L$  is the length of the interconnect and  $\tau$  is the interconnect transit time

# Material Limits

- The attributes of a semiconductor material that determine the properties of a device built with the material are
- **Carrier mobility  $\mu$**
- Carrier saturation velocity  $\sigma_s$
- Self-ionizing electric field strength  $E_c$
- **Thermal conductivity K**

- Consider an SOI structure by surrounding the above generic device in a hemispherical shell of  $\text{SiO}_2$  of radius  $r_i$ , indicating a two-order-of-magnitude reduction in thermal conductivity
- The response time of the global interconnect circuit is
$$\tau = (2.3 R_{tr} + R_{int}) C_{int}$$
where  $R_{tr}$  is the output resistance of the driving transistor and  $R_{int}$  and  $C_{int}$  are the total resistance and capacitance, respectively, of the global interconnect.

# System Limits

- The architecture of the chip
- The power-delay product of the CMOS technology used to implement the chip
- The heat removal capacity of the chip package
- The clock frequency
- Its physical size

# Energy characterization

- Transition-sensitive energy models
  - Single energy tables
    - Bit independent modules e.g., flipflops
  - Multiple energy tables
    - Large bit dependent modules e.g., 32-b adders
    - Large multi-element modules e.g., register files
  - Transition sensitive energy equations
  - System level interconnect capacitance values
- Analytical energy modes
  - Cache and main memory

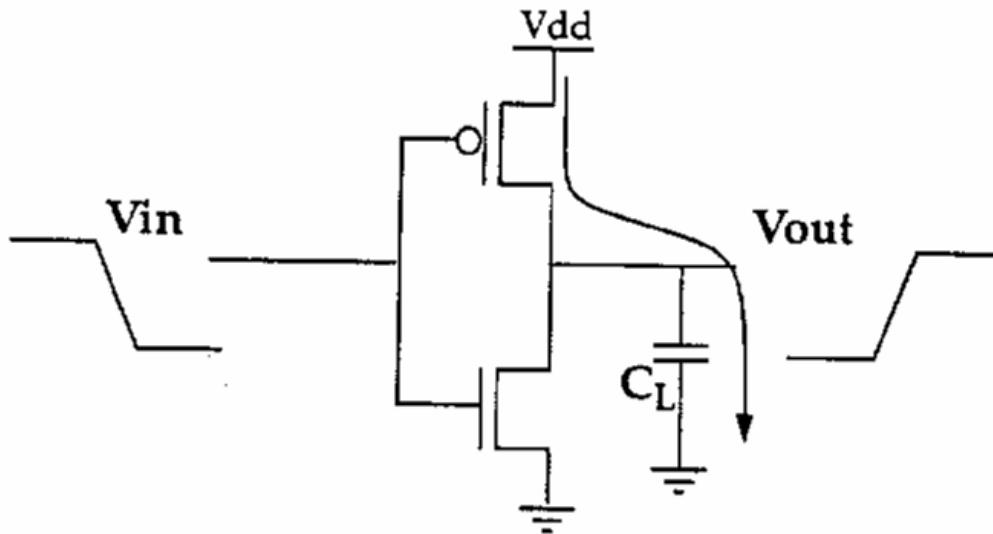
# Transition-sensitive energy model

- Must first design and layout a functional unit and then simulate it to capture switch capacitances
  - Bit independent – bus lines, pipeline registers
    - One bit switching does not affect other bit slices' operations
    - Bit dependent – ALU, decoders
- Once constructed, the models can be reused in simulations of other architectures built with the same technology

# Architectural Level Analysis Considerations

- Very computationally efficient
  - Requires predefined analytical and transition-sensitive energy characterization models
  - Requires design only to RTL (with some idea as to the kind of functional units planned)
  - Coarse grain – use of gated clocks implicit
- Reasonably accurate (within 5% - 15% of SPICE)

# Dynamic Power Consumption



$$\text{Energy/transition} = C_L * V_{dd2}$$

$$\text{Power} = \text{Energy/transition} * f = C_L * V_{dd}^2 * f$$

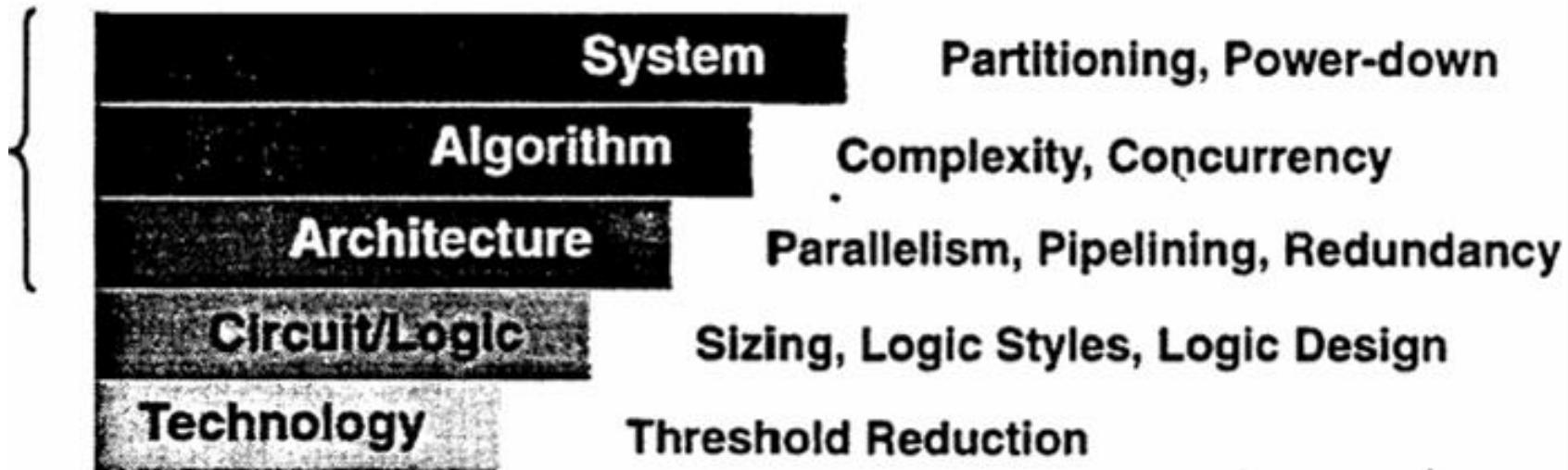
- Not a function of transistor sizes!
- Need to reduce  $C_L$ ,  $V_{dd}$ , and  $f$  to reduce power
- Reduce the probability,  $P_{0 \rightarrow 1}$

# Dynamic Power Consumption - Extended

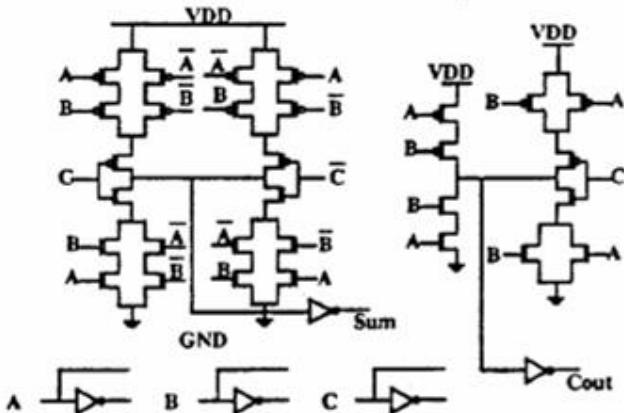
- Power = Energy/transition \* transition rate
  - =  $C_L * V_{dd}^2 * f_{0 \rightarrow I}$
  - =  $C_L * V_{dd}^2 * P_{0 \rightarrow I} * f$
  - =  $C_{EFF} * V_{dd}^2 * f$
- Power Dissipation is Data Dependent Function of Switching Activity
  - $C_{EFF} = \text{Effective Capacitance} = C_L * P_{0 \rightarrow I}$

# Ultra Low Power System Design

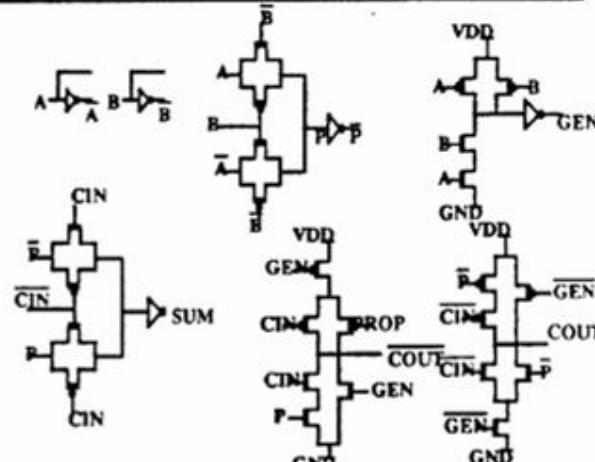
- Power minimization approaches:
  - Run at minimum allowable voltage
  - Minimize effective switching capacitance



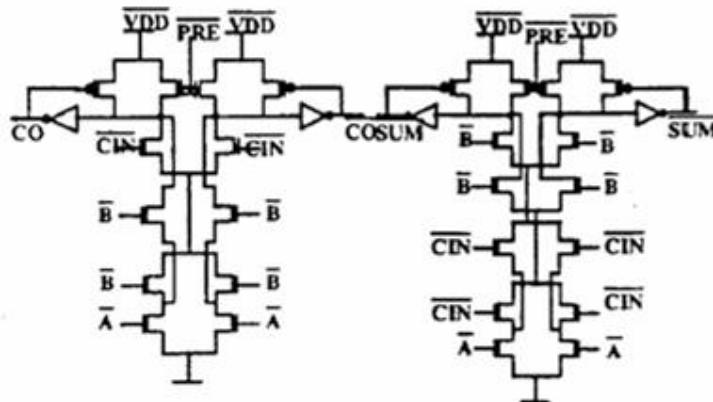
# Choice of Logic Style



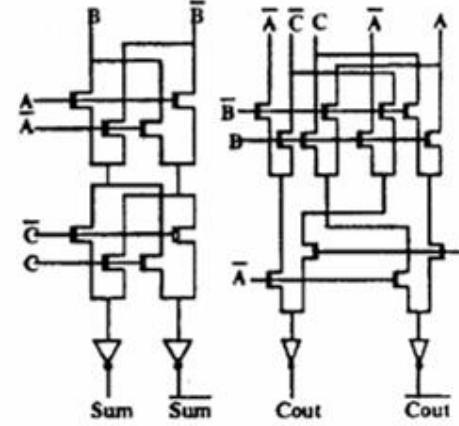
CONVENTIONAL CMOS Adder



OPTIMIZED static Adder

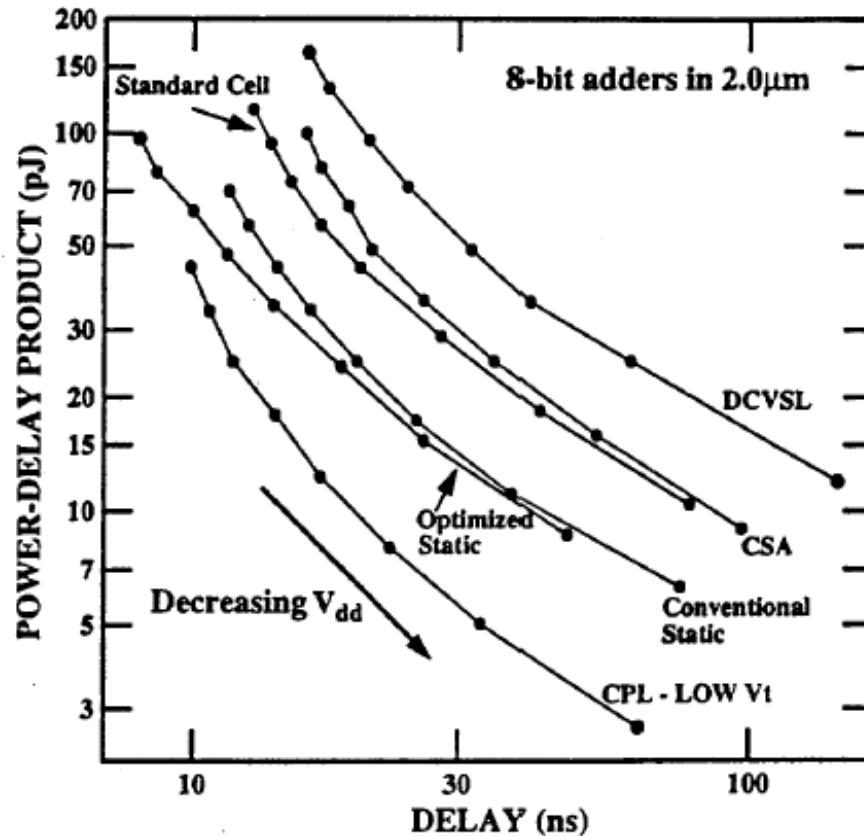


DCVSL Adder



CPI Adder

# Choice of Logic Style



- Power-delay product improves as voltage decreases
- The “best” logic style minimizes power-delay for a given delay constraint

# Power Consumption is Data Dependent

- Example : Static 2 Input NOR Gate

A	B	Out
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table of 2 input NOR gate

Assume :

$$P(A=1) = \frac{1}{2}$$

$$P(B=1) = \frac{1}{2}$$

Then :

$$P(\text{Out}=1) = \frac{1}{4}$$

$$P(0 \rightarrow 1)$$

$$= P(\text{Out}=0).P(\text{Out}=1)$$

$$= \frac{3}{4} * \frac{1}{4} = \frac{3}{16}$$

$$C_{\text{EFF}} = \frac{3}{16} * C_L$$

# Operating at the Lowest Possible Voltage!

- Desire to operate at lowest possible speeds (using low supply voltages)
- Use Architecture optimization to compensate for slower operation

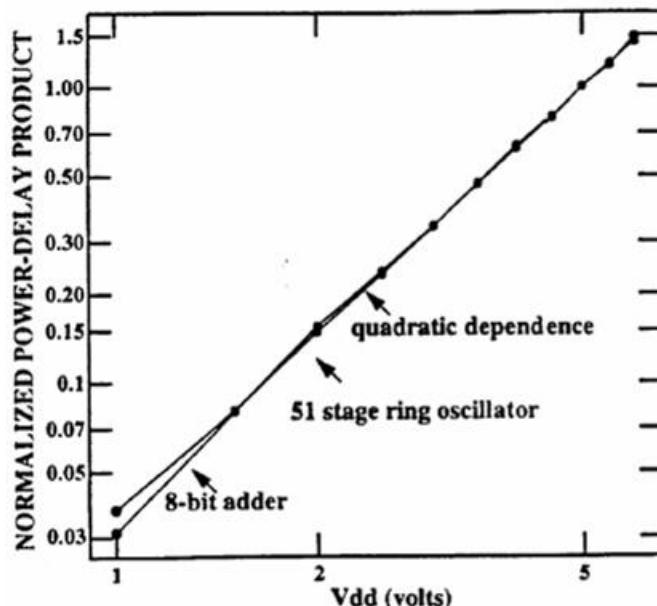
Approach : Trade-off **AREA** for lower **POWER**

# Operating at the Lowest Possible Voltage!

- Desire to operate at lowest possible speeds (using low supply voltages)
- Use Architecture optimization to compensate for slower operation

Approach : Trade-off **AREA** for lower **POWER**

# Reducing V<sub>dd</sub>



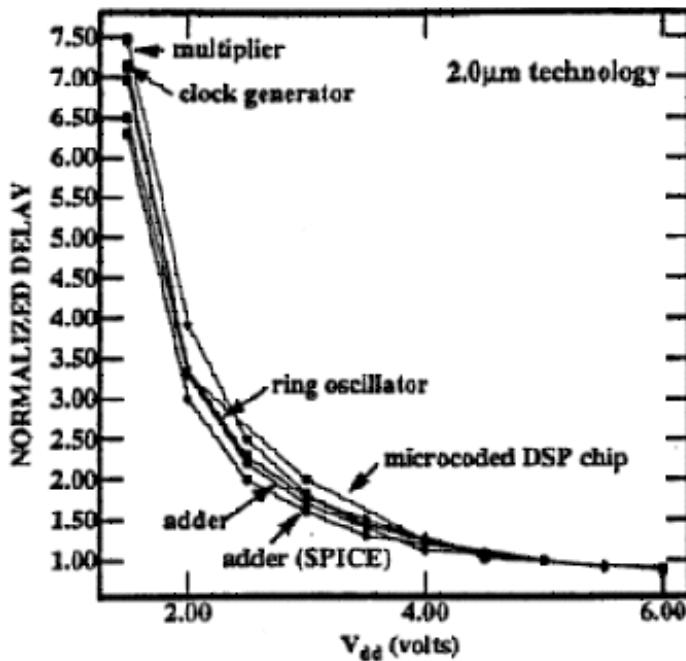
$$P \times t_d = E_t = C_L \cdot V_{dd}^2$$

$$\frac{E_{(V_{dd}=2)}}{E_{(V_{dd}=5)}} = \frac{(C_L) \cdot (2)^2}{(C_L) \cdot (5)^2}$$

$$E_{(V_{dd}=2)} \approx 0.16 E_{(V_{dd}=5)}$$

- Strong function of voltage ( $V^2$  dependence).
- Relatively independent of logic function and style.
- Power Delay Product Improves with lowering V<sub>DD</sub>.

# Lowering V<sub>dd</sub> Increases Delay



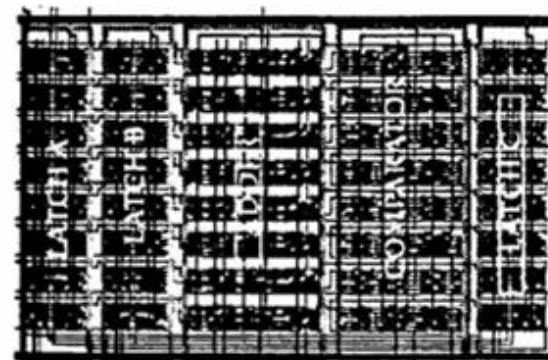
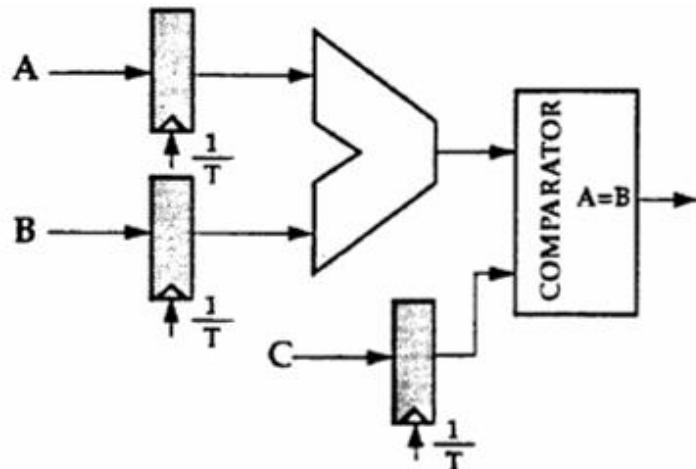
$$T_d = \frac{C_L * V_{dd}}{I}$$

$$I \sim (V_{dd} - V_t)^2$$

$$\frac{T_d(V_{dd=2})}{T_d(V_{dd=5})} = \frac{(2) * (5 - 0.7)^2}{(5) * (2 - 0.7)^2} \\ \approx 4$$

- Relatively independent of logic function and style.
- Concept of Dynamic Voltage Scaling (DVS)

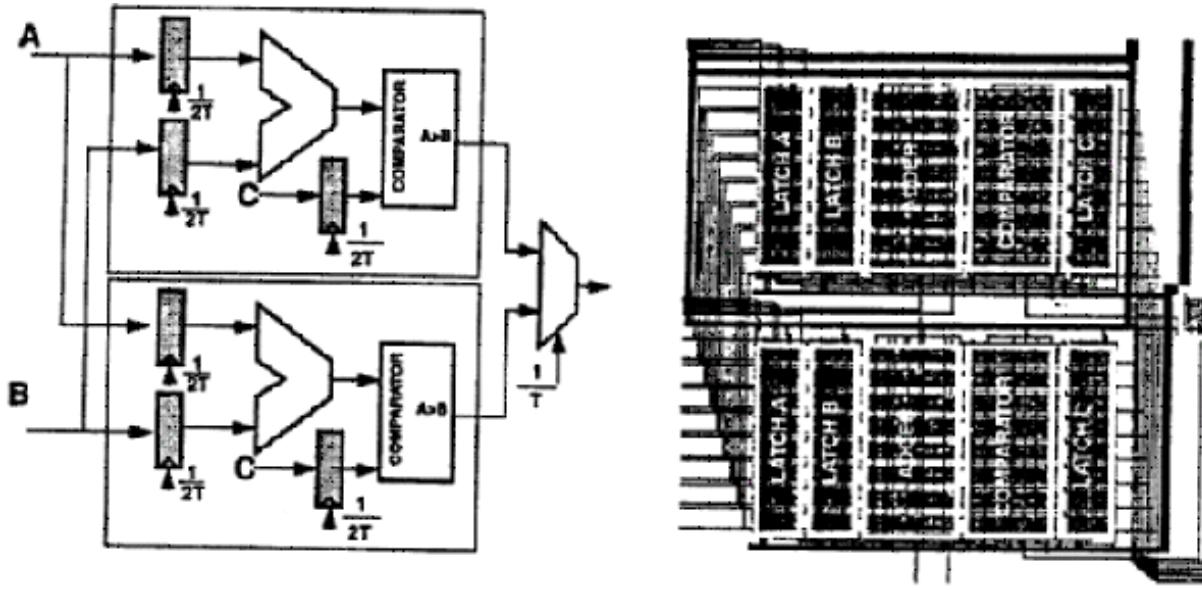
## Architecture Trade-offs : Reference Data Path



$$\text{Area} = 636 \times 833 \text{ u}^2$$

- Critical path delay  $\Rightarrow T_{\text{adder}} + T_{\text{comparator}} (= 25\text{ns})$   
 $\Rightarrow f_{\text{ref}} = 40\text{Mhz}$
  - Total capacitance being switched =  $C_{\text{ref}}$
  - $V_{dd} = V_{\text{ref}} = 5V$
  - Power for reference datapath =  $P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}}$

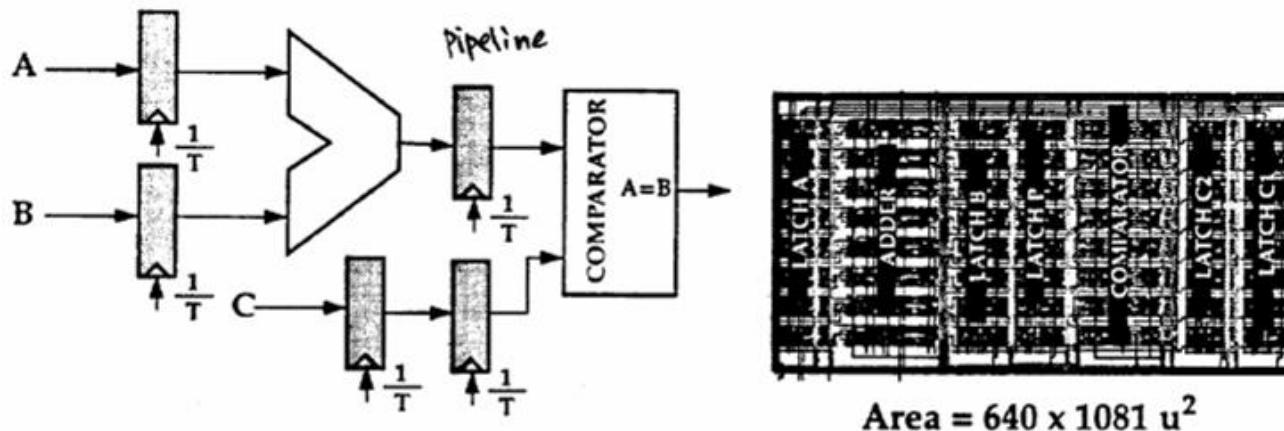
# Parallel Data Path



$$\text{Area} = 1476 \times 1219 \mu^2$$

- The clock rate can be reduced by half with the same throughput  $\Rightarrow f_{\text{par}} = f_{\text{ref}} / 2$
- $V_{\text{par}} = V_{\text{ref}} / 1.7$ ,  $C_{\text{par}} = 2.15C_{\text{ref}}$
- $P_{\text{par}} = (2.15C_{\text{ref}}) (V_{\text{ref}}/1.7)^2 (f_{\text{ref}}/2) \approx 0.36 P_{\text{ref}}$

# Pipelined Data Path



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

# A Simple Data Path : Summary

Architecture type	Voltage	Area	Power
Simple datapath (no pipelining or parallelism)	5V	1	1
Pipelined datapath	2.9V	1.3	0.37
Parallel datapath	2.9V	3.4	0.34
Pipeline-Parallel	2.0V	3.7	0.18

# Computational Complexity of DCT Algorithms

DCT Algorithm	Multiplies (8x8)	Additions (8x8)	Implemented by
Brute Force	4096	4096	-
Row-Col DCT	1024	1024	Bell core (16x16)
Chen's Algorithm	256	416	Telettra
Lee's Algorithm	192	464	SGS - Thompson
Feig's Algorithm (scaled DCT)	54	462	IBM (GP computer)

- Reducing # of operations (switching events) is important in reducing the power.
- Routing and layout issues for irregular structures *vs.* regular structures.

# Power Down Techniques

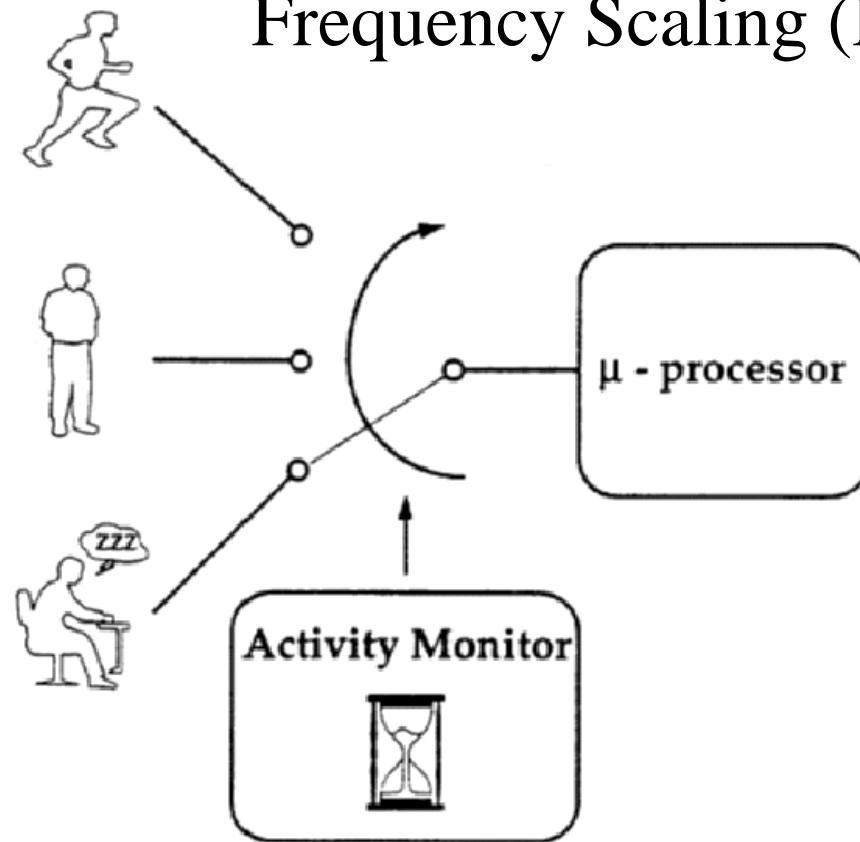
## Operating States

ACTIVE OR FULL-ON  
(FASTEST CLOCK)

STANDBY  
(SLOW CLOCK)

SUSPEND OR SLEEP  
(SLOWEST CLOCK or  
SHUT DOWN)

- Concept of Dynamic Frequency Scaling (DFS)



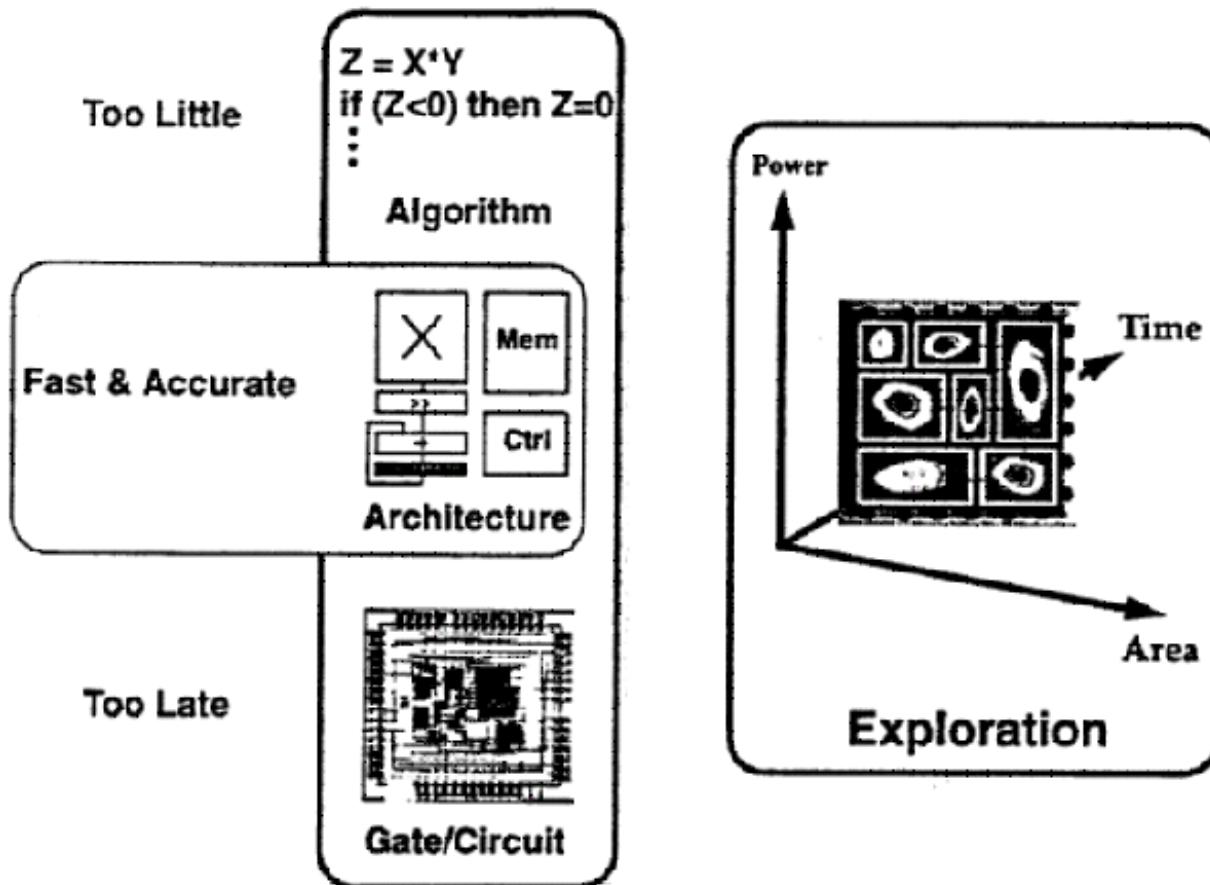
# Energy-efficient Software Coding

- Potential for power reduction via software modification is relatively unexploited.
- Code size and algorithmic efficiency can significantly affect energy dissipation
- Pipelining at software level- VLIW coding style
- Examples -
  - V. Tiwari et al, "Power analysis of embedded software: a first step towards software power miniization," IEEE Trans. on VLSI, vol.2, no. 4, Dec. 1994
  - J. Synder, et al., "Low-power software for low-power people," 1994 IEEE Symp. on Low Power Electronics

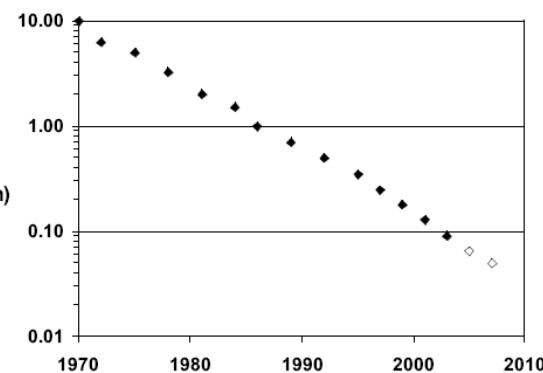
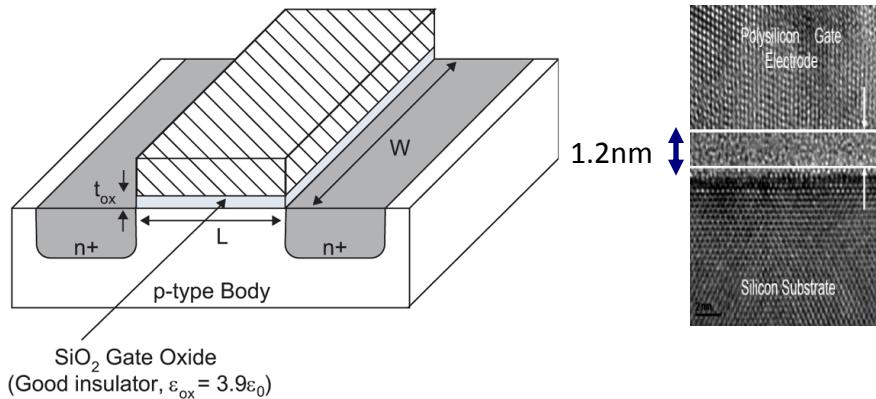
# Power Hunger – Clock Network (Always Ticking)

- H-Tree – design deficiencies based on Elmore delay model
- PLL – every designer (digital or analog) should have the knowledge of PLL
  - Multiple frequencies in chips/systems – by PLL
  - Low main frequency, But
  - Jitter and Noise, Gain and Bandwidth, Pull-in and Lock Time, Stability ...
- Local time zone
- Self-Timed
- Asynchronous => Use Gated Clocks, Sleep Mode

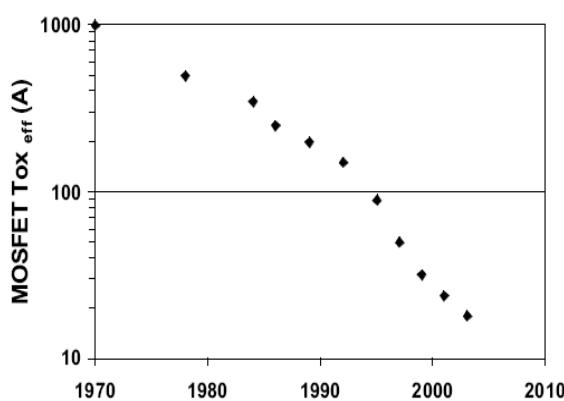
# Power Analysis in the Design Flow



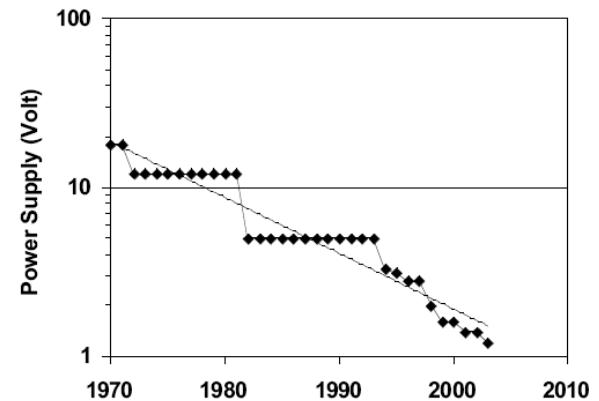
# Scaling of MOS transistors



minimum feature size  
(gate length)

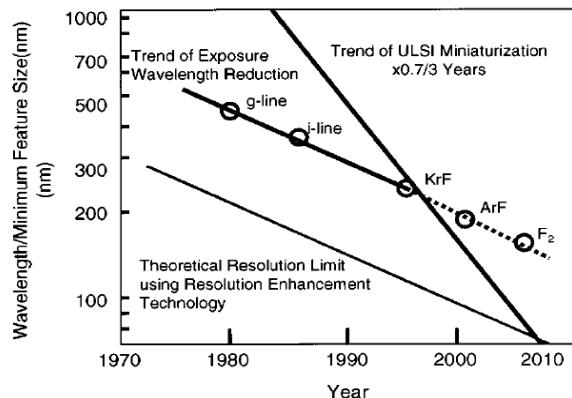


Current oxide thickness ~  
1.0 – 2.0nm thickness → 3 –  
4 atomic layers of oxide

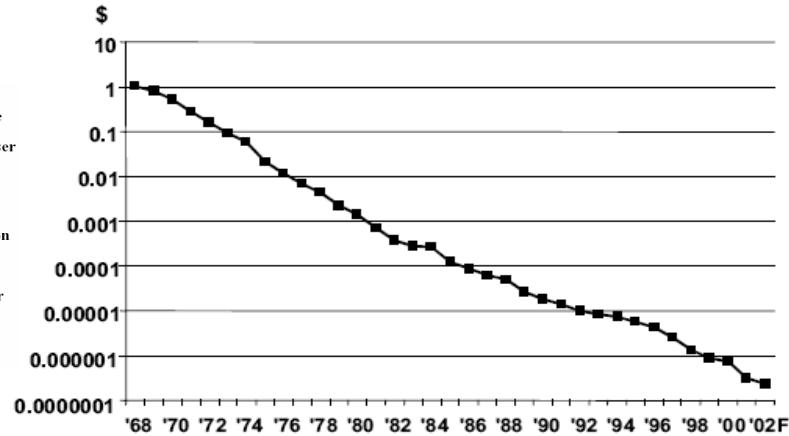
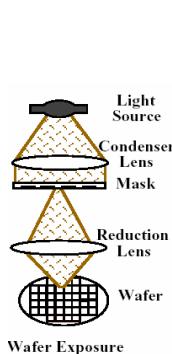


Power supply  
voltage scaling

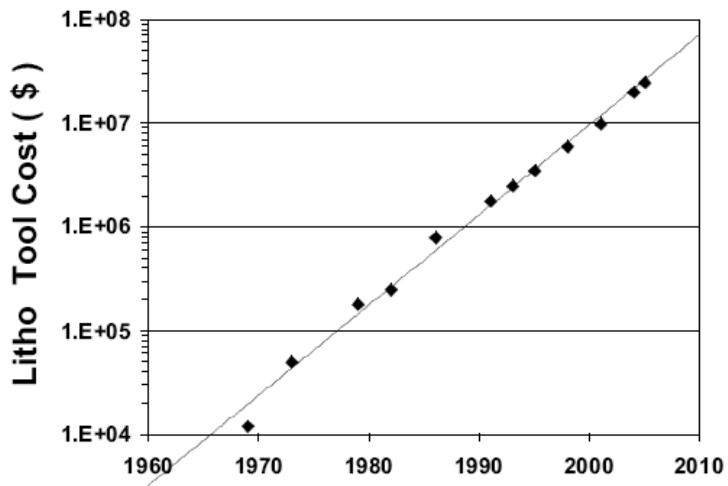
# Scaling



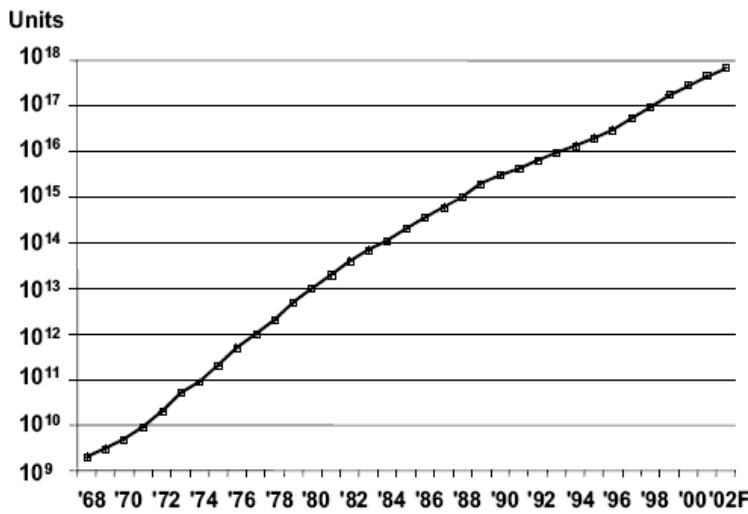
Scaling of lithographic wavelength



Avg transistor price is 0.1 µcent!



Fewer and fewer companies can afford to have their own foundries



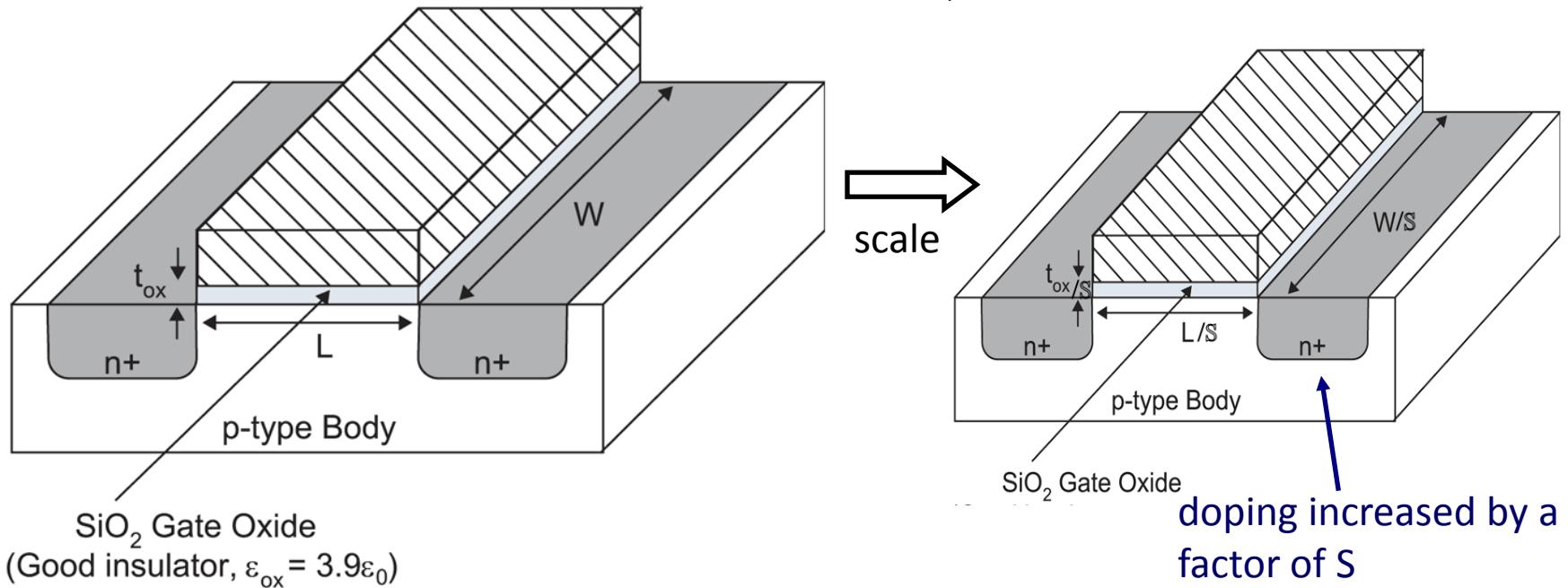
Number of transistors shipped

# Device scaling

(very idealistic NMOS transistor)

$$S = \sqrt{2}$$

(scaled down by  $L$ )



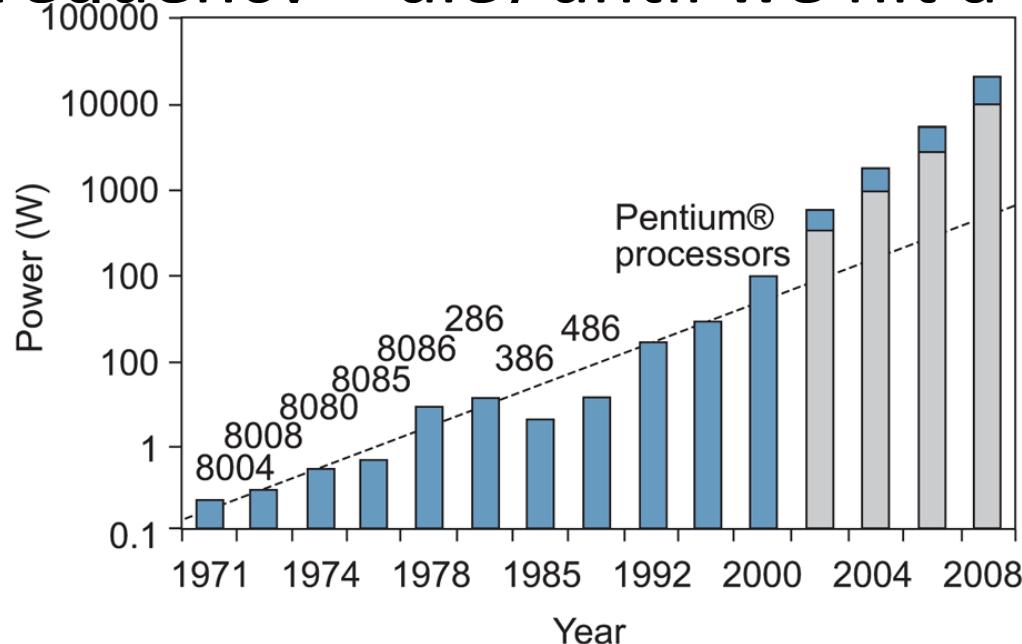
Increasing the channel doping density decreases the depletion width  
⇒ improves isolation between source and drain during OFF status  
⇒ permits distance between the source and drain regions to be scaled

# Implications of ideal device scaling

Parameter	Sensitivity	Constant Field	Lateral
<b>Scaling Parameters</b>			
Length: $L$		$1/S$	$1/S$
Width: $W$		$1/S$	1
Gate oxide thickness: $t_{ox}$		$1/S$	1
Supply voltage: $V_{DD}$		$1/S$	1
Threshold voltage: $V_{tn}, V_{tp}$		$1/S$	1
Substrate doping: $N_A$		$S$	1
<b>Device Characteristics</b>			
$\beta$	$\frac{W}{L} \frac{1}{t_{ox}}$	$S$	$S$
Current: $I_{ds}$	$\beta(V_{DD} - V_t)^2$	$1/S$	$S$
Resistance: $R$	$\frac{V_{DD}}{I_{ds}}$	1	$1/S$
Gate capacitance: $C$	$\frac{WL}{t_{ox}}$	$1/S$	$1/S$
Gate delay: $\tau$	$RC$	$1/S$	$1/S^2$
Clock frequency: $f$	$1/\tau$	$S$	$S^2$
Dynamic power dissipation (per gate): $P$	$CV^2f$	$1/S^2$	$S$
Chip area: $A$		$1/S^2$	1
Power density	$P/A$	1	$S$
Current density	$I_{ds}/A$	$S$	$S$

Influence of scaling on MOS device characteristics

Total power was increasing (mostly because of  $2 \times$  frequency + die) until we hit a “wall”



**FIG 4.69** Intel processor power consumption. © IEEE 2001.

Intel VP Patrick Gelsinger (ISSCC 2001)

“If scaling continues at present pace, by 2005, high speed processors would have power density of nuclear reactor, by 2010, a rocket nozzle, and by 2015, surface of sun.”

# Scaling of standby (leakage) power

Standby power

$$P_{off} \propto \frac{1}{t_{ox}} e^{-\frac{q}{mkT}}$$

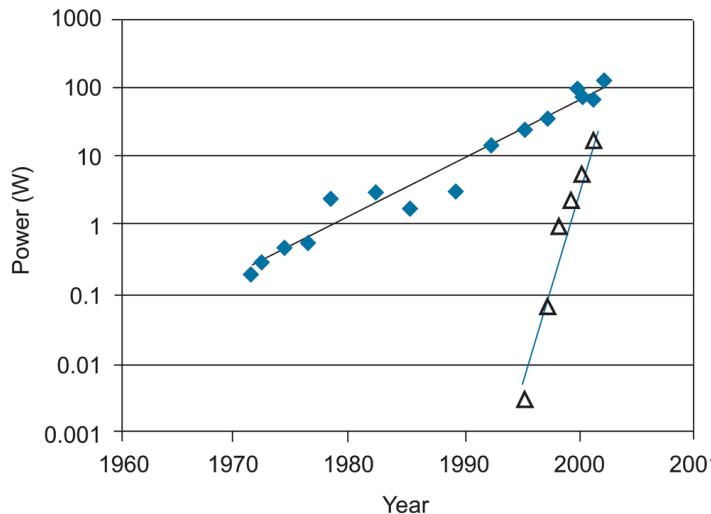
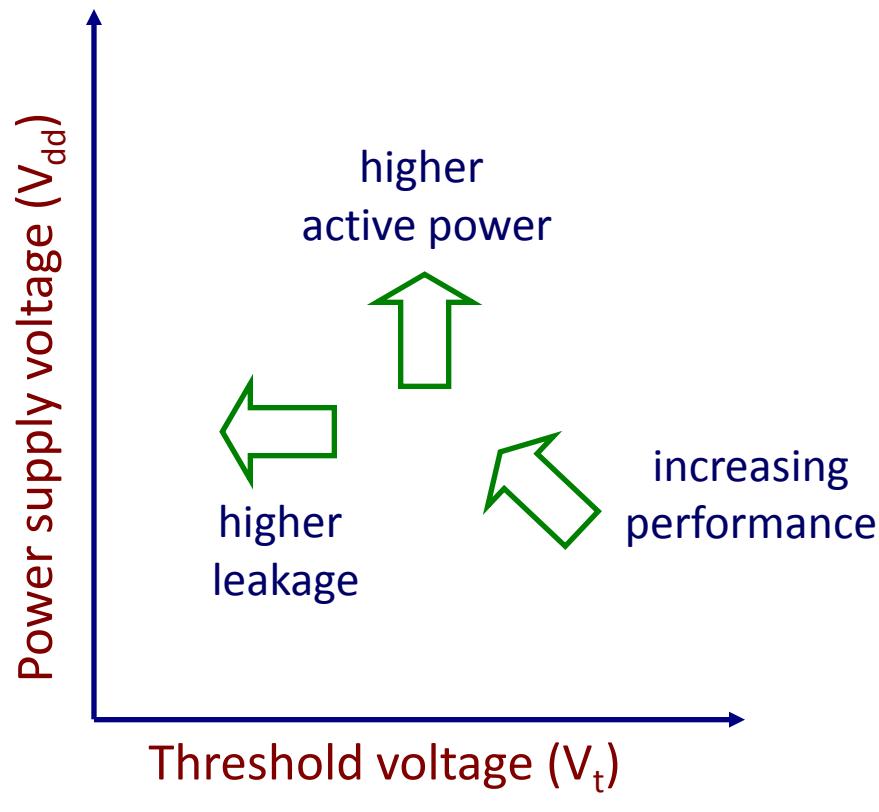


FIG 4.70 Dynamic and static power trends. © IEEE 2003.

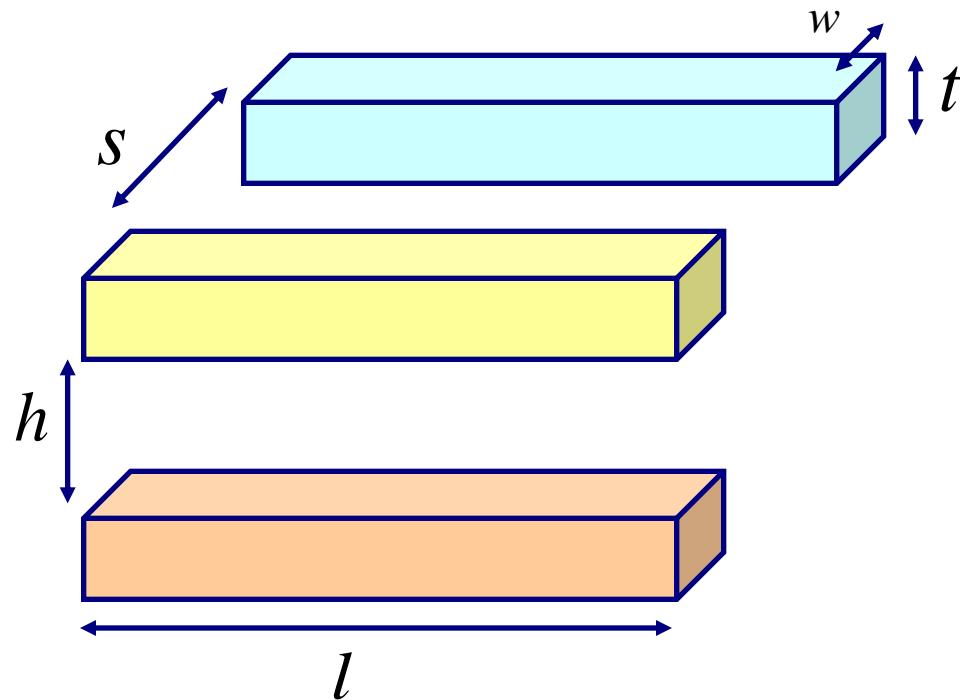
- Even if  $V_t$  is kept constant after scaling,  $P_{off}$  scales up by  $S$  if  $t_{ox}$  is scaled down by  $S$
- $V_t$  must be scaled down if  $V_{DD}$  is scaled down (otherwise  $I_{SAT}$  is weaker and transistor is slow)
- Standby power would further increase by 10× for every 0.1V reduction of  $V_t$

# Power/performance tradeoffs



[Taur, 01]

# Interconnect scaling



$w$ : width of interconnect (layer dependant)

$s$ : spacing between interconnects with same layer

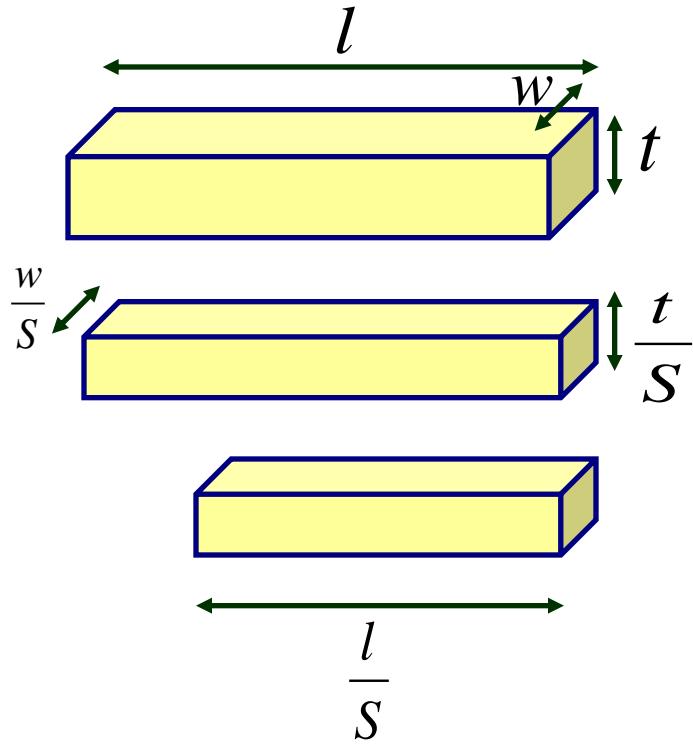
$h$ : dielectric thickness (spacing between interconnects in two vertically adjacent layers)

$l$ : length of interconnect

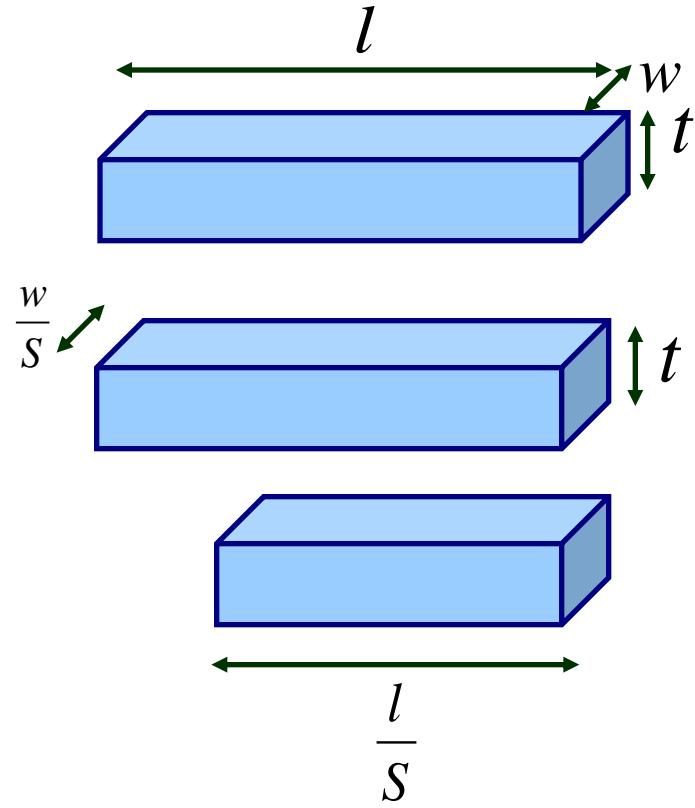
$t$ : thickness of interconnect

# Constant thickness scaling versus reduced thickness scaling

reduced thickness scaling



constant thickness scaling

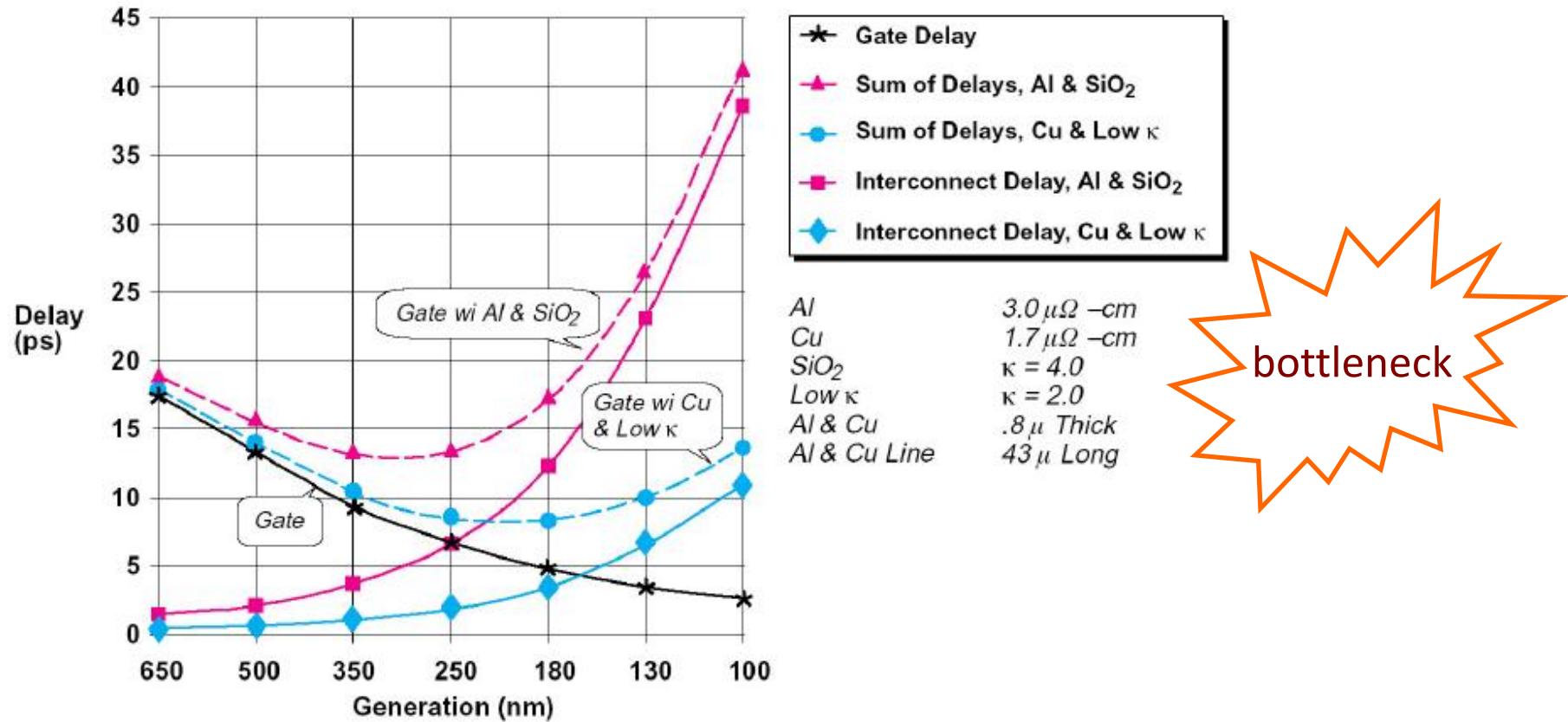


# Implications of ideal interconnect scaling

Parameter	Sensitivity	Reduced Thickness	Constant Thickness
<b>Scaling Parameters</b>			
Width: $w$		$1/S$	
Spacing: $s$		$1/S$	
Thickness: $t$		$1/S$	1
Interlayer oxide height: $h$			$1/S$
<b>Characteristics Per Unit Length</b>			
Wire resistance per unit length: $R_w$	$\frac{1}{wt}$	$S^2$	$S$
Fringing capacitance per unit length: $C_{wf}$	$\frac{t}{s}$	1	$S$
Parallel plate capacitance per unit length: $C_{wp}$	$\frac{w}{h}$	1	1
Total wire capacitance per unit length: $C_w$	$C_{wf} + C_{wp}$	1	between 1, $S$
Unrepeated RC constant per unit length: $t_{wu}$	$R_w C_w$	$S^2$	between $S$ , $S^2$
Repeated wire RC delay per unit length: $t_{wr}$ (assuming constant field scaling of gates in Table 4.15)	$\sqrt{RCR_w C_w}$	$\sqrt{S}$	between 1, $\sqrt{S}$
Crosstalk noise	$\frac{t}{s}$	1	$S$
<b>Local/Scaled Interconnect Characteristics</b>			
Length: $l$			$1/S$
Unrepeated wire RC delay	$\rho t_{wu}$	1	between $1/S$ , 1
Repeated wire delay	$lt_{wr}$	$\sqrt{1/S}$	between $1/S$ , $\sqrt{1/S}$
<b>Global Interconnect Characteristics</b>			
Length: $l$			$D_c$
Unrepeated wire RC delay	$\rho t_{wu}$	$S^2 D_c^2$	between $SD_c^2$ , $S^2 D_c^2$
Repeated wire delay	$lt_{wr}$	$D_c \sqrt{S}$	between $D_c$ , $D_c \sqrt{S}$

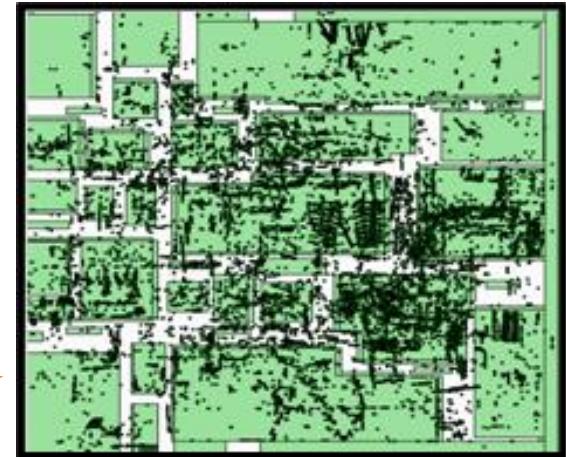
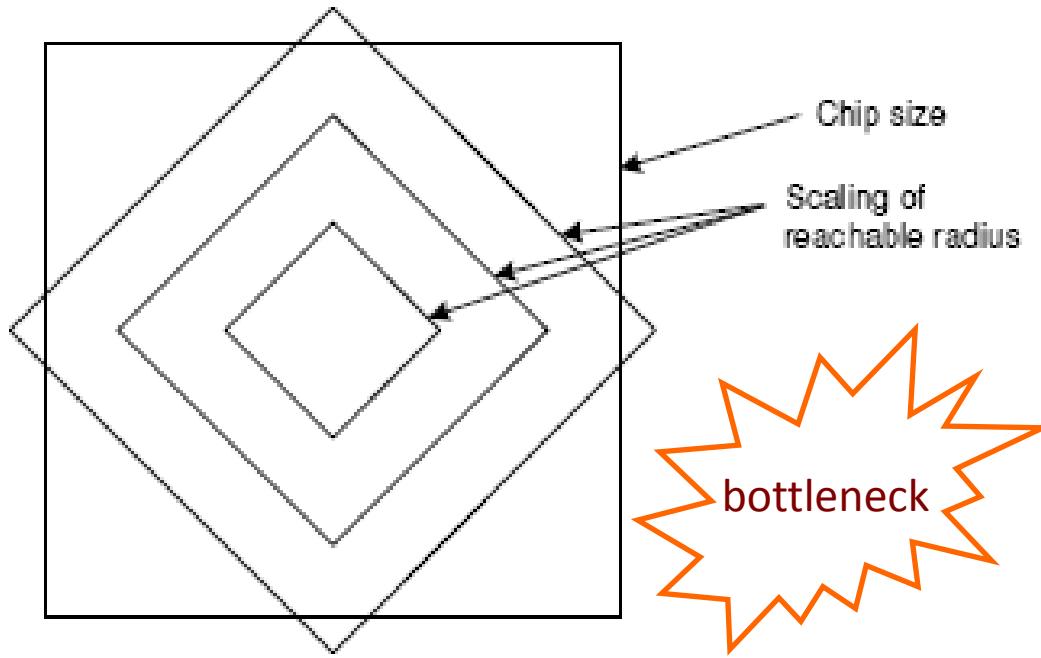
Influence of scaling on interconnect characteristics

# Interconnect delay is dominating gate delay



Repeaters can help but...

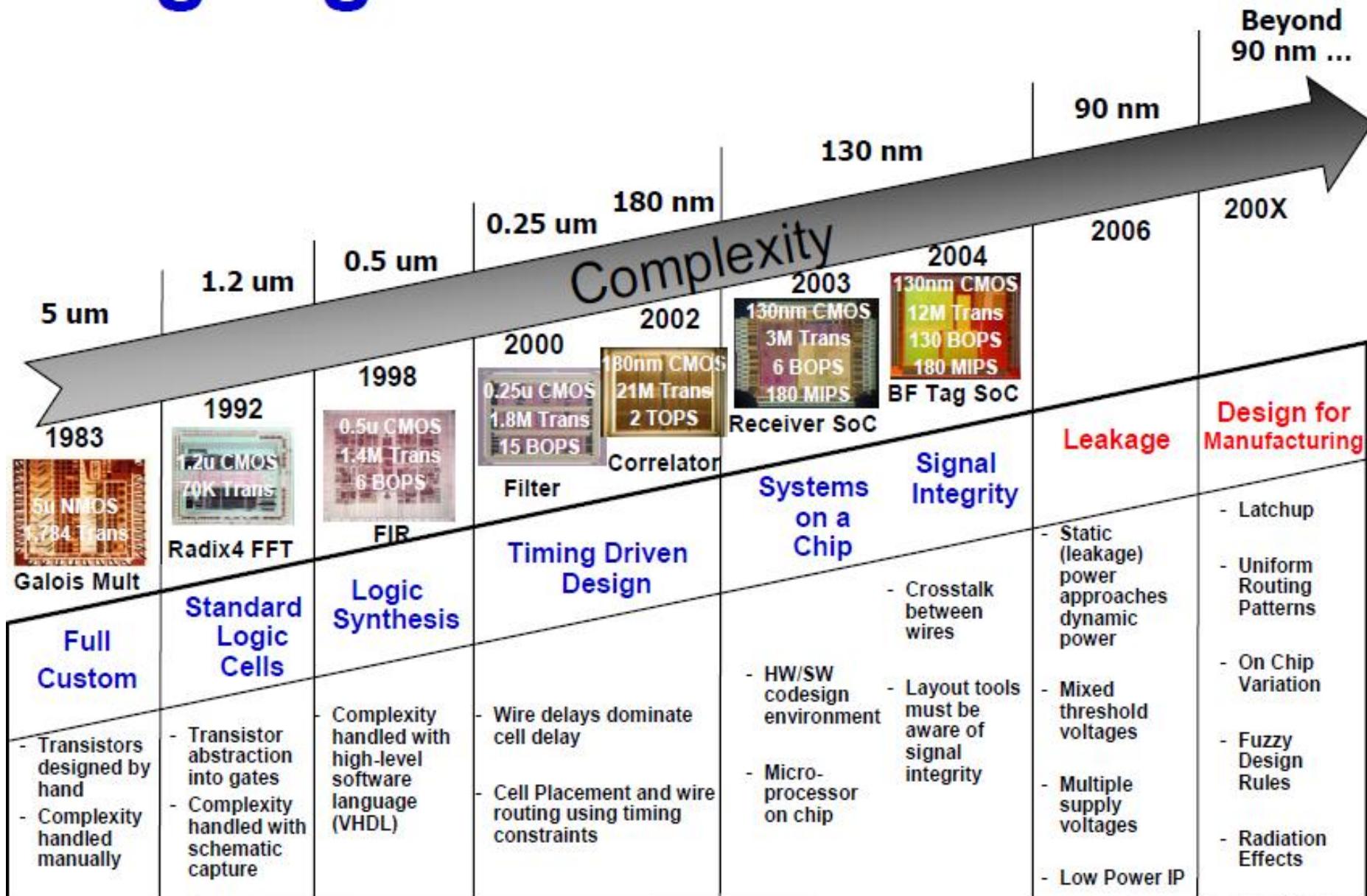
With scaling the reachable radius of a buffer decreases  
→ we need more and more buffers



repeaters required to buffer Itanium global interconnects

- A corner-to-corner (BL-UR) wire in Itanium (180nm) requires 6 repeaters to span die
- Repeaters consume chip area; consume power; add vias

# Highlight



# Conclusions

- High-speed design is a requirement for many applications
- Low-power design is also a requirement for IC designers.
- A new way of THINKING to simultaneously achieve both!!!
- Low power impacts in the cost, size, weight, performance, and reliability.
- Variable  $V_{dd}$  and  $V_t$  is a trend (DVS and DFS)
- CAD tools high-level power estimation and management
- Pay attention to **nano electro mechanical Systems (NEMS)** – lots of problems and potential is great.

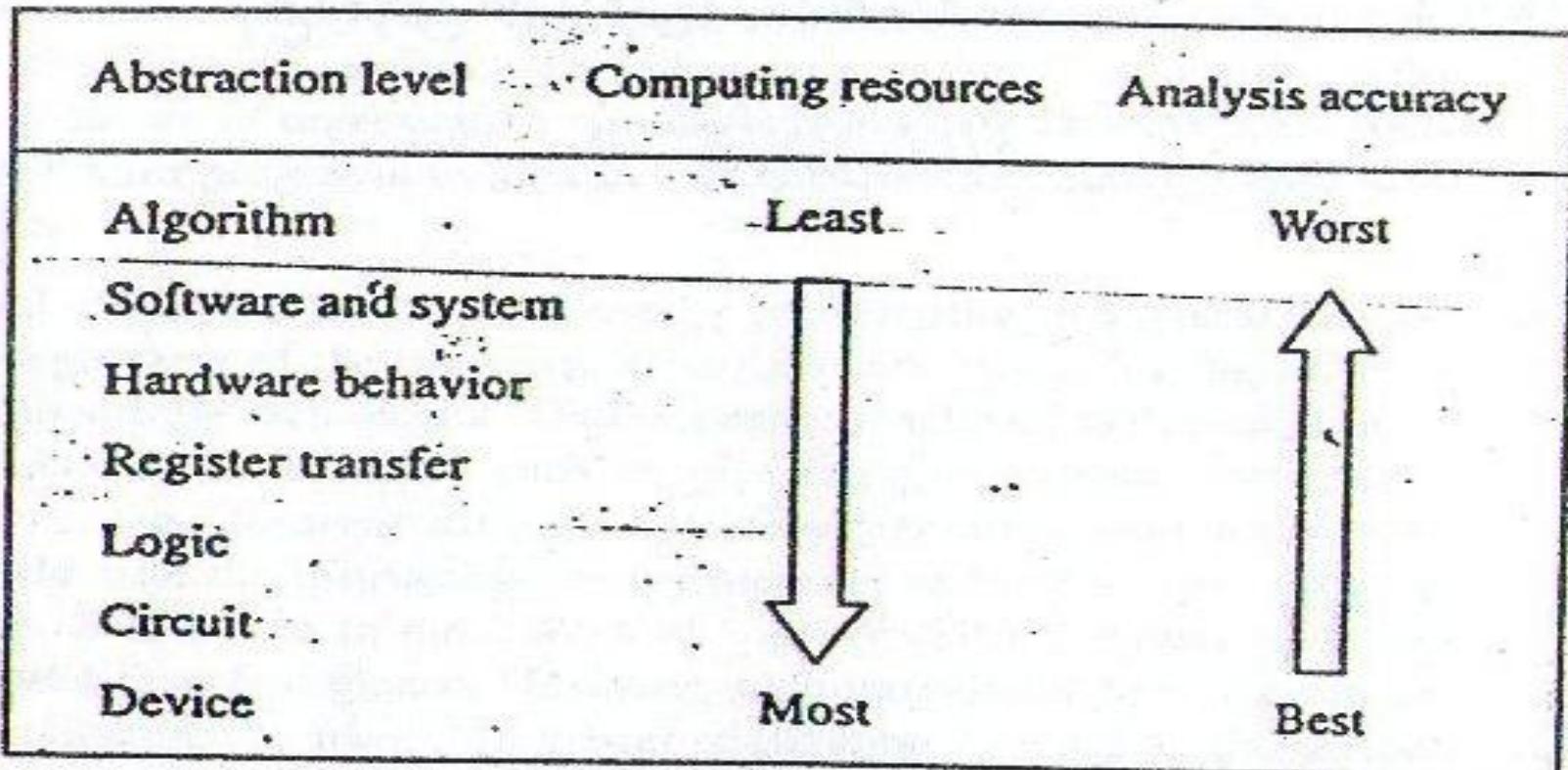
# Unit II

## Power estimation

# Simulation Power analysis

- Computer simulation has been applied to VLSI design for several decades
- Simulation programs operate on mathematical models which mimic the physical laws and properties of the object under simulation
- Simulation is used for
  - Functional verification
  - Performance
  - Cost
  - Reliability
  - Power analysis
- Digital logic simulation examples
  - VHDL (very high Speed IC Hardware Description Language)
  - Verilog

# Computing resources and analysis accuracy at various abstraction levels



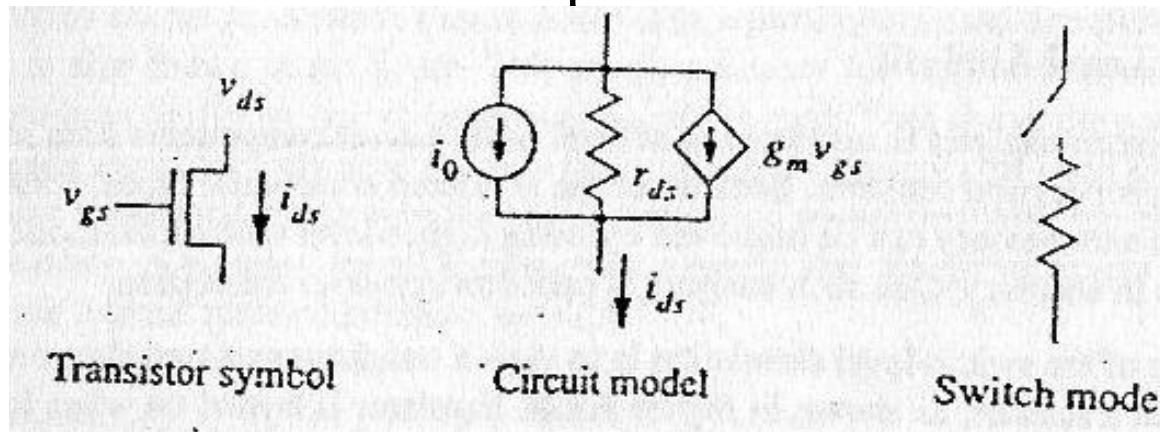
- The main difference between simulation at different levels of design abstraction is the tradeoff between computing resource (memory and CPU time) and accuracy of the results

# Points to remember

- Simulation based power estimation and analysis techniques have been developed and applied to VLSI design process
- Simulation at lower level design abstraction offers better accuracy at the expense of increased computer resource
- Circuit simulators such as SPICE attain excellent accuracy but cannot be applied to full chip analysis
- Logic simulation generally can handle full chip analysis but the accuracy is not as good and sometimes the execution speed is too low
- Behavioral level or functional level simulation offers rapid analysis but the accuracy is sacrificed

# SPICE circuit simulation

- SPICE (Simulation program with IC emphasis) is the power analysis tool at the circuit level
- SPICE operates by solving a large matrix of nodal current using the Kirchoffs current law (KCL)
- The basic components of SPICE are the primitive elements of circuit theory such as resistors, capacitors, inductors, current sources and voltage sources
- Complex device models such as diodes and transistors are constructed from the basic components



Circuit level and switch level representation of a transistor

# SPICE

- Device models are subsequently used to construct a circuit for simulation
- Circuit parameters such as voltage, current, charge etc are reported by SPICE with high degree of precision
- Circuit power dissipation can be directly derived from SPICE simulation
- Transient mode analysis for digital IC power estimation is widely used in SPICE
  - Involves DC solution of the circuit at time zero and makes small time increments
  - Used for dynamic behavior of the circuit over time

# Gate level logic simulation

- Simulation based gate level timing analysis has been a very mature technique in today's VLSI design
- The component abstraction at this level is logic gates and nets
- The circuit consists of components having defined logic behavior at its input and output such as NAND gates, latches and flip flops
- Gate level logic simulation software is one of the earliest CAD tools being developed
- In present world, Gate level logic simulator can perform full chip simulation up to several million gates

# Basics of Gate level analysis

- Event driven logic simulation
  - Events are zero-one logic switching of nets in a circuit at a particular simulation time point
- Cycle based simulators
- Gate level simulators
  - Hardware acceleration
  - Hardware emulation
- VHDL and verilog are two popular languages used to describe gate-level design

# Capacitive power estimation

- The major advantage of gate level power analysis is that the  $P = C V^2 f$  can be computed precisely
- In non logic abstraction such as SPICE, the notion of the frequency of a node is not well defined because it has an analog waveform that is potentially non-periodic and non-digital
- In logic simulation, the switching activities of each node can be monitored to determine its frequency
- The capacitive power dissipation of the circuit is

$$P_{cap} = \sum_{net i} C_i V^2 f_i$$

where

$$f_i = t_i / (2T)$$

(1)

T – Simulation time elapsed

C - Capacitance

ti – Counter variable

# Internal switching energy

- The Equation (1) computes the power dissipated due to charging and discharging of node capacitance
- Switching activities are not accounted
- Short circuit power is also not captured
- The dynamic power dissipated inside the logic cell is called internal power which consists of short circuit power and charging and discharging of internal nodes
- Idea is to simulate the “dynamic energy dissipation events” of the gate with SPICE or other lower level power simulation tools
- Switching from 1 to 0 or vice versa consumes some amount of dynamic energy internally

# Internal switching energy

- Computation of dynamic internal power uses the concept of logic events
- Each gate has a pre-defined set of logic events in which a quantum of energy is consumed for each event
- The energy value of each event can be computed with SPICE circuit simulation
- The total dynamic internal power dissipation is given by

$$P_{\text{int}} = \sum_{\text{gate } g} \sum_{\text{event } e} E(g, e) f(g, e)$$

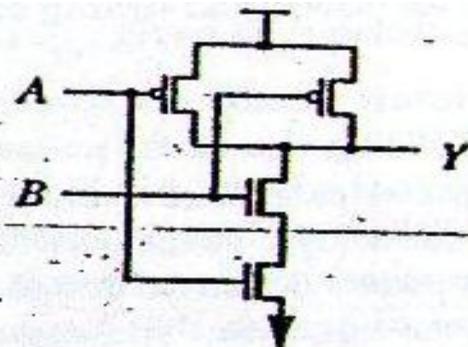
Where  $E(g, e)$  is the energy of the event  $e$  of gate  $g$  obtained from logic gate characterization

$f(g, e)$  is the occurrence frequency of the event on the gate observed from logic simulation

$E(g, e)$  depends on process conditions, operating voltage, temperature, output loading capacitance, input signal slopes etc

# Internal switching energy

- For example a simple 4 transistor CMOS NAND gate has four dynamic energy dissipation events as shown in below Fig.



(a) A 4-transistor CMOS NAND gate.

A	B	Y	Dyn energy (pJ)
I	r	f	1.67
I	f	r	1.39
r	I	f	1.94
f	I	r	1.72

(b) Dynamic energy dissipation events.

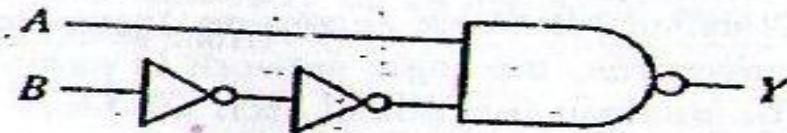
A	B	Y	Static power (pW)
0	0	I	5.05
0	I	I	13.1
I	0	I	5.10
I	I	0	28.5

(c) Static power dissipation states.

Dynamic events and static states of a 2-input CMOS NAND gate

# Internal switching energy

- The first implementation has only four energy dissipation events
- Second implementation has two additional events due to switching of its internal nodes



A	B	Y
I	r	s
I	s	r
r	I	s
s	I	r

A	B	Y
I	r	s
I	s	r
r	I	s
s	I	r
o	r	r
o	s	I

Two different implementation of NAND gate result in different dynamic energy dissipation events

# Static state power

- A similar event characterization idea can also be used to compute the static power dissipation of a logic gate
- Here, the power dissipation depends on the state of the logic gate
- The total static power is

$$P_{stat} = \sum_{\text{gate } g} \sum_{\text{state } s} P(g, s) \frac{T(g, s)}{T}$$

A	B	Y	Static power (pW)
0	0	1	5.05
0	1	1	13.1
1	0	1	5.10'
1	1	0	28.5

(c) Static power dissipation states.

- $P(g, s)$  is the static power dissipation of gate  $g$  at state  $s$  obtained from characterization
- State duration  $T(g, s)$  is obtained from logic simulation
  - It is the total time gate  $g$  stays at state  $s$

# Gate level capacitance estimation

- As discussed earlier, capacitance is the most important attribute that affects the power dissipation of CMOS circuits
- Capacitance also has an impact on delays and signal slopes of logic gates
- Change in gate delay may affect the switching characteristics of the circuit and influence power dissipation
- Short circuit current is affected by the input signal slopes and output capacitance loading
- Thus, capacitance has a direct and indirect impact on power analysis
- The accurate estimation of capacitance is important for power analysis and optimization

# Gate level capacitance estimation

- Two types of parasitic capacitance exist in CMOS circuits:
  - Device parasitic capacitance
  - Wiring capacitance
- Parasitic capacitance of MOS devices is associated with terminals
- The gate capacitance depends on the oxide thickness of the gate that is process dependent
- Design dependent factors are
  - Width, length and shape of the gate
- In general a larger transistor has more capacitance in all its terminals
- The second source of parasitic capacitance is wiring capacitance
  - Depends on the layer, area and shape of the wire

# Gate level power analysis

- The event driven gate level power simulation is summarized as follows:
  - Run logic simulation with a set of input vectors
  - Monitor the toggle count for each net
  - Obtain capacitive power dissipation  $P_{cap}$
  - Monitor the dynamic energy dissipation events of each gate
  - Obtain internal switching power dissipation  $P_{int}$
  - Monitor the static power dissipation states of each gate
  - Obtain static power dissipation  $P_{stat}$
  - Sum up all power dissipation components

$$P = P_{cap} + P_{int} + P_{stat}$$

# Signal glitches - Gate level power analysis

- The major disadvantage of gate level analysis is that signal glitches cannot be modeled precisely
- Signal glitches are inherently analog phenomena and the simplified zero-one logic model in gate-level analysis fails to capture their effects
- The presence of glitches is very sensitive to the signal and gate delays of the circuit
- As signal glitches are significant source of power dissipation in some VLSI circuits, it cannot be ignored
- However, it is difficult for any analysis model above the logic level to account for the signal glitches precisely

# Architecture level analysis

- Over the years, the design abstraction has moved from the mask, the transistors, to gates and now to the architecture level
- Architecture level abstraction is called as block level or macro level design
- Building blocks
  - Registers
  - Adders
  - Multipliers
  - Buses
  - Multiplexers
  - Memories
  - State machines etc
- Today, architecture level power analysis is becoming more important because more digital circuits are now synthesized from architecture description

# Power model based on activities

- One way to characterize the **architectural** components is to express the power dissipation as a function of the number of bits of the components and their operating frequencies

- Eg: the power dissipation of an adder can be expressed as:

$$P = (n K_1 + K_2) f$$

Where n – number of bits

f – frequency of the addition operation

$K_1$  and  $K_2$  – empirical coefficients derived from gate level simulation

- The above model depends only on the operating frequency and size of the adder
- The model does not account the data dependency of the power dissipation

# Power model based on activities

- A more accurate model that can capture data dependency is to characterize the power dissipation as

$$P = \sum_{\text{input } i} K_i f_i$$

- For multipliers, the model is given by

$$P = \sum_{\text{input } i} K_i f_i + \sum_{\text{input } j} K_j f_j$$

- Since it is tedious to characterize  $K_i$  for each input  $i$ , the  $P$  can be simplified as follows

$$P = K_1 \sum_{\text{input } i} f_i + K_2 \sum_{\text{input } j} f_j$$

$$P = K_1 f_{in} + K_2 f_{out}$$

# Power dissipation based on Component operations

- In this model the power dissipation is expressed in terms of the frequency of some primitive operations of an architecture component
- The power dissipation is given by

$$P = K_1 f_{read} + K_2 f_{write}$$

- The parameters  $f_{read}$  and  $f_{write}$  are the frequencies of READ and WRITE operations, respectively
- Coefficients  $K_1$  and  $K_2$  are obtained from characterization and properties of the component

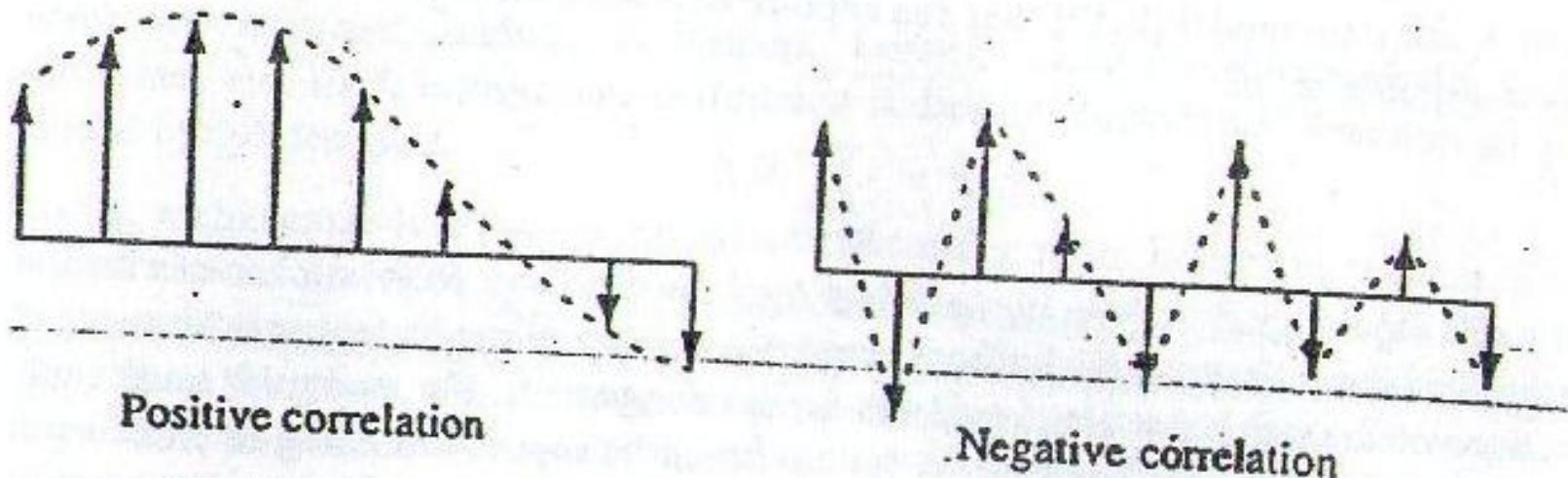
# Abstract statistical power models

- The previous power models have to make certain assumptions on the statistical behavior of components in order to model the power dissipation efficiently
- There are several fundamental issues in composing a power model
  - Which parameters have major influences on the power dissipation ? Size, signal frequency, operation frequency etc
  - What is the relationship of the power dissipation with the chosen parameters? Equation or lookup table
- The solution depends on the type, implementation and accuracy level tolerated by the analysis

# Data correlation analysis in DSP systems

- Sample correlation
  - Refers to the property that successive data samples are very close in their numerical values and consequently their binary representations have many bits in common
- Negative correlation (anti-correlation)
  - Successive samples jump from a large positive value to a large negative value
- Positive or negative correlation has a significant effect on the power dissipation of a DSP system because of the switching activities on the system data path
- Hence, here we discuss how to estimate power of the architecture level component based on the frequency and correlation measures of the data stream

# Data correlation analysis in DSP systems

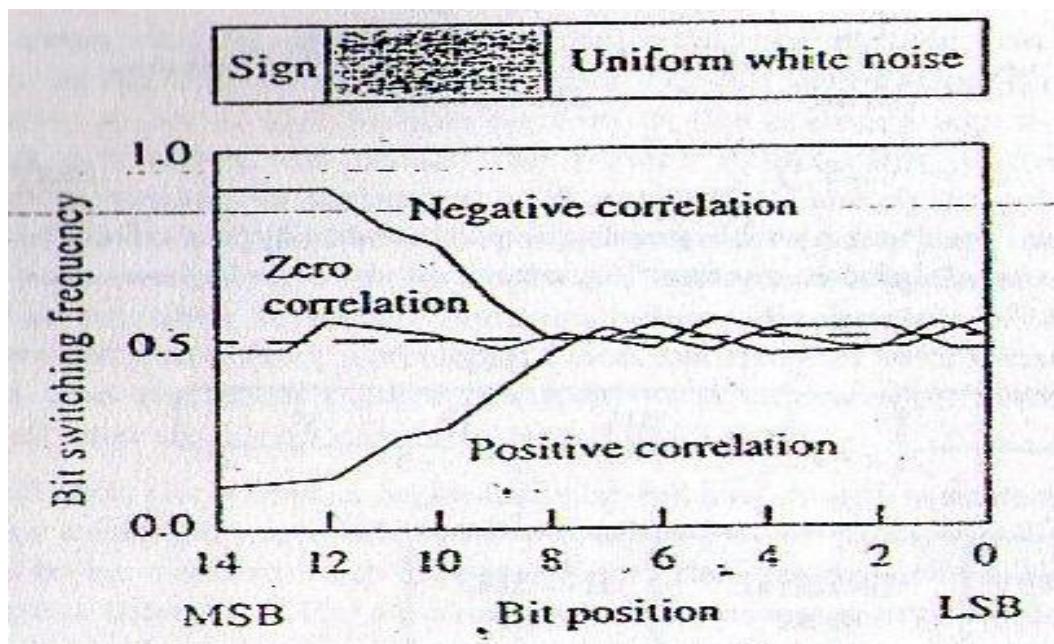


## Correlation resulting from sampling analog signals

- Dual bit type signal model
  - The effect of data correlation on power dissipation depends on the numerical representation of the digital system
    - Two's complement
    - Signed magnitude
  - Uniform white noise region
    - Bits toggle in a random fashion

# Data correlation analysis in DSP systems

- The data stream can be characterized with only a few parameters
  - Sample frequency
  - Data correlation factor from -1.0 to +1.0
  - The sign bit and uniform white noise regions with two integers
- Such characterization of data signals is called the dual bit type model proposed by Landman and Rabaey



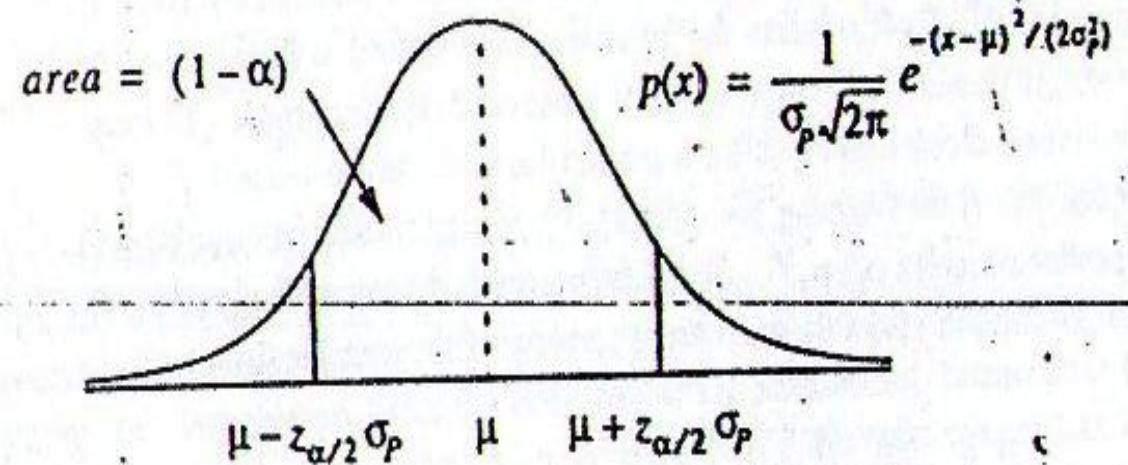
Effects of data correlation on bit switch frequency

# Monte Carlo Simulation

- Statistical estimation of mean

$$P = \frac{(p_0 + p_1 + \dots + p_N)}{N}$$

Where p refers to the power samples



Normal distribution curve of sample average P

$(1 - \alpha)$	$z_{\alpha/2}$
0.9	1.65
0.95	1.96
0.99	2.58
0.998	3.00
1.0	$\infty$

Table of Z distribution

# Monte Carlo power Simulation

- As we simulate the circuit, we collect samples  $p_i$  at a fixed interval to compute the sample mean  $P$  and sample variance  $S^2$

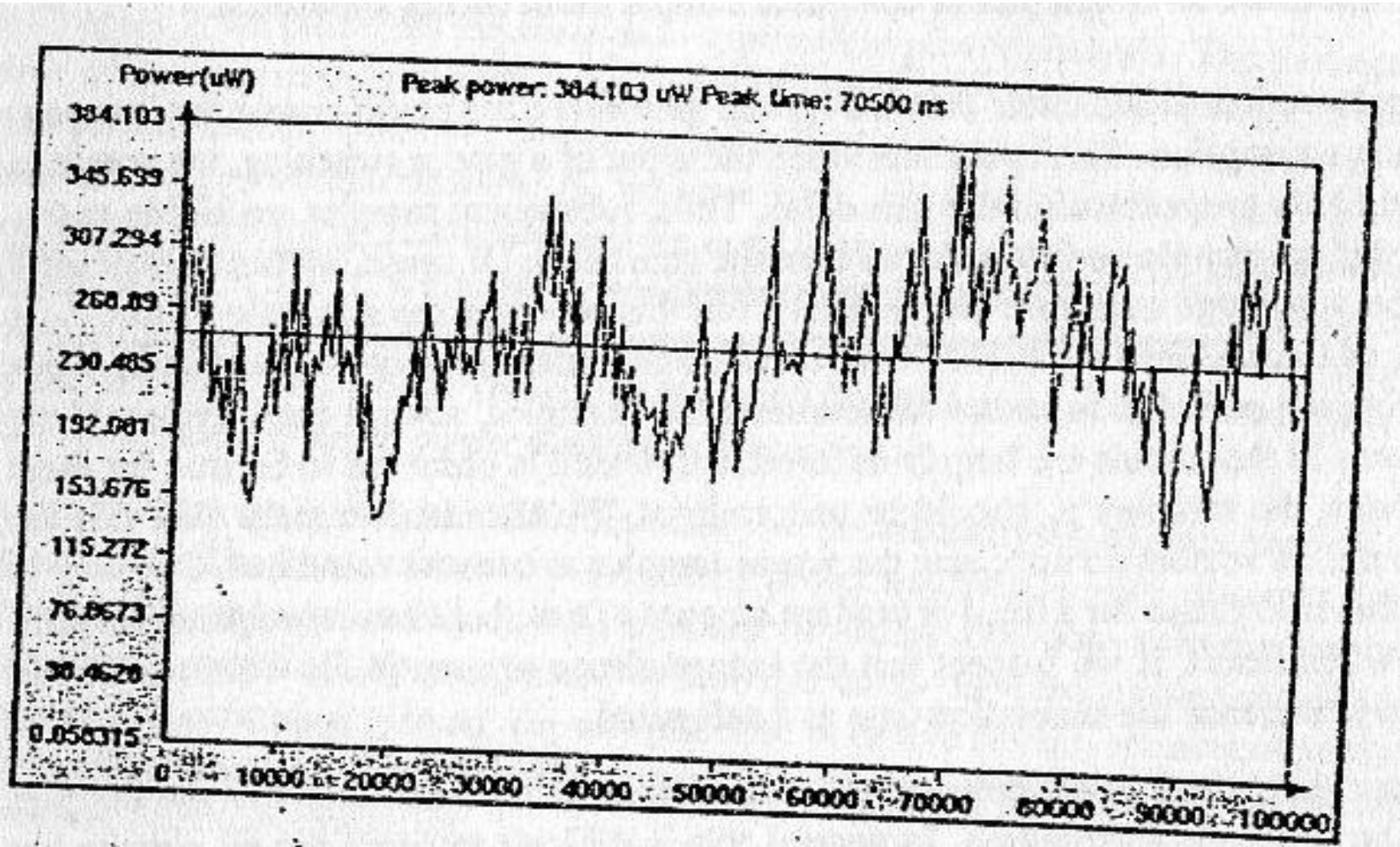
$$S^2 = \frac{1}{N} \sum_{i=1}^N (p_i - P)^2$$

- Given the confidence level (1-alpha) and error tolerance, we check whether the following equation is satisfied

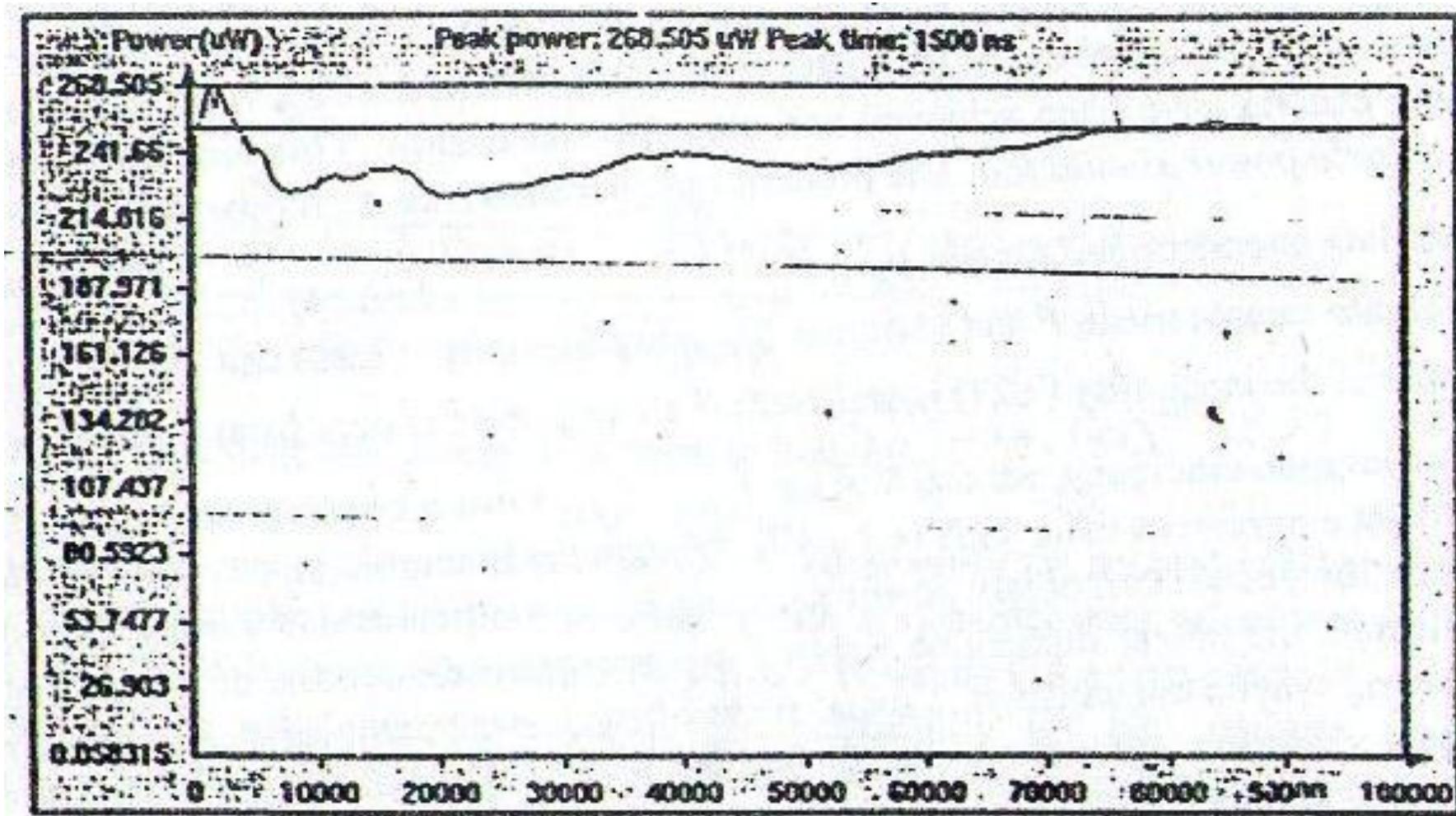
$$N \geq \left( \frac{t_{[N-1, \alpha/2]} S}{\varepsilon P} \right)^2$$

- If so, we claim that the stopping criteria have been achieved and stop the simulation. This process is called Monte Carlo power simulation

# Graph plot of instantaneous power samples



# Graph plot of computed sample mean during simulation



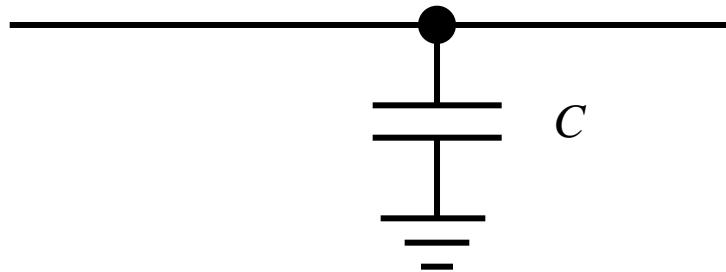
# Power Analysis: Probabilistic Methods

# Basic Idea

- View signals as a random processes

$$\text{Prob}\{s(t) = 1\} = p_1$$

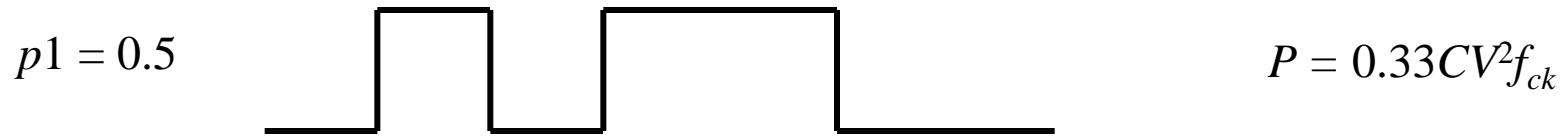
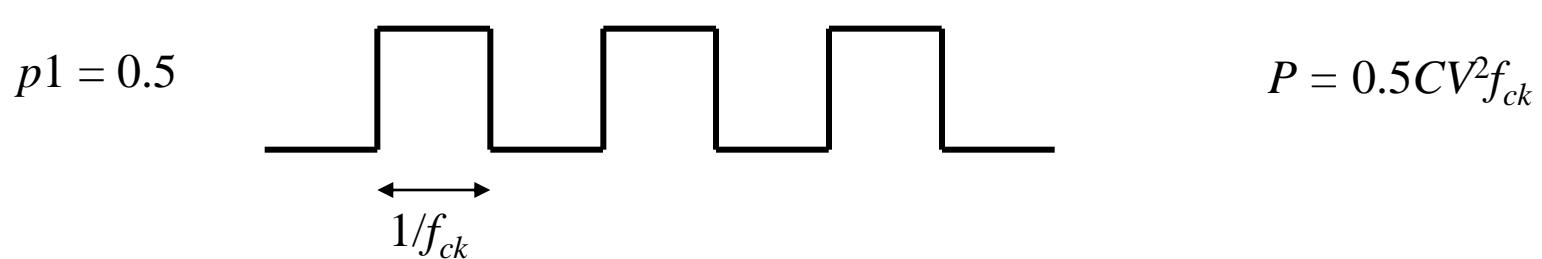
$$p_0 = 1 - p_1$$



$$0 \rightarrow 1 \text{ transition probability} = (1 - p_1) p_1$$

$$\text{Power, } P = (1 - p_1) p_1 CV^2 f_{ck}$$

# Source of Inaccuracy



Observe that the formula, Power,  $P = (1 - p1) p1 CV^2f_{ck}$ , is not Correct.

# Switching Frequency

Number of transitions per unit time:

$$T = \frac{N(t)}{t}$$

*For a continuous signal:*

$$T = \lim_{t \rightarrow \infty} \frac{N(t)}{t}$$

*T* is defined as the transition density.

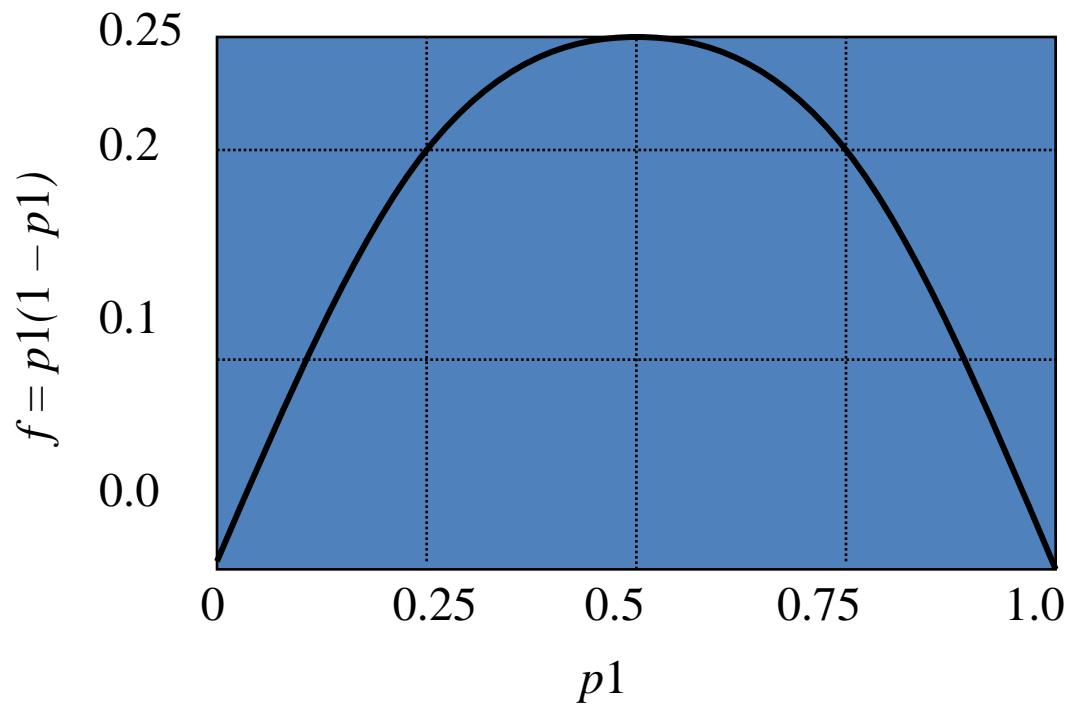
# Static Signal Probabilities

- Observe signal for interval  $t0 + t1$ 
  - Signal is 1 for duration  $t1$
  - Signal is 0 for duration  $t0$
  - Signal probabilities:
    - $p1 = t1/(t0 + t1)$
    - $p0 = t0/(t0 + t1) = 1 - p1$

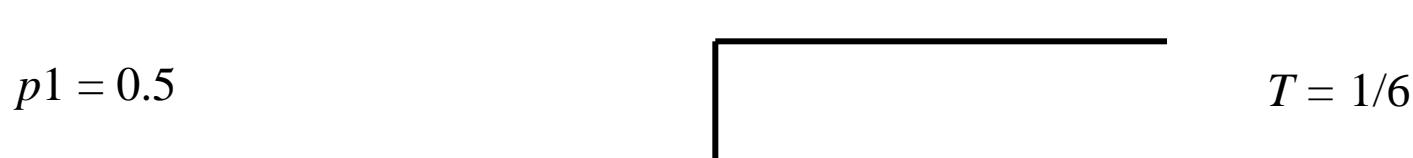
# Static Transition Probabilities

- Transition probabilities:
  - $T_{01} = p_0 \text{ Prob}\{\text{signal is } 1 \mid \text{signal was } 0\} = p_0 p_1$
  - $T_{10} = p_1 \text{ Prob}\{\text{signal is } 0 \mid \text{signal was } 1\} = p_1 p_0$
  - $T = T_{01} + T_{10} = 2 p_0 p_1 = 2 p_1 (1 - p_1)$
- Transition density:  $T = 2 p_1 (1 - p_1)$
- Transition frequency:  $f = T/2$
- Power =  $CV^2T/2$  (correct formula)

# Static Transition Frequency



# Inaccuracy in Transition Density



Observe that the formula,  $T = 2 p1 (1 - p1)$ , is not correct.

# Cause for Error and Correction

- Probability of transition is not independent of the present state of the signal
- Consider probability  $p_{01}$  of a  $0 \rightarrow 1$  transition,
- Then  $p_{01} \neq p_0 p_1$
- We can write  $p_1 = (1 - p_1)p_{01} + p_1 p_{11}$

$$p_1 = \frac{p_{01}}{1 - p_{11} + p_{01}}$$

# Correction (Cont.)

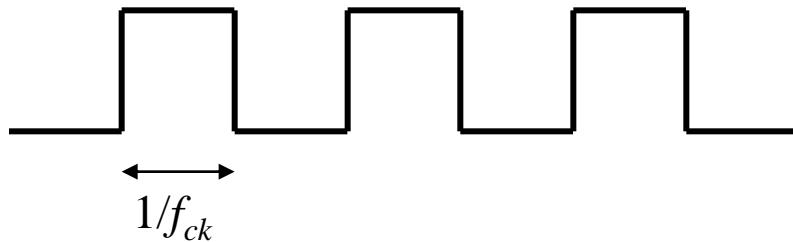
- Since  $p_{11} + p_{10} = 1$ , i.e., given that the signal was previously 1, its present value can be either 1 or 0
- Therefore,

$$p_1 = \frac{p_{01}}{p_{10} + p_{01}}$$

This uniquely gives signal probability as a function of transition probabilities

# Transition and Signal Probabilities

$$p01 = p10 = 0.5$$



$$p1 = 0.5$$

$$p01 = p10 = 1/3$$



$$p1 = 0.5$$

$$p01 = p10 = 1/6$$



$$p1 = 0.5$$

# Probabilities: $p0, p1, p00, p01, p10, p11$

- $p01 + p00 = 1$
- $p11 + p10 = 1$
- $p0 = 1 - p1$
- $$p1 = \frac{p01}{p10 + p01}$$

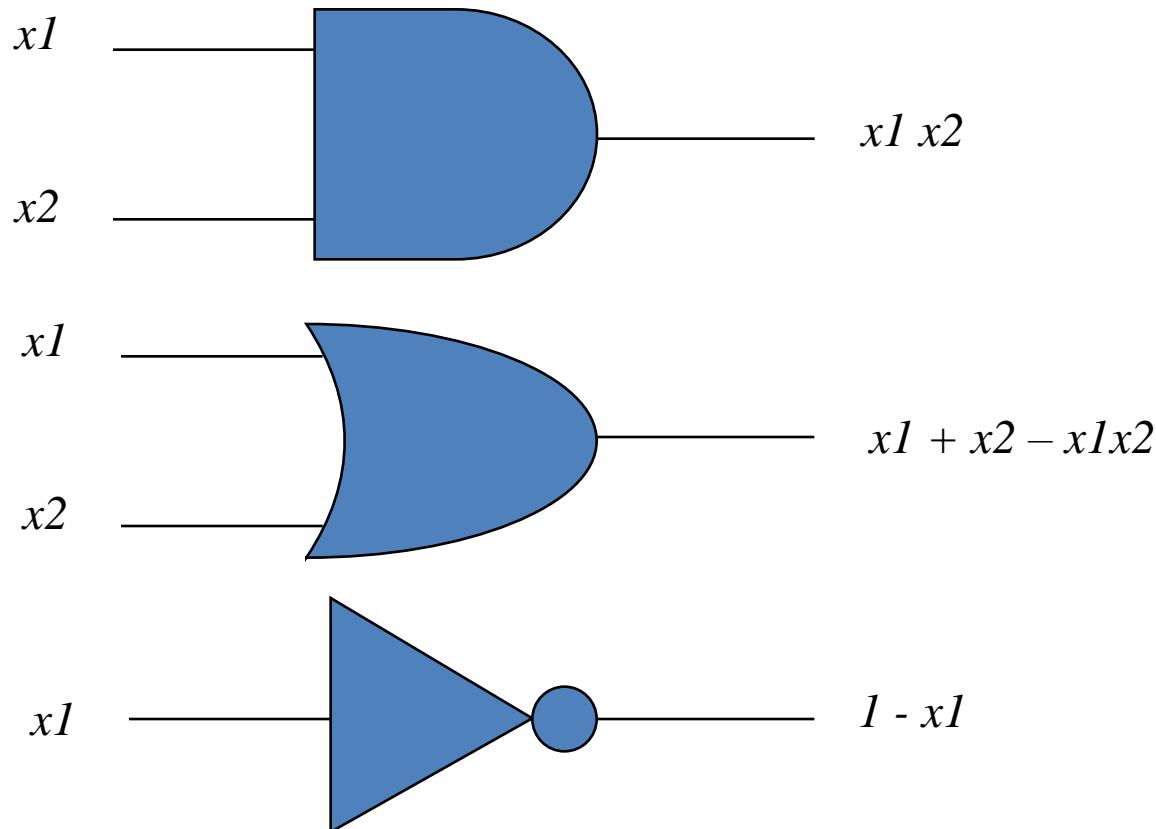
# Transition Density

- $T = 2 p1(1 - p1) = p0 p01 + p1 p10$   
 $= 2 p10 p01/(p10 + p01)$   
 $= 2 p1 p10 = 2 p0 p01$

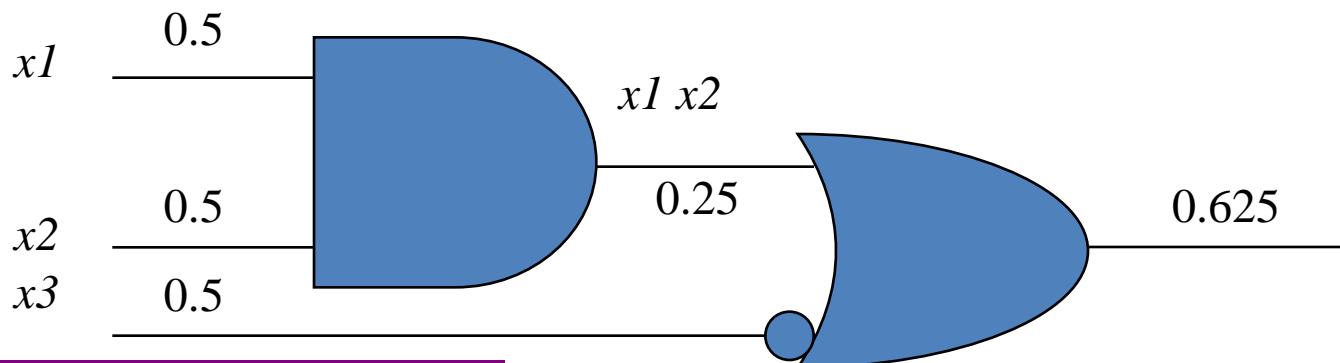
# Power Calculation

- Power can be estimated if transition density is known for all signals.
- Calculation of transition density requires
  - Signal probabilities
  - Transition densities for primary inputs; computed from vector statistics

# Signal Probabilities



# Signal Probabilities

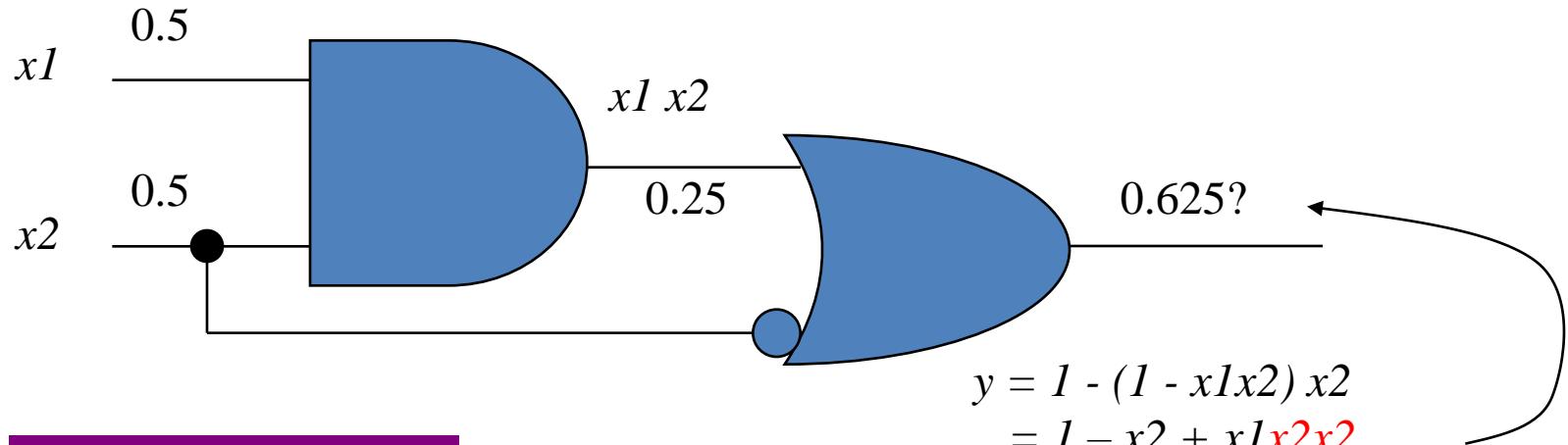


$X_1$	$X_2$	$X_3$	$Y$
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

$$\begin{aligned}
 y &= 1 - (1 - x_1 x_2) x_3 \\
 &= 1 - x_3 + x_1 x_2 x_3 \\
 &= 0.625
 \end{aligned}$$

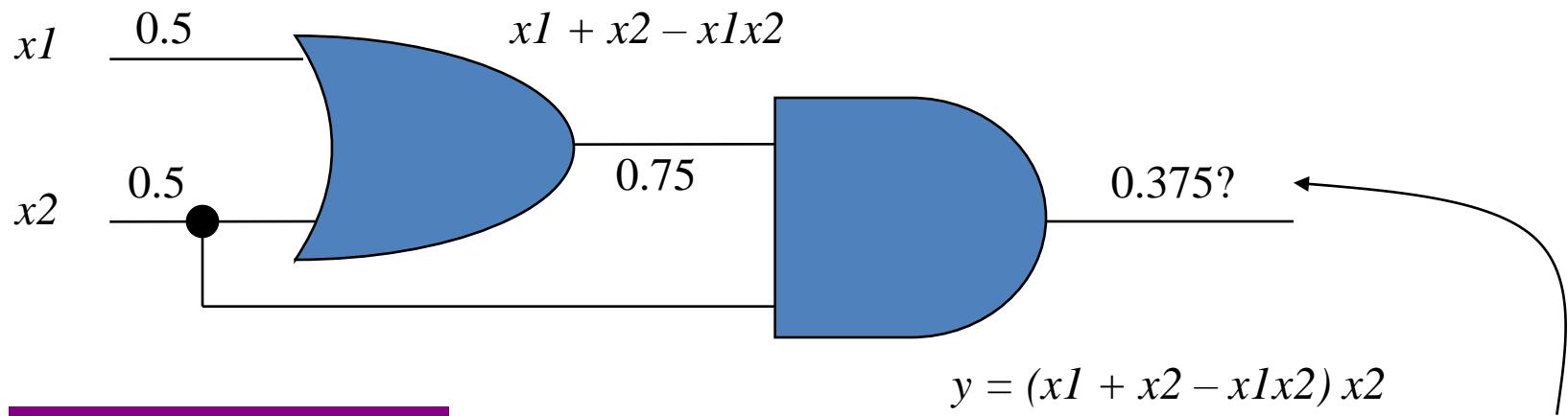
*Ref: K. P. Parker and E. J. McCluskey,  
“Probabilistic Treatment of General  
Combinational Networks,” IEEE Trans. on  
Computers, vol. C-24, no. 6, pp. 668-670, June  
1975.*

# Correlated Signal Probabilities



X1	X2	Y
0	0	1
0	1	0
1	0	1
1	1	1

# Correlated Signal Probabilities



X1	X2	Y
0	0	0
0	1	1
1	0	0
1	1	1

$$\begin{aligned}y &= (x1 + x2 - x1x2)x2 \\&= x1x2 + \cancel{x2x2} - x1\cancel{x2x2} \\&= x1x2 + x2 - x1x2 \\&= x2 \\&= 0.5\end{aligned}$$

# Observation

- Numerical computation of signal probabilities is accurate for fanout-free circuits.

# Remedies

- Use Shannon's expansion theorem to compute signal probabilities.
- Use Boolean difference formula to compute transition densities.

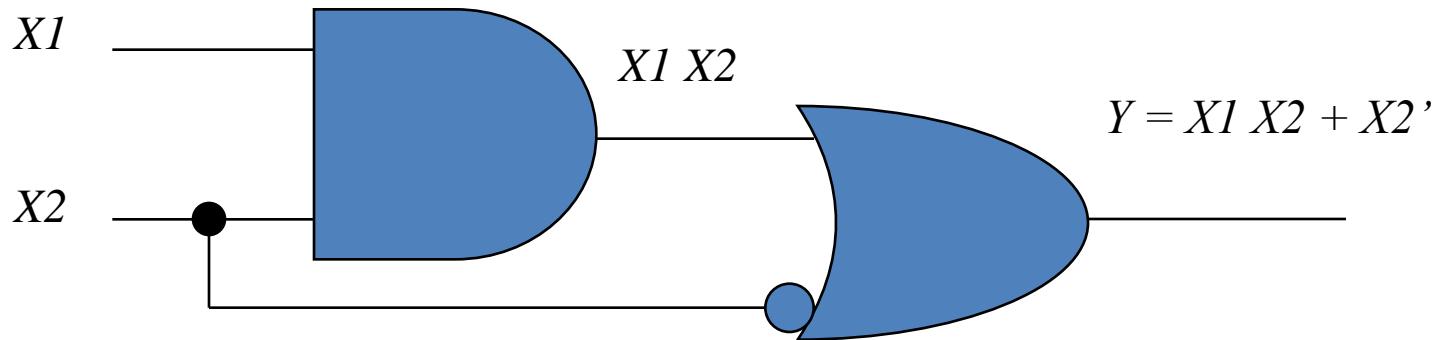
# Shannon's Expansion Theorem

- C. E. Shannon, “A Symbolic Analysis of Relay and Switching Circuits,” *Trans. AIEE*, vol. 57, pp. 713-723, 1938.
- Consider:
  - Boolean variables,  $X_1, X_2, \dots, X_n$
  - Boolean function,  $F(X_1, X_2, \dots, X_n)$
- Then  $F = \sum_i F(X_i=1) + \sum_i F(X_i=0)$
- Where
  - $X_i'$  is complement of  $X_i$
  - Cofactors,  $F(X_i=j) = F(X_1, X_2, \dots, X_i=j, \dots, X_n)$ ,  $j = 0$  or  $1$

# Expansion About Two Inputs

- $$F = X_i X_j F(X_i=1, X_j=1) + X_i X_j' F(X_i=1, X_j=0) \\ + X_i' X_j F(X_i=0, X_j=1) + X_i' X_j' F(X_i=0, X_j=0)$$
- In general, a Boolean function can be expanded about any number of input variables.
- Expansion about  $k$  variables will have  $2^k$  terms.

# Correlated Signal Probabilities



$X1$	$X2$	$Y$
0	0	1
0	1	0
1	0	1
1	1	1

Shannon expansion about the reconverging input:

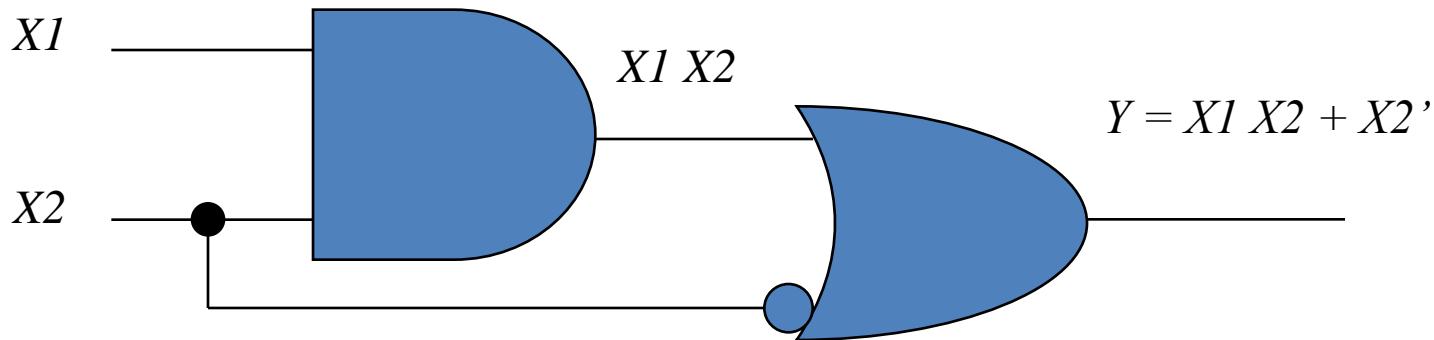
$$\begin{aligned} Y &= X2 \cdot Y(X2=1) + X2' \cdot Y(X2=0) \\ &= X2 \cdot (X1) + X2' \cdot (1) \end{aligned}$$

# Correlated Signals

- When the output function is expanded about all reconverging input variables,
  - All cofactors correspond to fanout-free circuits.
  - Signal probabilities for cofactor outputs can be calculated without error.
  - A weighted sum of cofactor probabilities gives the correct probability of the output.
- For two reconverging inputs:

$$\begin{aligned} f = & x_i x_j f(X_i=1, X_j=1) + x_i(1-x_j) f(X_i=1, X_j=0) \\ & + (1-x_i)x_j f(X_i=0, X_j=1) + (1-x_i)(1-x_j) f(X_i=0, X_j=0) \end{aligned}$$

# Correlated Signal Probabilities



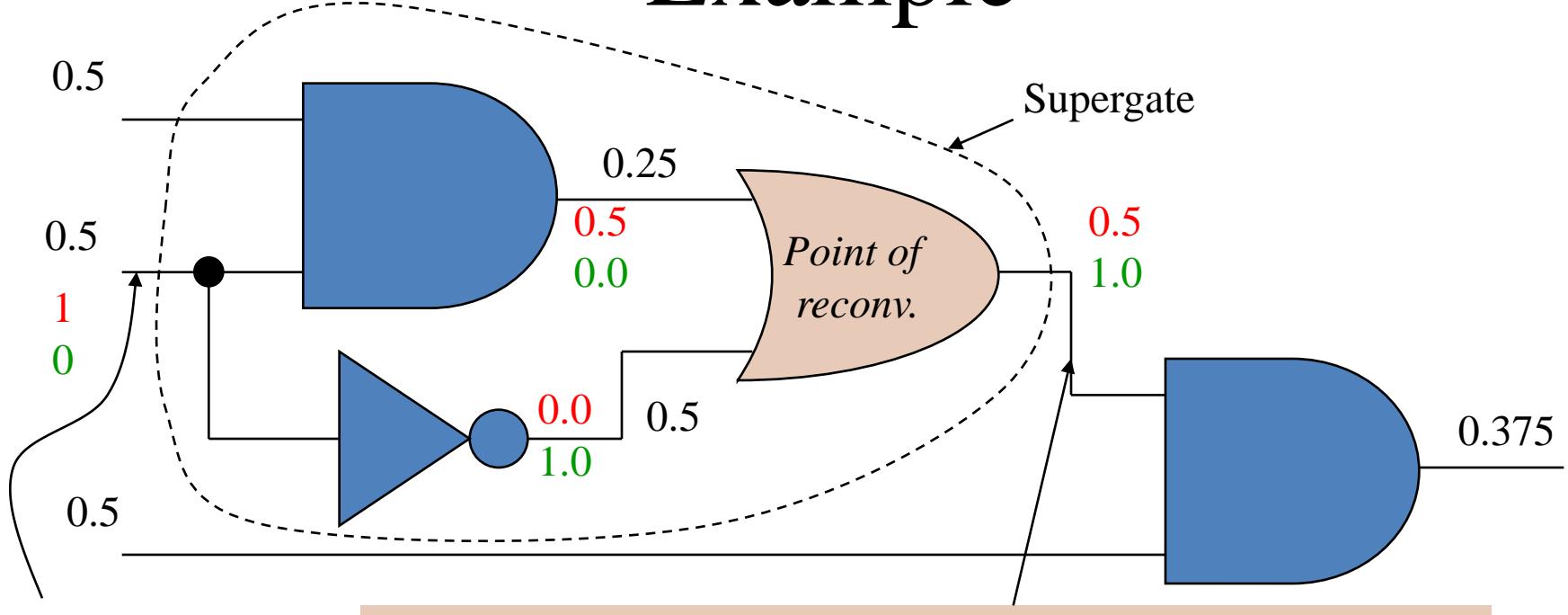
$X1$	$X2$	$Y$
0	0	1
0	1	0
1	0	1
1	1	1

Shannon expansion about the reconverging input:

$$\begin{aligned} Y &= X2 Y(X2=1) + X2' Y(X2=0) \\ &= X2 (X1) + X2' (1) \end{aligned}$$

$$\begin{aligned} y &= x2 (0.5) + (1-x2) (1) \\ &= 0.5 (0.5) + (1-0.5) (1) \\ &= 0.75 \end{aligned}$$

# Example



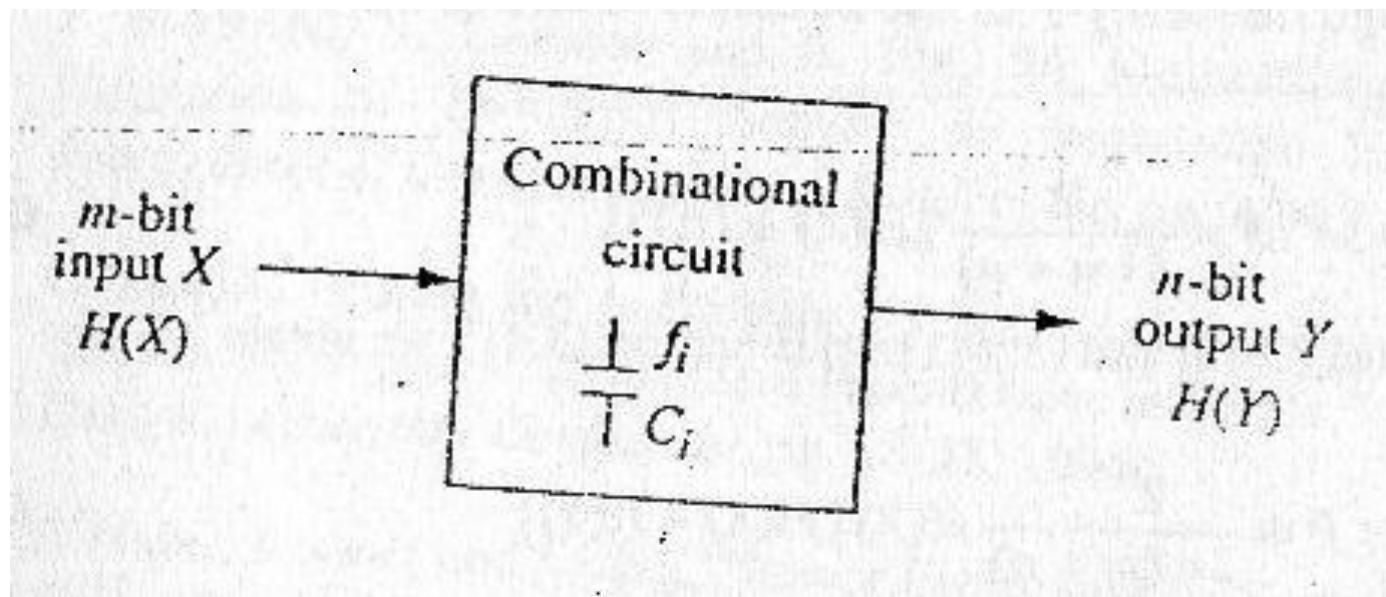
Signal probability for supergate output  
 $= 0.5 \text{ Prob}\{\text{rec. signal} = 1\} + 1.0 \text{ Prob}\{\text{rec. signal} = 0\}$   
 $= 0.5 \times 0.5 + 1.0 \times 0.5 = 0.75$

S. C. Seth and V. D. Agrawal, “A New Model for Computation of Probabilistic Testability in Combinational Circuits,” *Integration, the VLSI Journal*, vol. 7, no. 1, pp. 49-75, April 1989.

# Signal entropy

- The entropy of a set of logic signals is a measure of its randomness
- Entropy correlates to the average switching frequency of the signals
- Skewed occurrence probability gives a low probability measure
- If signal switching is active, it maximizes the entropy of the signals
- These observations prompts the idea of using signal entropy for power estimation

# Power estimation of combinational logic using entropy analysis



# Entropy

- Entropy based approach
  - Entropy: Measure of uncertainty in a random variable
  - Entropy H of a random variable x is given by

$$H(x) = p \log \frac{1}{p} + (1-p) \log \frac{1}{1-p}$$

- p: probability of x being 1

- Recall that

$$P_{avg} \propto D_{avg} \cdot GE \cdot C_{avg}$$

- $D_{avg}$ : Average node switching activity
- GE: Gate equivalents
- $C_{avg}$ : Average gate capacitance

# Entropy

- Hypothesis
  - Can  $D_{avg}$  be estimated only from knowledge of inputs and output behavior?

- Answer: Yes!

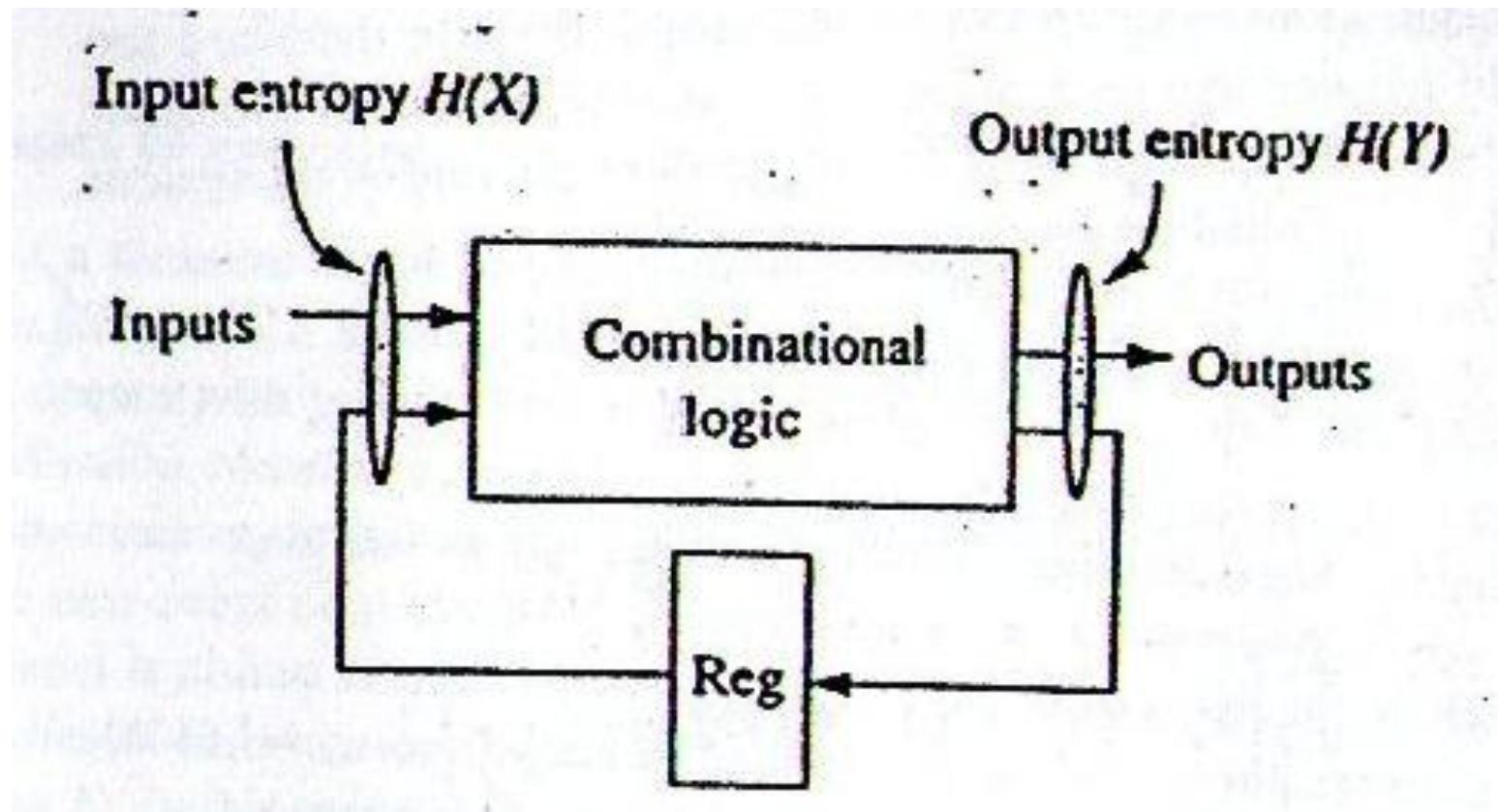
$$P_{avg} \propto H.GE.C_{avg}$$

- Entropy H is given by

$$H \approx \frac{2/3}{n+m} (H(X) + H(Y))$$

- $H(X)$  and  $H(Y)$  are respectively the input and output entropies

# Entropy analysis of a sequential circuit



# Entropy

- Entropy Based Power Estimation Methodology:
  - Run a structural RTL simulation to measure input/output entropies
  - Using input/output entropies, estimate  $P_{avg}$  for the combinational block
  - Use other techniques to estimate latch and clock power

## **UNIT-III**

**Low Power Design Circuit level**

# Power consumption in circuits

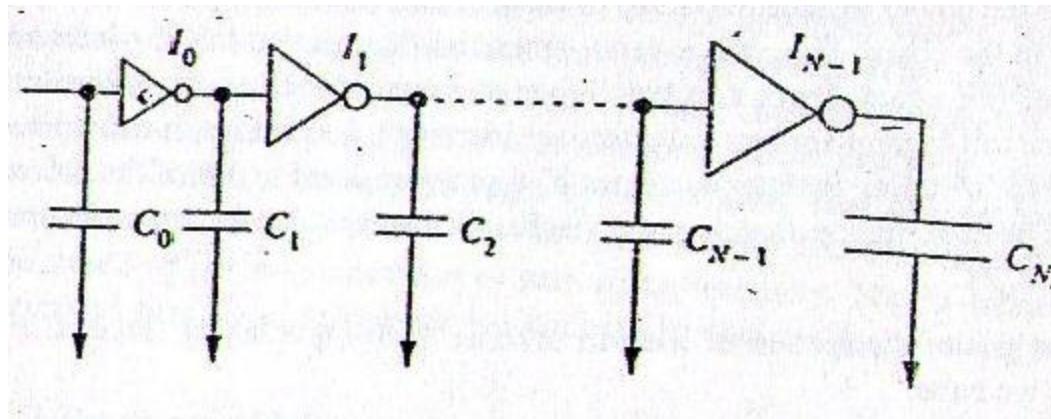
- The power reduction techniques at the circuit level are quite limited if compared with the other techniques at higher abstraction levels
- At the circuit level , percentage power reduction in the teens is considered good
- However, circuit techniques can have major impact because some circuits, especially in cell- based design, are repeated thousands of times on a chip
- Therefore, circuit techniques with a small percentage improvement should not be overlooked
- Circuits designed manually can often be analyzed in great details
- This allows us to optimize the circuit speed, power and area to suit our specification
- One important circuit technique is the reduction of operating voltage
  - The general rule is to select the lowest voltage that is acceptable

# Transistor and Gate Sizing

- At the circuit level, transistors are the basic building blocks and a digital circuit can be viewed as a network of transistors with some additional parasitic elements such as capacitors and resistors
- Transistor sizes are the most important factor affecting the quality ie area, and power dissipation of a circuit
- Some studies assume that the sizing problem is a convex function and linear programming can be used to solve the sizing problem optimally
- Another problem encountered in cell based design is gate sizing
- The goal is to choose a set of gate sizes that best fits the design constraints

# Sizing an Inverter Chain

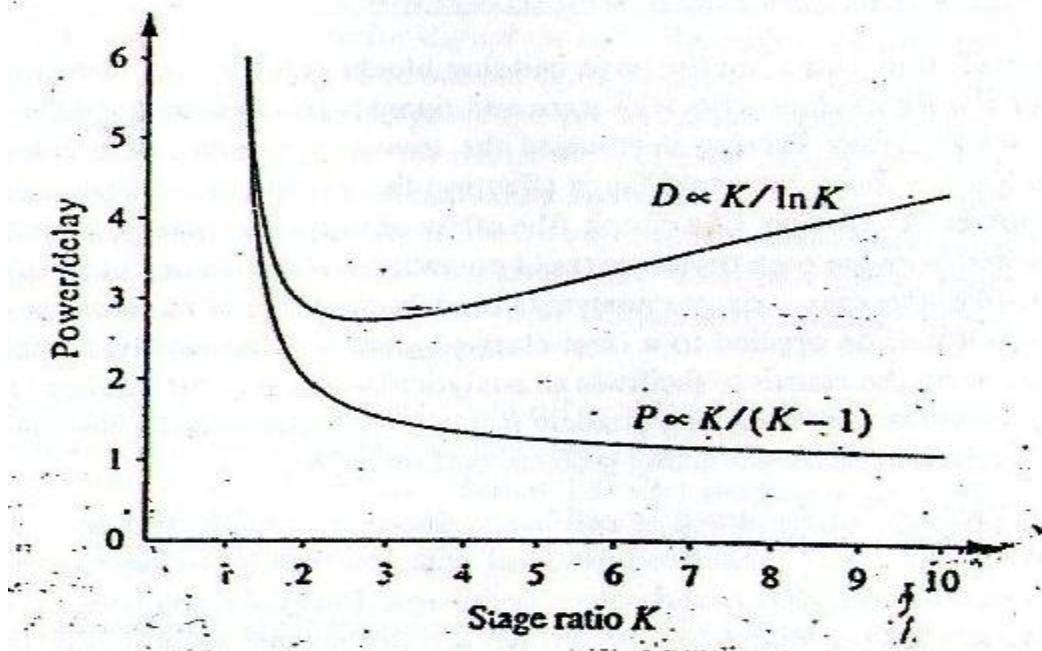
- The simplest transistor sizing problem is that of an inverter chain
- The general design problem is to drive a large capacitive load without excessive delay, area and power requirements
- A large inverter is required to drive the large capacitive load at the final stage



Sizing an inverter chain

# Sizing an Inverter Chain

- If the chain is too long, the signal delay will be too large due to intrinsic delay of each inverter
- If the chain is too short, the output signal slope will be very weak with long rise and fall times, which again causes long delay
- The challenge is to decide the length of the chain ie. How many inverters, and the size of each inverter

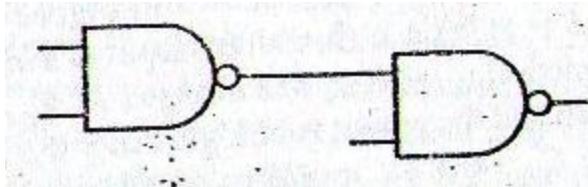


Graph plot of inverter chain delay and power dissipation

# Transistor and Gate Sizing for Dynamic Power Dissipation

- As seen in the previous graph, a large gate is required to drive a large load with acceptable delay but requires more power
- This is similar to the classical delay area tradeoff problem
- If we are not allowed to restructure the transistor network, the sizing for dynamic power reduction generally has the same goal as the area reduction problem
- The basic rule is to use the smallest transistors or gates that satisfy the delay constraints
- To reduce dynamic power, the gates that toggle with higher frequency should be made smaller
- Although the basic rule sounds simple, the actual sizing problem is very complicated

# Transistor and Gate Sizing for Dynamic Power Dissipation



Gate sizing problem

- Consider a part of the circuit as shown above
- Suppose that the gates are not on the critical delay path and should be sized down
- W can size down the first gate, the second gate or both, subject to the available sizes in the cell library as long as the path delay is not violated
- If the path contains many gates, the optimization problem becomes very complicated
- Stack time
  - Used to express the timing constraints of the circuit
  - It is the difference between the signal required time and signal arrival time at the output of a gate

# Transistor and Gate Sizing for Dynamic Power Dissipation

- A positive stack time mean that the signal arrived earlier than its required time and the gate can be sized down
- The goal of gate sizing is to adjust the gate sizes such that the stack time of each gate is as low as possible without any gate having a negative stack i.e. timing violation
- The area minimum sizing problem has been a subject of research in logic synthesis for dozen of years
- Today in top down cell based design environment gate sizing has been much automated by the logic synthesis system

# Transistor Sizing for Leakage Power Reduction

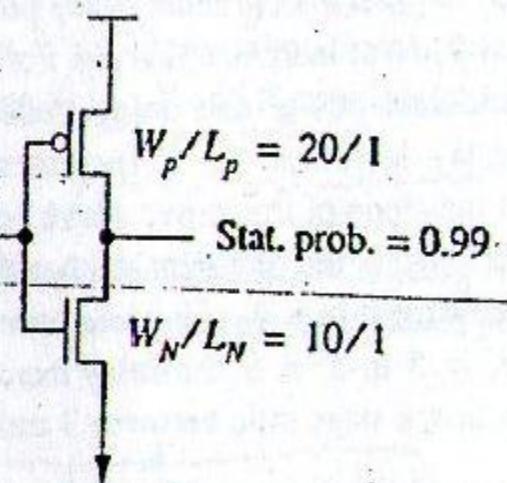
- An interesting problem occurs when the sizing goal is to reduce the leakage power of a circuit
- The leakage current of a transistor increases with decreasing threshold voltage and channel length
- In general, a lower threshold or shorter channel transistor can provide more saturation current and thus offers a faster transistor
- This presents a tradeoff between leakage power and delay
- The leakage power of a digital circuit depends on the logic state of a circuit
- Consider a simple two transistor inverter
- If the output of the inverter is at logic high, the leakage current of the inverter is determined by the N- transistor that is turned off

# Transistor Sizing for Leakage Power Reduction

- Conversely if the output is low, the leakage current depends on the P- transistor
- In order to suppress the leakage current, we can increase the threshold voltage or the channel length of the N-transistor
- However, by doing so we also increase the delay of the inverter because the N- transistor now offers less saturation current when it is turned ON
- If we are fortunate that the falling transition of the inverter is not the critical delay, we can use skewed transistor sizing method to reduce the leakage power without incurring any delay penalty

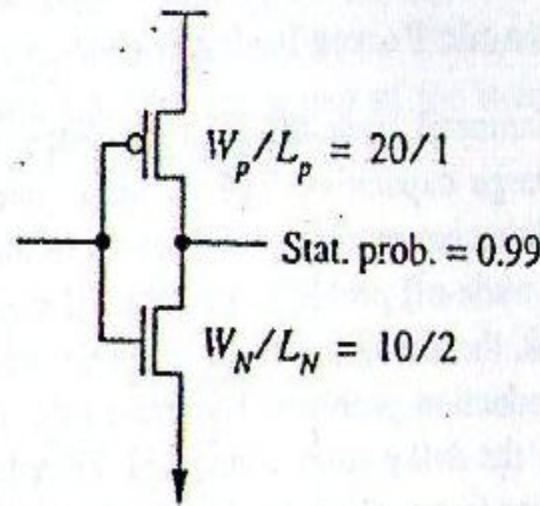
# Transistor Sizing for Leakage Power Reduction

- The transistor sizing concept is illustrated below



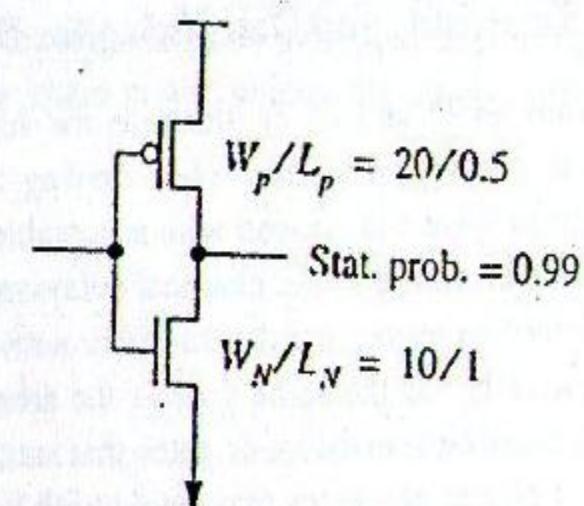
$$T_{rise} = 1 \quad T_{fall} \approx 1$$

$$P_{leakage} = 1$$



$$T_{rise} = 1 \quad T_{fall} = 2$$

$$P_{leakage} = 0.1$$



$$T_{rise} = 0.5 \quad T_{fall} = 1$$

$$P_{leakage} = 1$$

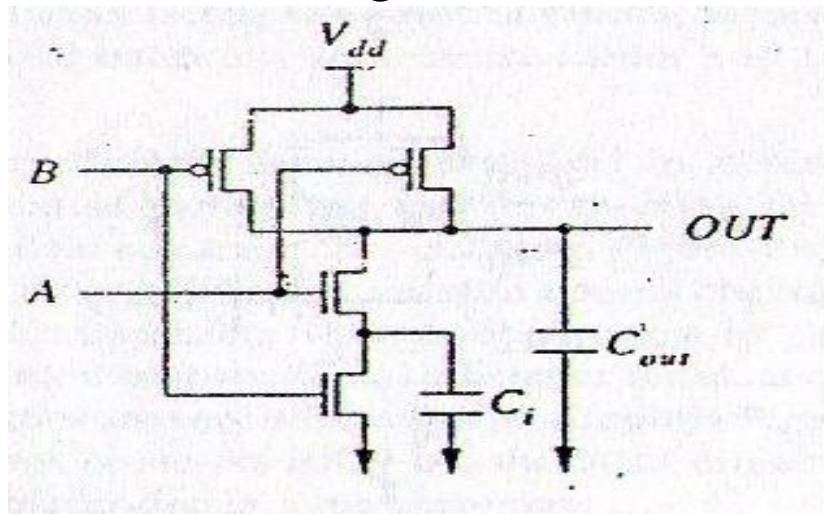
**Transistor sizing for leakage power reduction or speed increase**

# Equivalent Pin Ordering

- Most combinational digital logic gates have input pins that logically equivalent
- Eg: AND, OR, XOR
- Such gates are use frequently because they are natural to the human thinking process
- As for circuit implementation, the gates are robust and easy to design
- Logically equivalent pins may not have identical circuit characteristics, which means that the pins have different delay and power consumption
- Such property can be exploited for low power design

# Equivalent Pin Ordering

- Consider a CMOS NAND gate as shown below



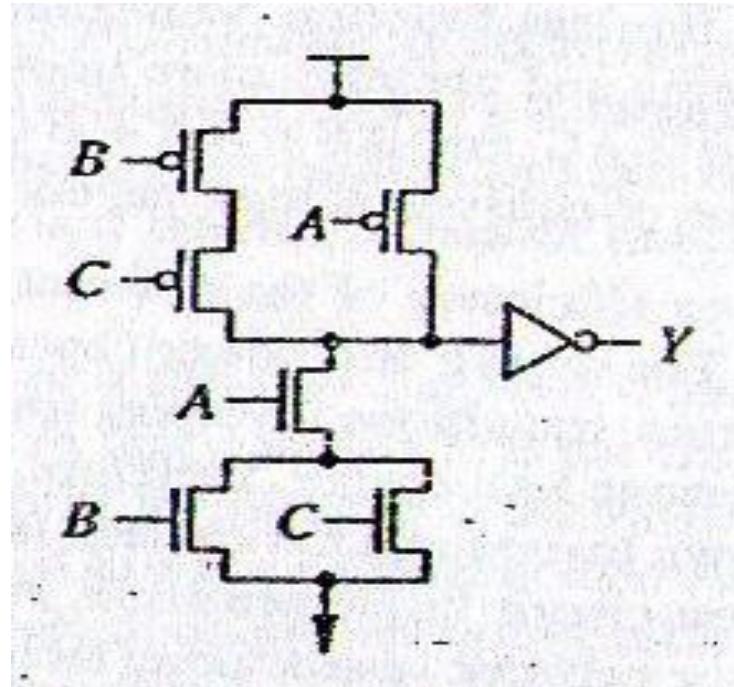
- We examine the condition when the input A is at high logic and the input B switches from logic low to high
- The difference in power dissipation varies depending on various factors such as capacitances and transistor sizes
- To conserve power the inputs should be connected such that transitions from input A to OUT occur more frequently than transitions from input B to OUT. This low power technique is known as *pin ordering*

# Network Restructuring and Reorganization

- The pin reordering technique is a special case of more general method called transistor restructuring
- In this method, we restructure the transistors of a combinational cell, based on signal probabilities, to achieve better power efficiency within the allowable timing constraints
- **Transistor Network Restructuring**
  - In CMOS logic design, there is a well known technique in which a Boolean function composed of AND and OR operators is directly mapped to a complex transistor network that implements the function
  - The mapping steps are as follows:
    1. Each variable in the Boolean function corresponds to a pair of P and N transistors
    2. For the N transistor network, an AND operator corresponds to a serial connection and an OR operator corresponds to a parallel connection

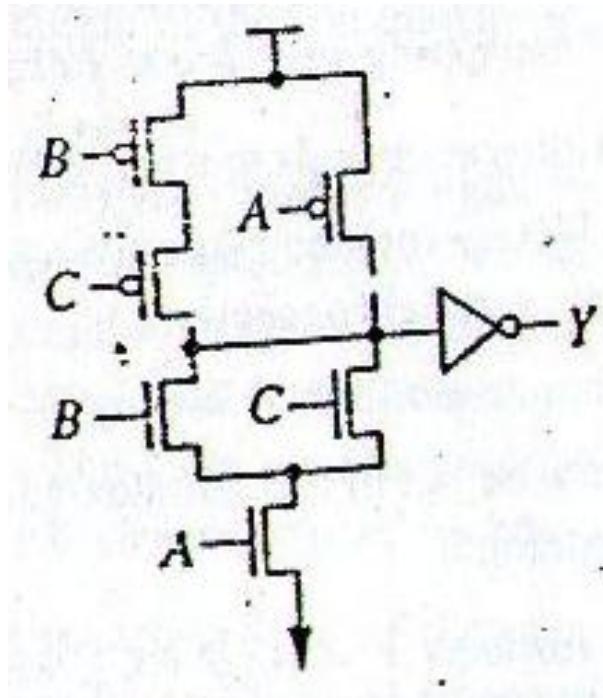
# Network Restructuring and Reorganization

3. For the P transistor network, the composition rule is inverted
4. An inverter is optionally added to the output of the complex gate to maintain the proper polarity or to ensure signal strength

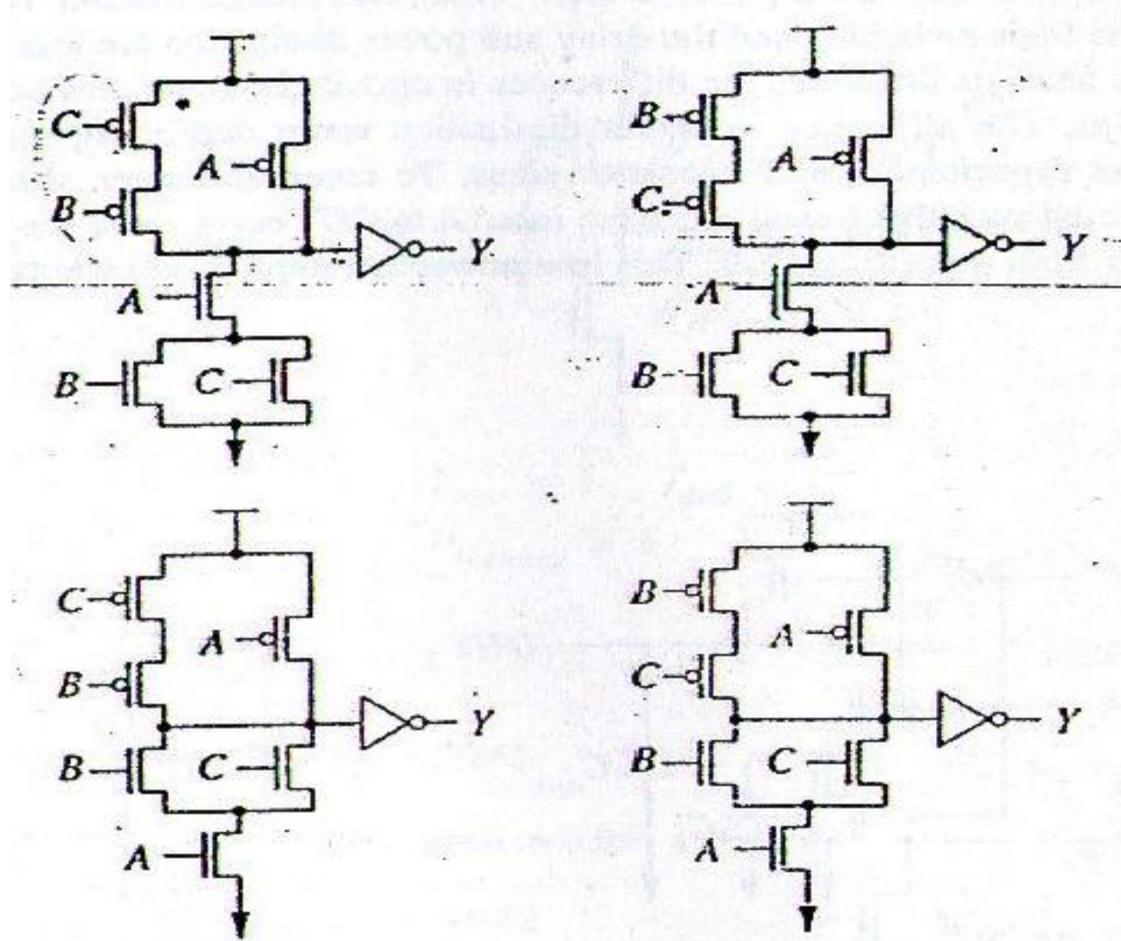


**Network composition of CMOS complex logic gate  $Y = A ( B + C )$**

# Network Restructuring and Reorganization



Alternative circuit implementation  
of  $Y = A(B + C)$

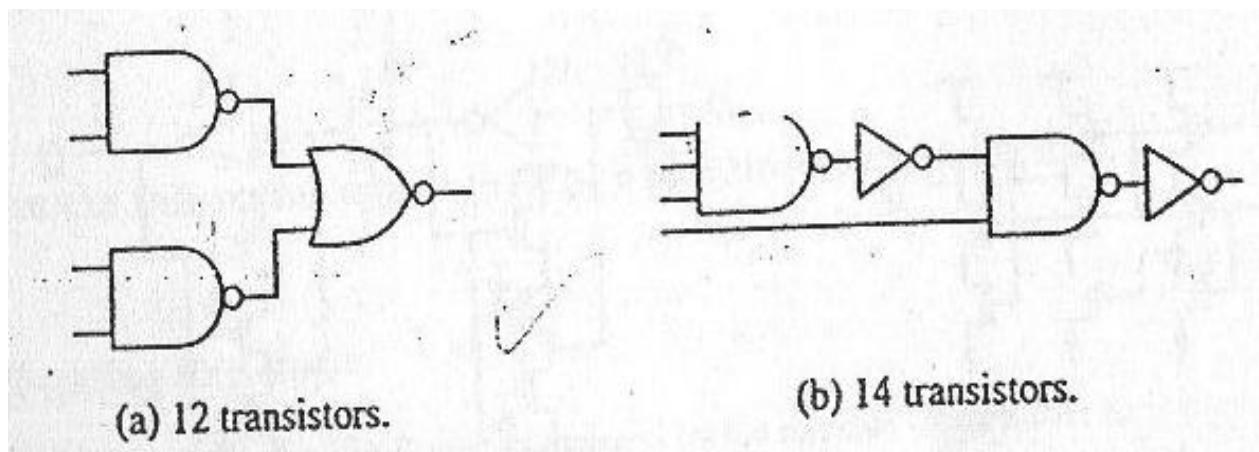


Four different circuit implementation of  $Y = A(B + C)$

- Power reduction up to 20% was reported using the transistor network restructuring technique

# Transistor Network Partitioning and Reorganization

- In this section, we look beyond a CMOS gate and consider a transistor network
- We study the problem of partitioning and reorganizing the network to explore the different power-area-delay trade off
- *Network reorganization* by definition is the task of composing different transistor networks that can implement the same functionality
- The figure below show two ways to implement a 4 input AND operation with a serial chain limit of three



# Flip Flops & Latches design

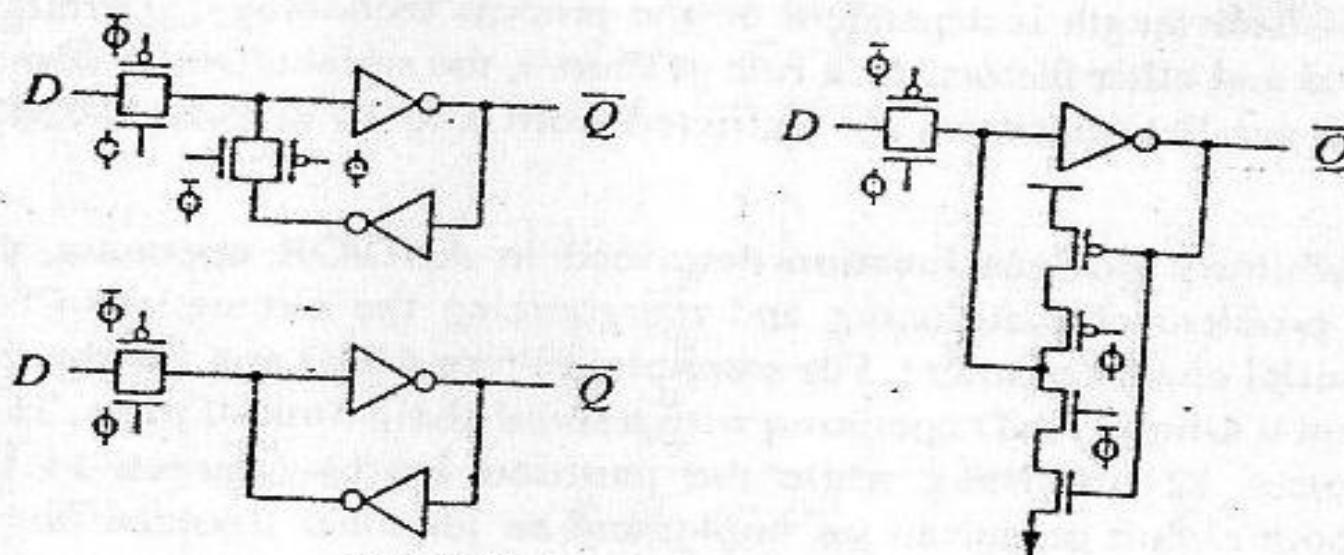
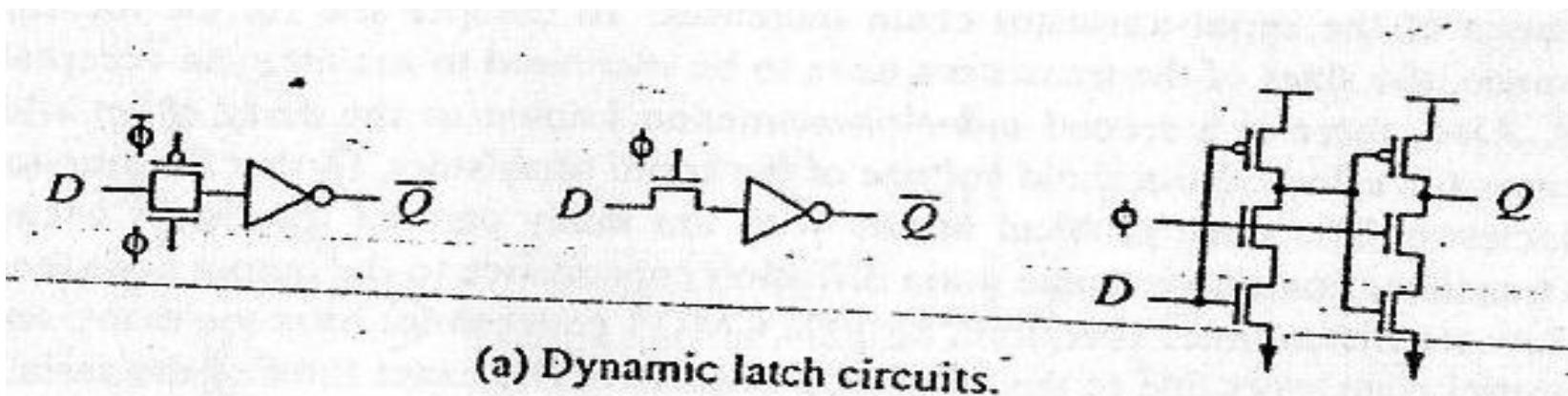
- Flip flops and latches are some of the most frequently used elements in digital VLSI
- In synchronous systems, they are the starting and ending points of signal delay paths, which decide the maximum speed of the system
- Typically, they consume more power because they are clocked at the system operating frequency
- Careful design of the flip flop and latch circuits is important to a low power VLSI design
- The energy dissipation of a flip flop can be divided into two components:
  1. Clock energy
  2. Data energy

# Flip Flops & Latches design

- The first component is the energy dissipated when the flip flop is clocked while the data of the flip flop is unchanged
- The second component is the additional energy required to write a different data value into flip flop
- In a typical flip flop the two energy components are comparable
- However, in most systems, the data rate of a flip flop is typically much lower than its clock rate
- This means that identical data values is being loaded with very high probability
- Thus, the power saving techniques for flip flops mostly concentrate on the clock energy reduction

# Flip Flop & Latch Circuits

- The figure below shows various implementation of CMOS latches

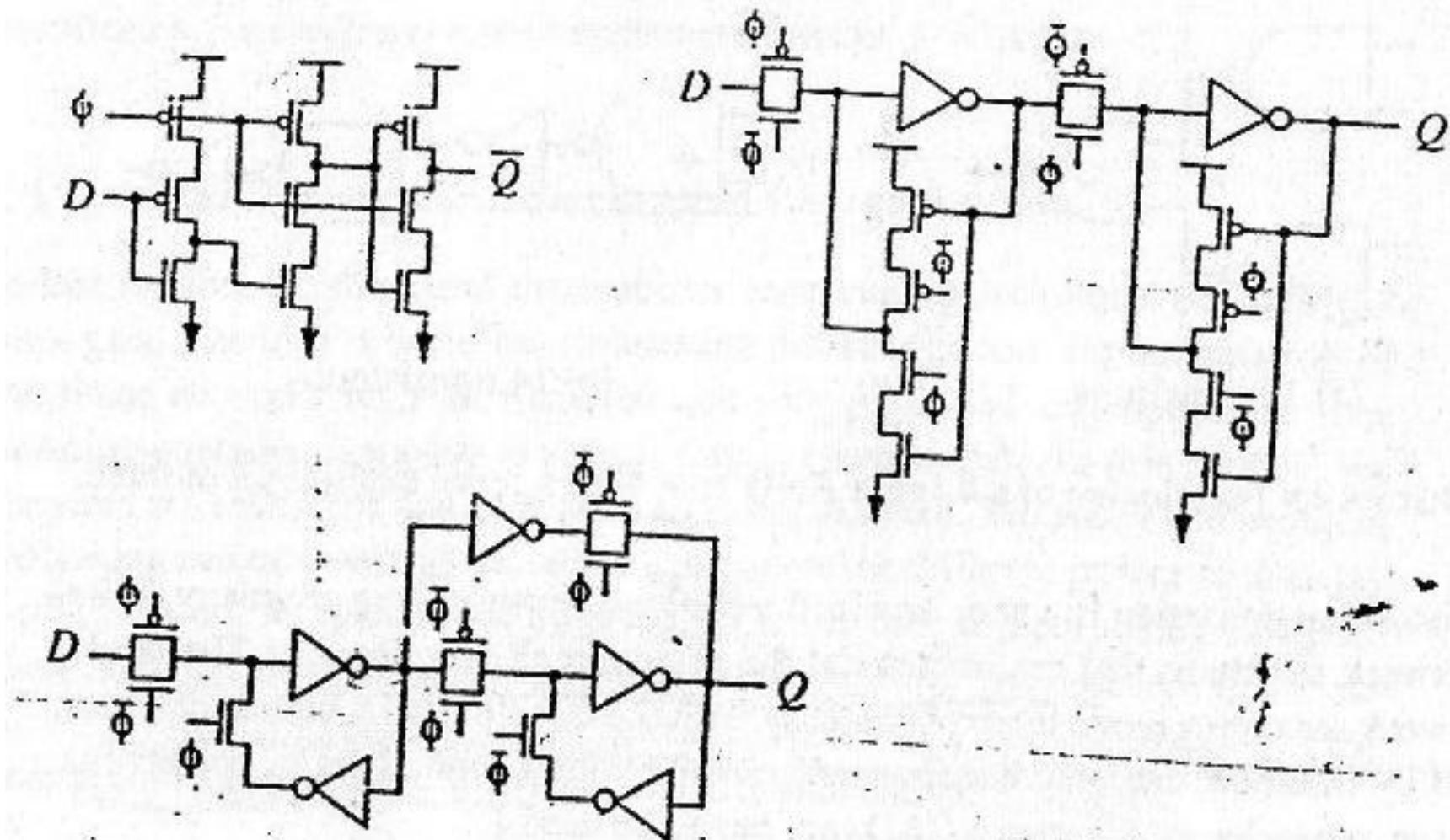


# Flip Flop & Latch Circuits

- The above circuits provides a different tradeoff among setup time, hold time, data to output and clock to output delay
- The use of NMOS pass transistors instead of transmission gates reduces the loading capacitance of the clock pin at the cost of reduced speed
- This eliminates the need for a two phase non overlapping clock on the system or a phase splitter inverter that consumes power inside the cell
- The circuit suffers from threshold voltage loss when logic 1 is propagated through the NMOS pass transistor
- The *single phase latch circuit* avoids the threshold voltage problem but relies on charge storage effect to retain its data value
- This cause some loss in the noise margin but the circuit has been successfully used in high performance processor design

# Flip Flop & Latch Circuits

- The figure below shows some circuit configuration of flip flops

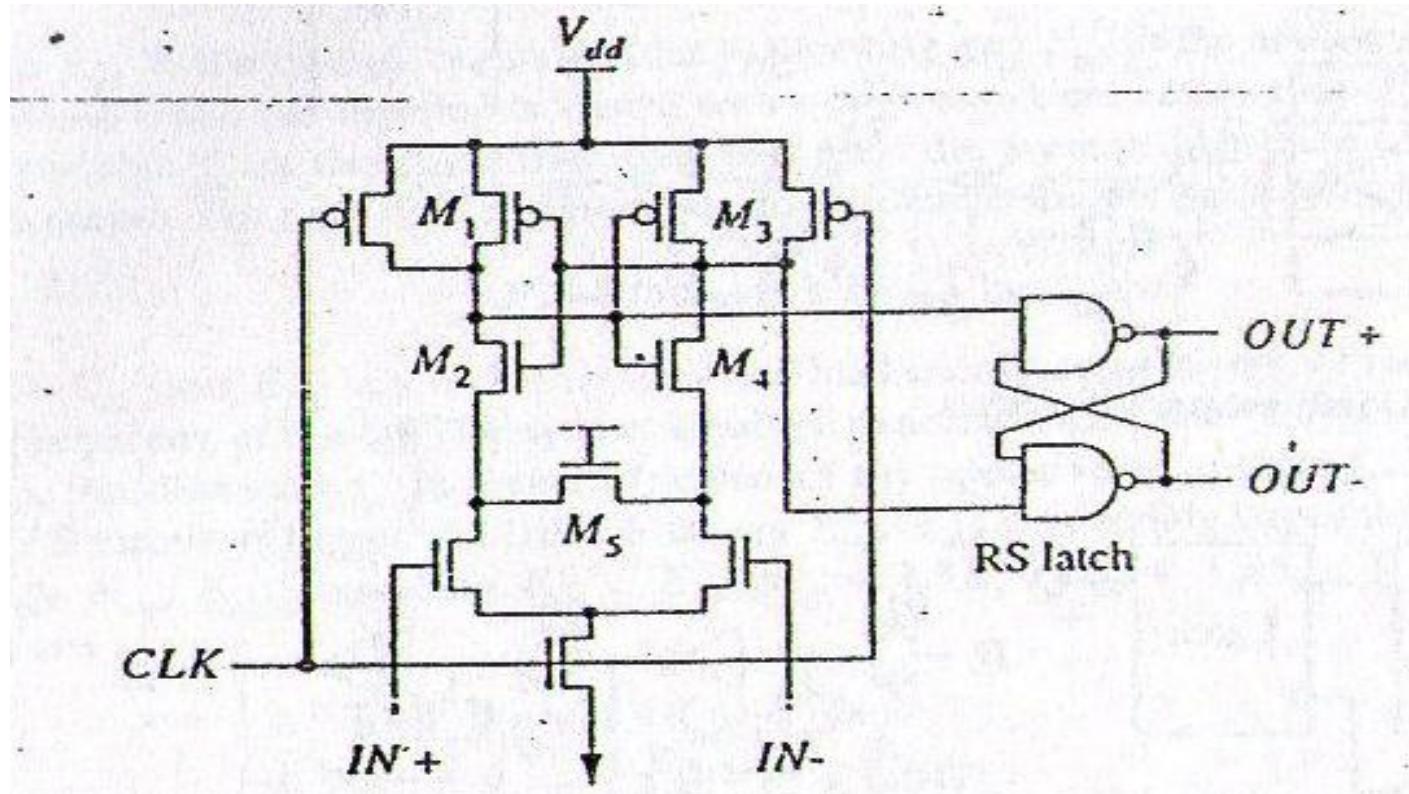


# Flip Flop & Latch Circuits

- A flip flop is typically constructed from two latches connected together
- The single phase dynamic flip flop at the top left is a cascaded version of two single phase latches
- It is suitable for some low power applications because it does not require internal phase splitting inverter for the clock pin
- The circuit at the bottom was reported to achieve lower power at the same speed with more transistors, compared to a standard flip flop design on the top right

# Flip Flop & Latch Circuits

- One circuit that uses an exotic differential signaling method to increase speed at the expense of area and power is shown below



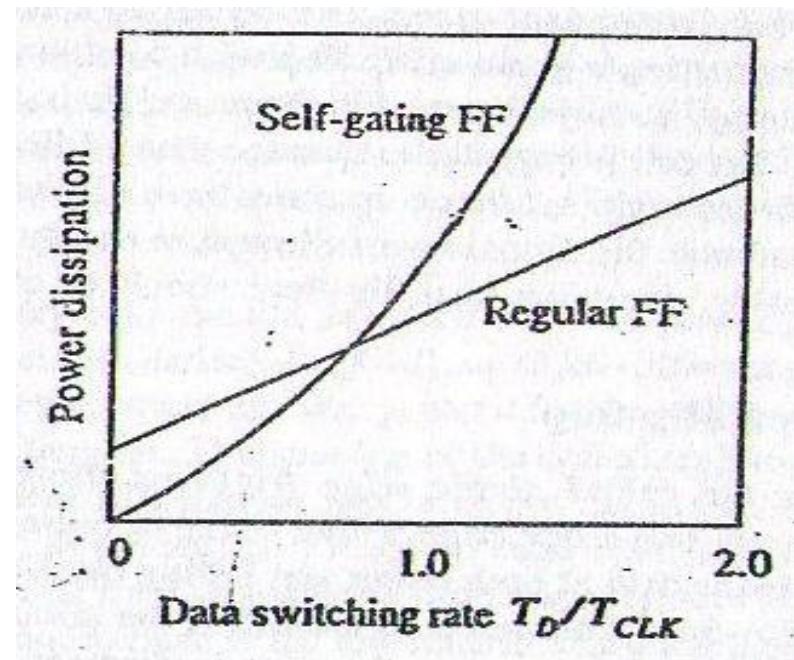
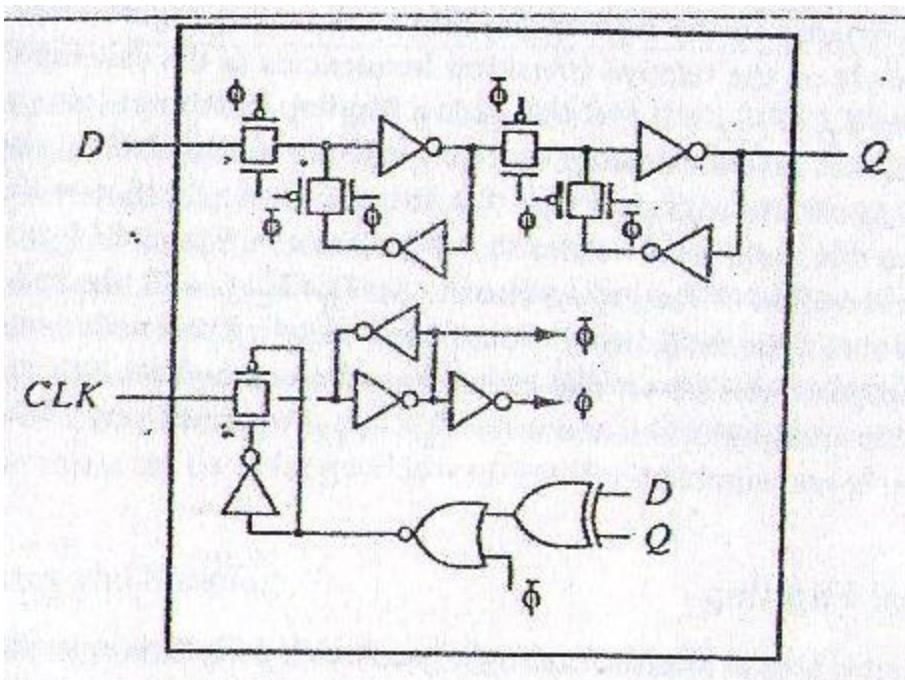
Differential input latch for high speed low voltage application

# Self-gating Flip-flop

- Part of the clock energy of a flip flop is consumed by the internal clock buffer to control the transmission gates
- If the input of the flip flop is identical to its output, the switching of its clock signal can be suppressed to conserve power
- This is similar to clock gating techniques
- The difference is that the gating function is derived within the flip flop without any external control signal
- Power is saved by holding the internal clock signals of the flip flop when allowed
- The external clock signal of the flip flop still switches

# Self-gating Flip-flop

- An example of self gating flip flop is shown below

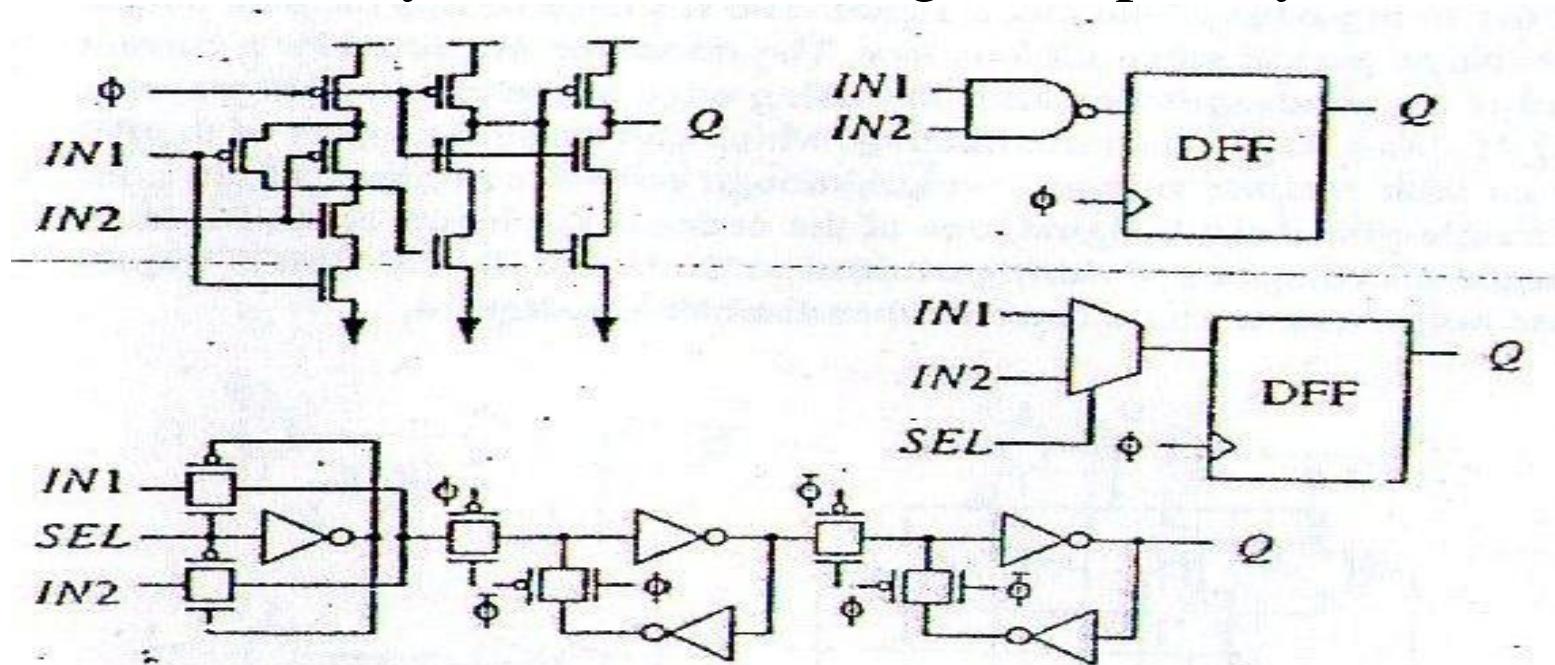


Flip flop with self clock gating

Power dissipation of self gating flip flop and regular flip flop

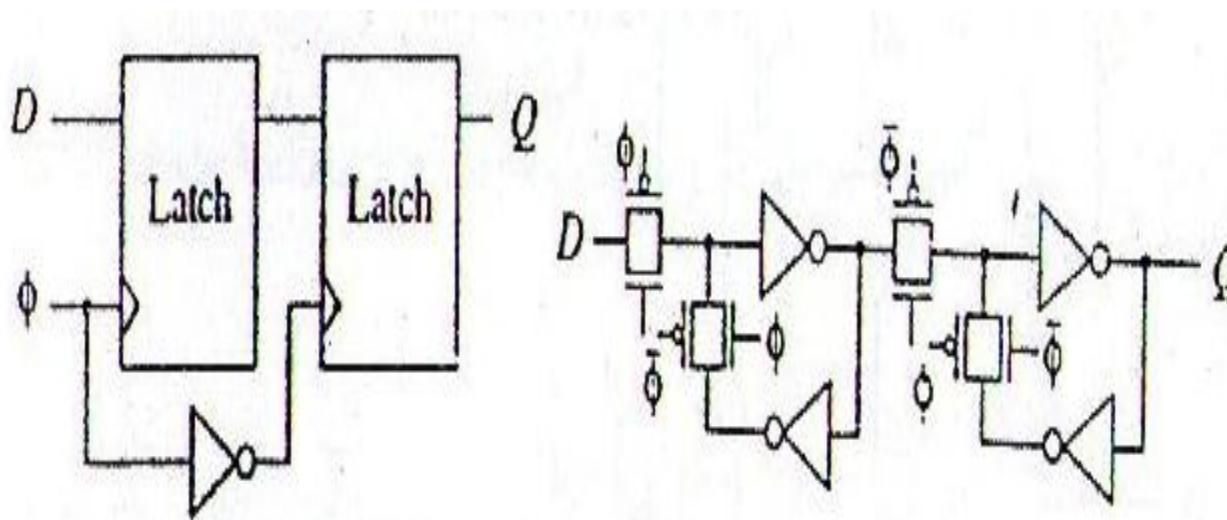
# Combinational flip flop

- One way to reduce circuit size is to associate logic gates with a flip flop to produce a combinational flip flop
- Combinational flip flops are efficient because they are able to eliminate or share redundant gates
- In terms of area, power and delay, combinational flip flops are desirable but they increase the design complexity



# Double Edge Triggered flip flop

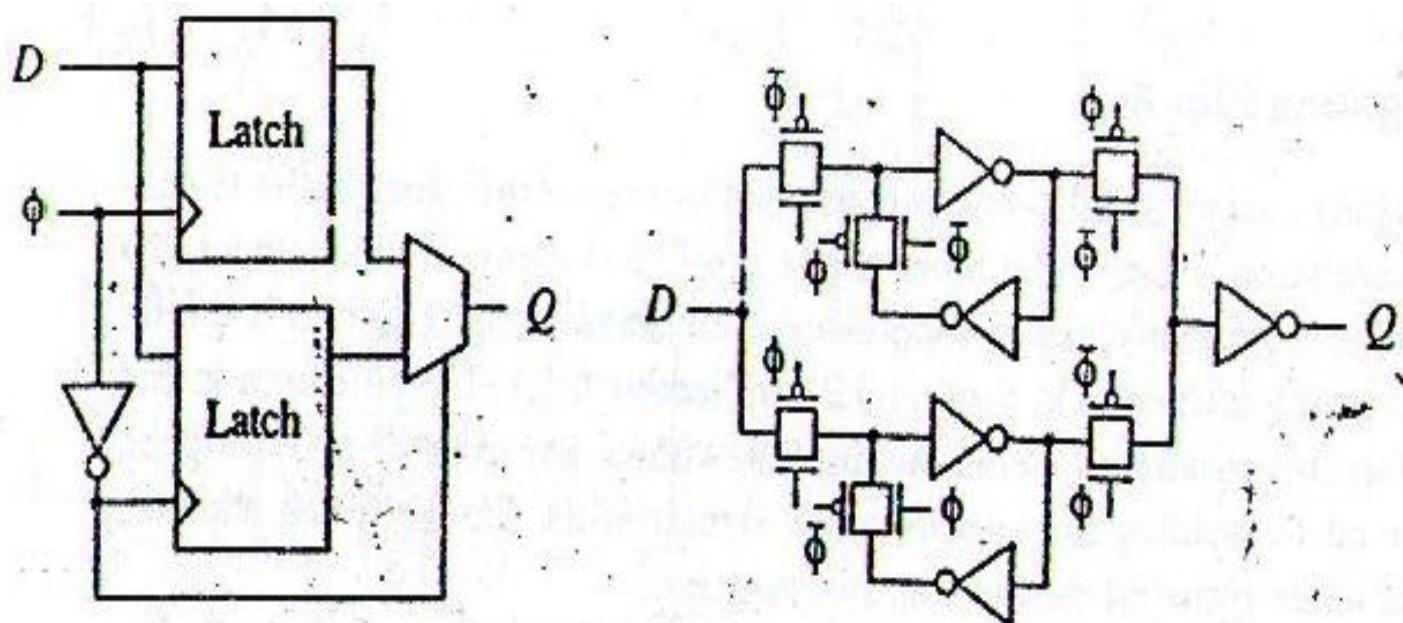
- Most flip flops in use today are called single edge triggered flip flop (SETFF) because the data are loaded at only one clock edge, either rising or falling
- A flip flop can also be designed so that it loads data at both rising and falling clock edges



(a) Single edge triggered flip-flop.

# Double Edge Triggered flip flop

- Compared to SETFF, the double edge triggered flip flop (DETFF) requires slightly more transistors to implement
- The flip flop retains its data when the clock signal is not toggling



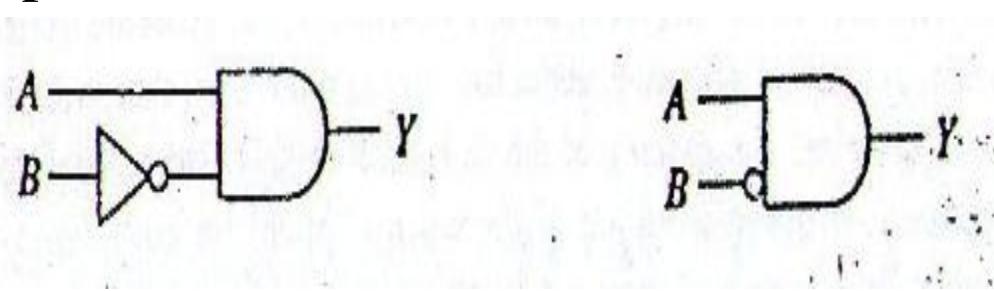
(b) Double edge triggered flip-flop.

# Low Power Digital Cell Library

- Over the years, the major VLSI design focus has shifted from masks, to transistors, to gates and to register transfer level
- Undoubtedly, the quality of gate level circuit synthesized depends on the quality of the cell library
- Cell Sizes and Spacing
  - In the top-down cell based design methodology, the tradeoff among power, area and delay is performed by selecting the appropriate sizes of the cells
  - Therefore, the important attribute that constitute a good low power cell library is the availability of wide ranges of cell sizes for commonly used gates
  - Further, the library cell count can be reduced without too much compromise in quality is to have more size selections for gates that are commonly used than those are less likely used

# Low Power Digital Cell Library

- Varieties of Boolean Functions
  - The lack of varieties of Boolean functions in a cell library can result in inferior circuits to be generated
  - For example if the Boolean function  $Y = A \bar{B}$  were to be implemented and the inverted input cells are not available, the logic synthesis system has to use an INVERTER and an AND gate to implement the function



Inverted input cells for low power cell library

- The two cell implementation is less power efficient compared to the single cell implementation because of the external routing capacitance at the output of the inverter

# **UNIT-III**

## Low Power Design-Logic level

# Logic Design

- *Logic design* was once the primary abstraction level where automatic design synthesis begins
- The most prevalent theme in logic level power optimization techniques is the reduction of *switching activities*
- Switching activities directly contribute to the *charging and discharging capacitance* and the *short circuit power*
- Some switching activities are the result of *unspecified or undefined behavior* of a logic system that are not related to its power operation
- Such *stray switching activities* should be eliminated or reduced if possible
- However, suppressing unnecessary activities usually requires *additional hardware logic* that increases the area and consumes power
- The challenge is to justify the low power techniques via *intelligent analysis*

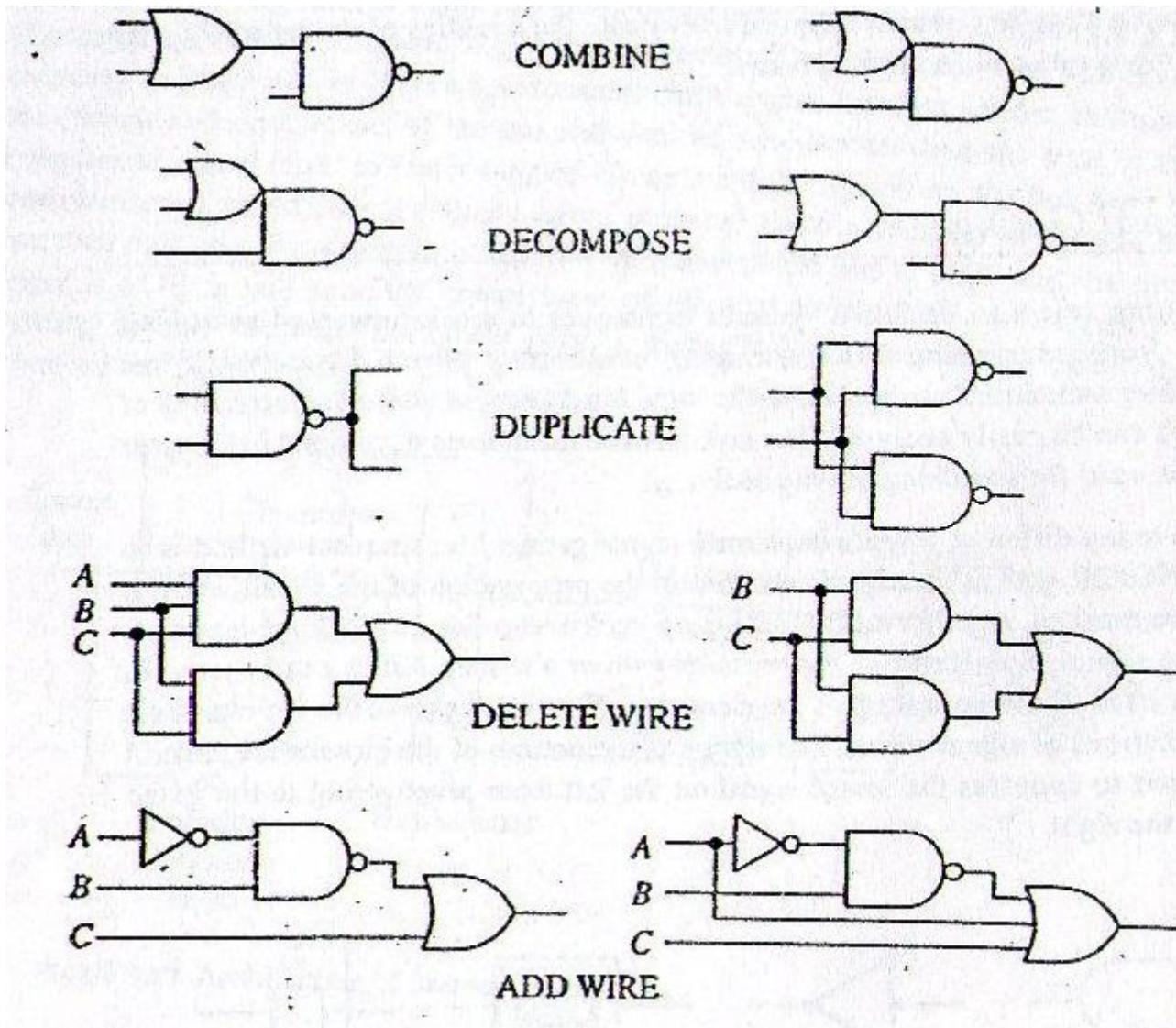
# Gate Reorganization

- The reorganization idea is not limited to transistor networks only since the same problem exists in gate level networks
- Network reorganization is applied to the gate level network to produce logically equivalent networks with different qualities for power, area and delay
- Technology mapping
  - Original network is expressed in a generic form such as two input NAND gates only
- The reorganized network hopefully has better power efficiency than the original network
- The complexity of the gate reorganization problem limits manual solution to small circuits only
- Most gate reorganization tasks are performed by automated software in the logic synthesis system

# Local Restructuring

- Gate reorganization is an operation to transform one logic circuit to another that is functionally equivalent
- Logic restructuring techniques use local restructuring rules to transform one network to another
- Some basic transformation operators are:
  1. Combine several gates into a single gate
  2. Decompose a single gate into several gates
  3. Duplicate a gate and redistribute its output connections
  4. Delete a wire
  5. Add a wire
  6. Eliminate unconnected gates

# Local transformation operators for gate reorganization



# Local Restructuring

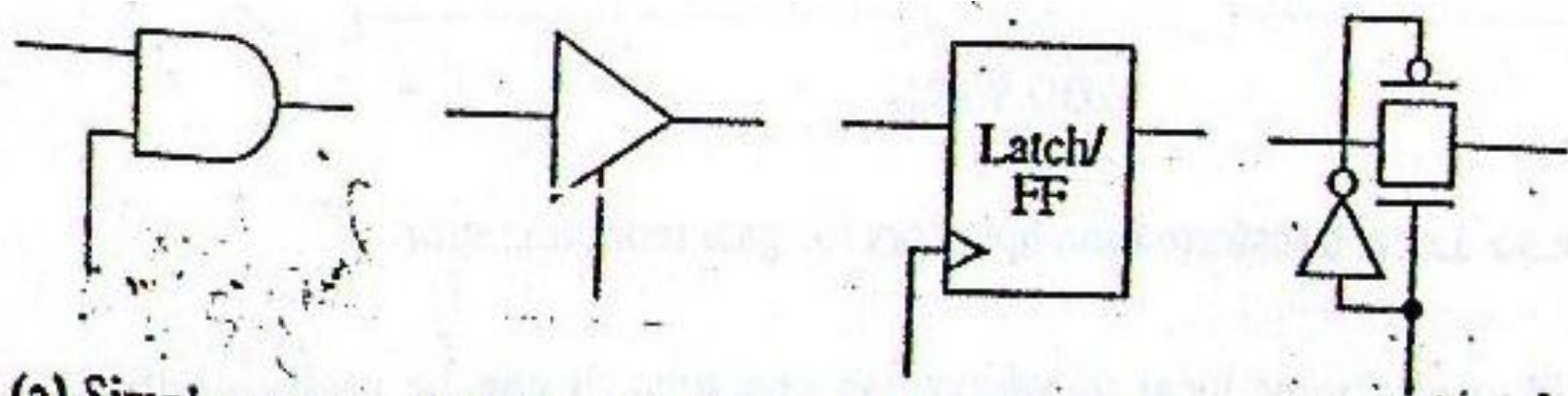
- **COMBINE** operator can be used to hide high frequency nodes inside a cell so that the node capacitance is not being switched
- **DECOMPOSE** and **DUPLICATE** operators help to separate the critical path from the non critical ones so that the latter can be sized down
- **DELETE WIRE** operator reduces the circuit size
- **ADD WIRE** operator helps to provide an intermediate circuit that may eventually lead to a better one

# Signal Gating

- Signal gating refers to a class of general techniques to mask unwanted switching activities from propagating forward, causing unnecessary power dissipation
- The probabilistic techniques are often used for switching activity analysis
- The simplest method to implement signal gating is to put an AND/OR gate at the signal path to stop the propagation of the signal when it needs to be masked
- Another method is to use a latch or flip flop to block the propagation of the signal
- Sometimes a transmission gate or tristate buffer can be used in place of a latch if charge leakage is not a concern

# Signal Gating

- The various logic implementation of signal gating is shown below



(a) Simple gate. (b) Tristate buffer. (c) Latch/FF. (d) Transmission gate.

- The signals at the bottom of the circuits are control signals used to suppress the source signal on the left from propagating to the gated signal on the right

# Logic Encoding

- The logic designer of a digital circuit often has the freedom of choosing a different encoding scheme as long as the functional specification of the circuit is met
- For Eg. An 8 bit counter can be implemented using the binary counting sequence or the gray code sequence
- Different encoding implementation often lead to different power, area and delay tradeoff
- The encoding techniques require the knowledge of signal statistics in order to make design decisions
- The next slide discusses some techniques for using different logic encoding to achieve low power consumption

# Binary versus Gray Code Counting

- Consider two n-bit counters implemented with Binary and Gray code counting sequences
- The counting sequences of the two counters are shown below

Binary code		Gray code	
Sequence	No. toggles	Sequence	No. toggles
000	3	000	1
001	1	001	1
010	2	011	1
011	1	010	1
100	3	110	1
101	1	111	1
110	2	101	1
111	1	100	1

# Binary versus Gray Code Counting

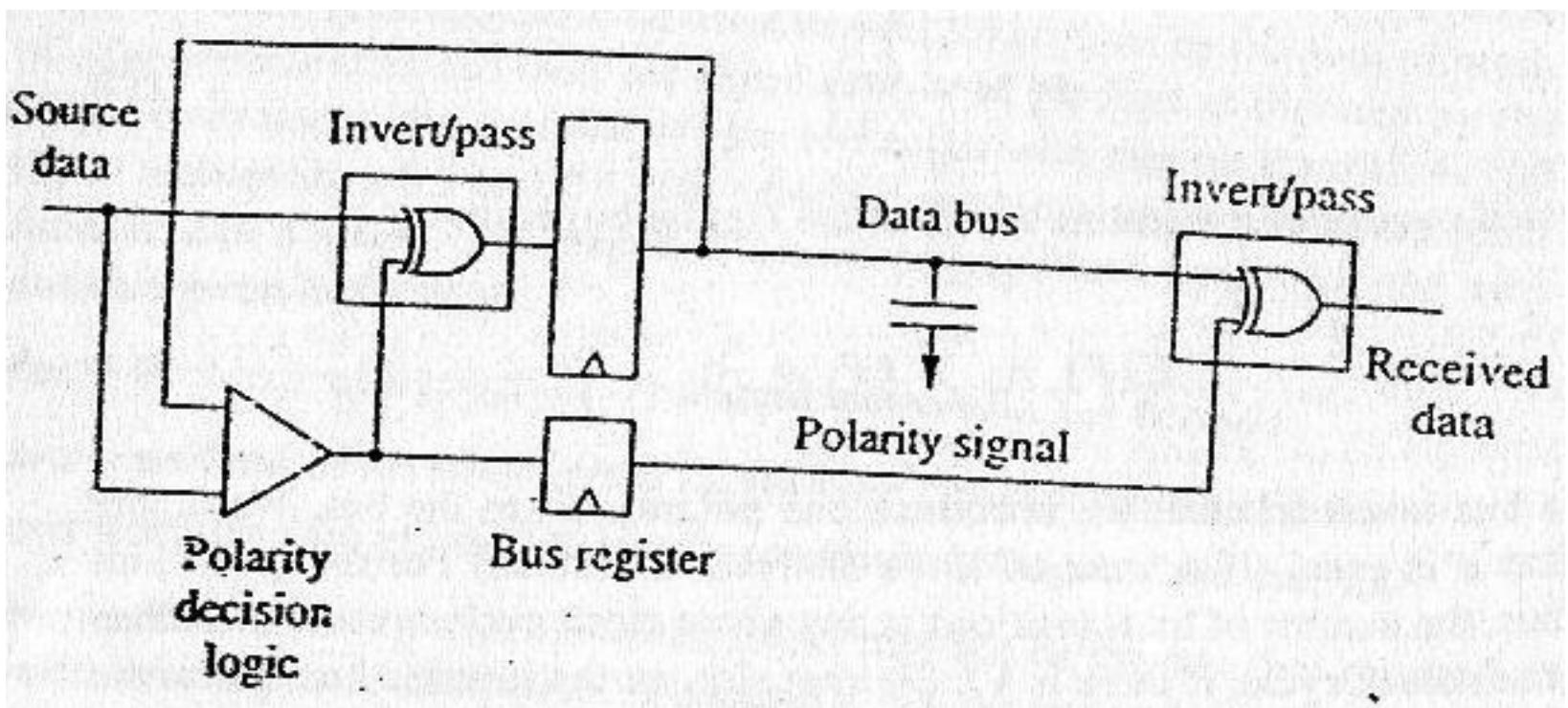
- Toggle activities of binary versus gray code counter

No. bits	No. of toggles		$B_n / G_n$
	Binary $B_n = 2(2^n - 1)$	Gray $G_n = 2^n$	
1	2	2	1
2	6	4	1.5
3	14	8	1.75
4	30	16	1.88
5	62	32	1.94
6	126	64	1.99
$\infty$	-	-	2.00

- When  $n$  is large, the Binary counter has twice as many transitions as the Gray counter
- Since the power dissipation is related to toggle activities, a Gray counter is generally more power efficient than a Binary counter

# Bus Invert Encoding

- Bus invert encoding is a low power encoding technique that is suitable for a set of parallel synchronous signals e.g.: off-chip busses
- The architecture of bus invert encoding is illustrated below



# Bus Invert Encoding

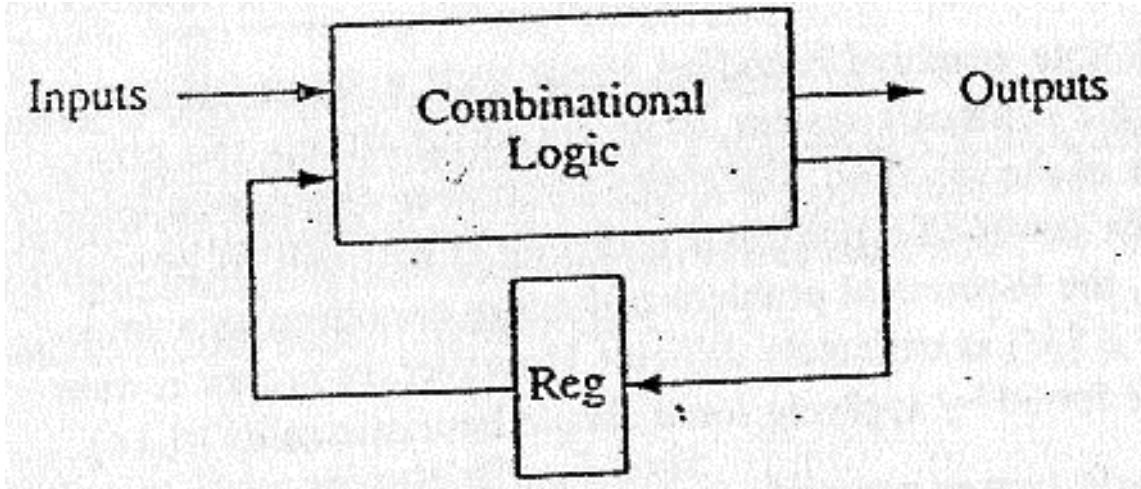
- At each clock cycle, the data sender examines the current and next values of the bus and decides whether sending the true or the compliment signal leads to fewer toggles
- Since the data signals on the bus may be complemented, an additional polarity signals is sent to the bus receiver to decode the bus data properly
- The assertion of the polarity signal tells the receiver to invert the received bus signals

Num. bits	Regular bus $E[P]$	Invert bus $E[Q]$	Invert / Regular $E[Q] / E[P]$
2	1	0.75	0.75
4	2	1.56	0.781
8	4	3.27	0.817
16	8	6.83	0.854
32	16	14.19	0.886
64	32	29.27	0.915
128	64	59.96	0.937
256	128	122.1	0.954
∞	-	-	1.00

**Efficiency of bus invert encoding under uniform random signal**

# State Machine Encoding

- A state machine is an abstract computation model that can be readily implemented using Boolean logic and flip flops as shown below



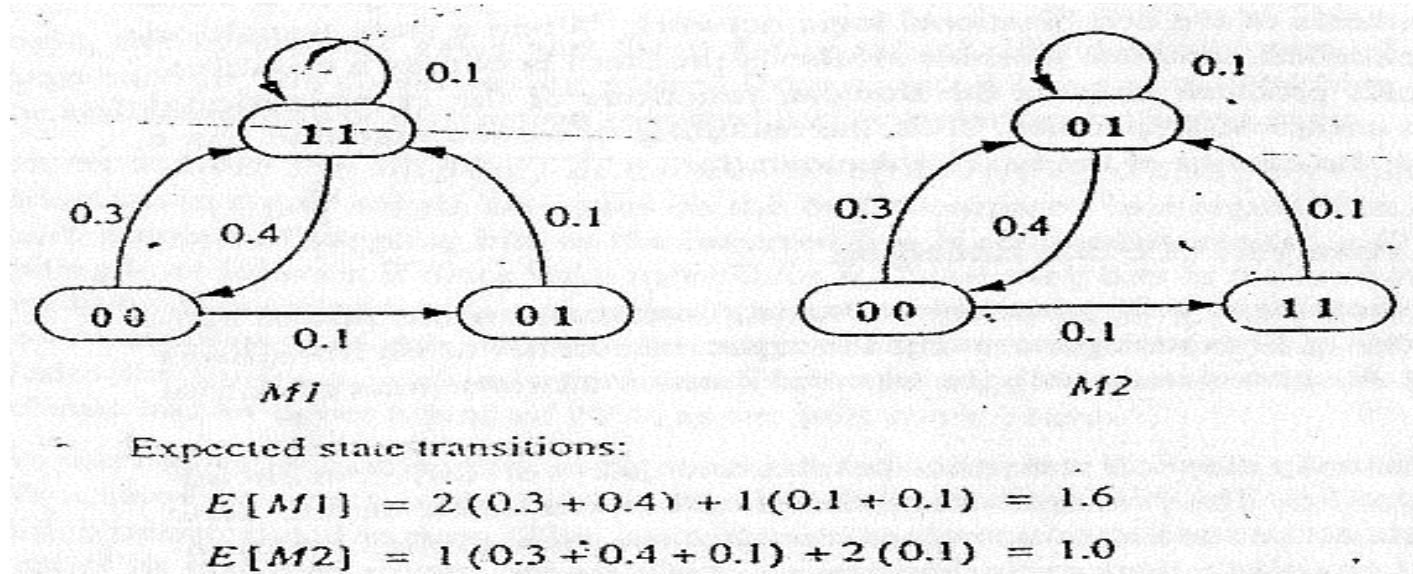
- In today's logic synthesis environment, a state transition graph is specified by the designer and the synthesis system will produce a gate level circuit based on the machines specification
- The state transition graph is a functional description of a machine specifying the inputs and outputs of the machine under a particular state and its transition to the next state

# State Machine Encoding

- The very first step of a state machine synthesis process is to allocate the state register and assign binary codes to represent the symbolic states. This process is called the encoding of a state machine
- The encoding of a state machine is one of the most important factors that determine the quality (area, power, speed etc) of the gate level circuit
- **Transition Analysis of State Encoding**
  - The key parameter to the power efficiency of state encoding is the expected number of bit transitions  $E[M]$  in the state register
  - Another parameter is the expected number of transitions of output signals
- In general machines with lower  $E[M]$  are more power efficient because
  - Fewer transitions of the state register lead to low power dissipation and
  - Fewer transitions are propagated into the combinational logic of the machine

# State Machine Encoding

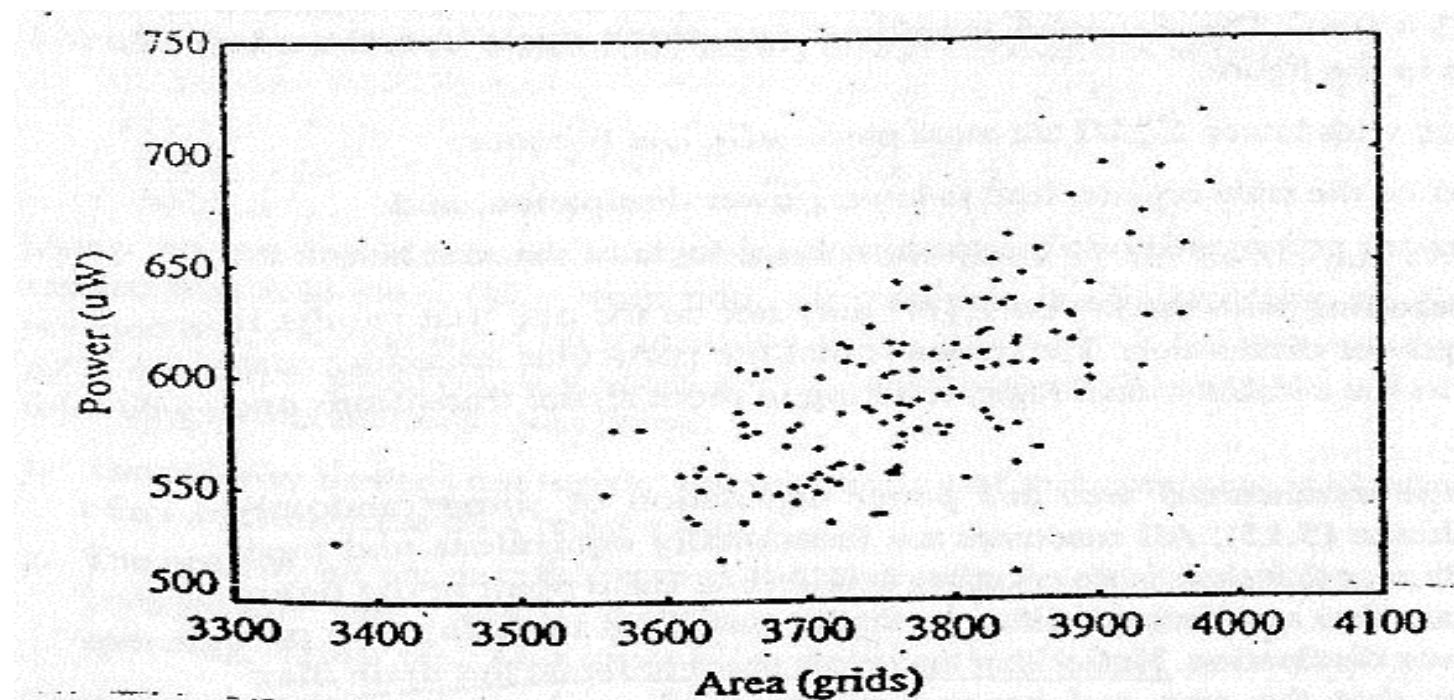
- Consider two functionality identical state machines M1 and M2 with different encoding as shown below



- The binary codes in the state bubbles indicate the state encoding
- The labels at the state transition edges represent the probabilities that transition will occur at any given clock cycle
- The sum of all edge probabilities equals to unity
- The expected number of state bit transitions  $E[M]$  is given by the sum of products of edge probabilities and their associated number of bit flips as dictated by the encoding

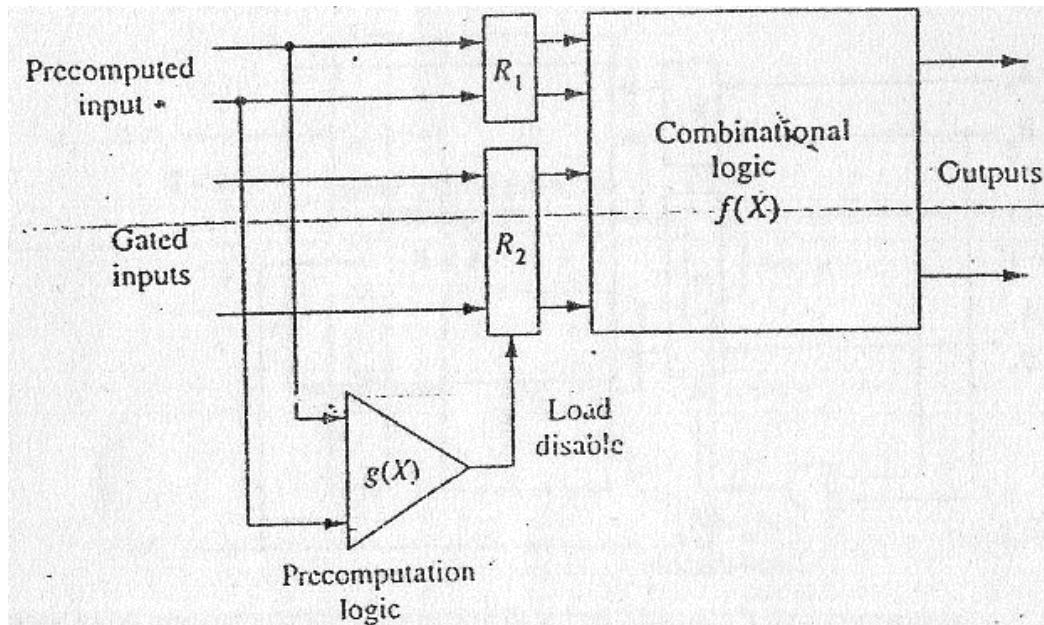
# State Machine Encoding

- However, a state encoding with the lowest  $E[M]$  may not be the one that results in the lowest overall power dissipation
- The reason is that the particular encoding may require more gates in the combinational logic, resulting in more signal transitions and power
- The synthesized area and power dissipation of some randomly encoded state machines is shown below



# Precomputation Logic

- Precomputation logic optimization is a method to trade area for power in a synchronous digital circuit
- The principle of precomputation logic is to identify logical conditions at some inputs to a combinational logic that is invariant to the output
- Since those input values do not affect the output, the input transitions can be disabled to reduce switching activities
- One variant of precomputation logic is shown below

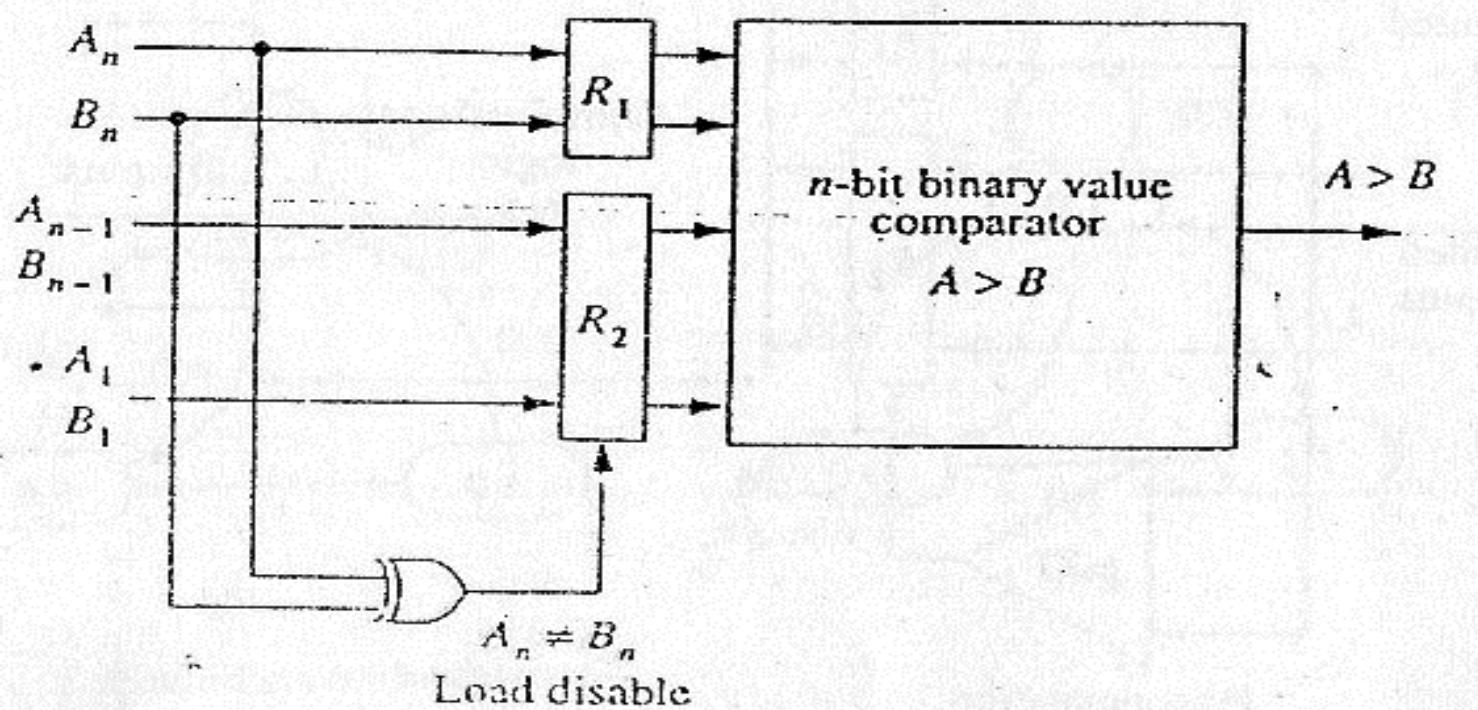


# Precomputation Logic

- Let R1 and R2 are registers with a common clock feeding a combinational logic circuit with a known Boolean function  $f(x)$
- Due to the nature of the function  $f(x)$ , there may be some conditions under which the output of  $f(x)$  is independent of the logic value of R2
- Under such conditions, we can disable the register loading of R2 to avoid causing unnecessary switching activities, thus conserving power
- The Boolean function  $f(x)$  is correctly computed because it receives all required values from R1
- To generate the load disable signal to R2, a precomputation Boolean function  $g(x)$  is required to detect the condition at which  $f(x)$  is independent of R2
- $g(x)$  depends on the input signals of R1 only because the load disable condition is independent of R2, otherwise  $f(x)$  will depend on the inputs of R2 when the load disable signal is active

# Binary comparator function using Precomputation Logic

- Assuming uncorrelated input bits with uniform random probabilities where every bit has an equal probability of 0 or 1
- There is 50% probability that  $A_n \oplus B_n = 1$  and the register R2 is disabled in 50% of the clock cycles
- Therefore, with only one additional 2 input XOR gate, we have reduced the signal switching activities of the  $2n-2$  least significant bits at R2 to half of its original expected switching frequency



# Binary comparator function using Precomputation Logic

- Also, when the load disable signal is asserted, the combinational logic of the comparator has fewer switching activities because the outputs of R2 are not switched
- The extra power required to compute  $A_n \oplus B_n$  is negligible compared to the power saving even for moderate size of n

## Alternate Precomputation Architectures

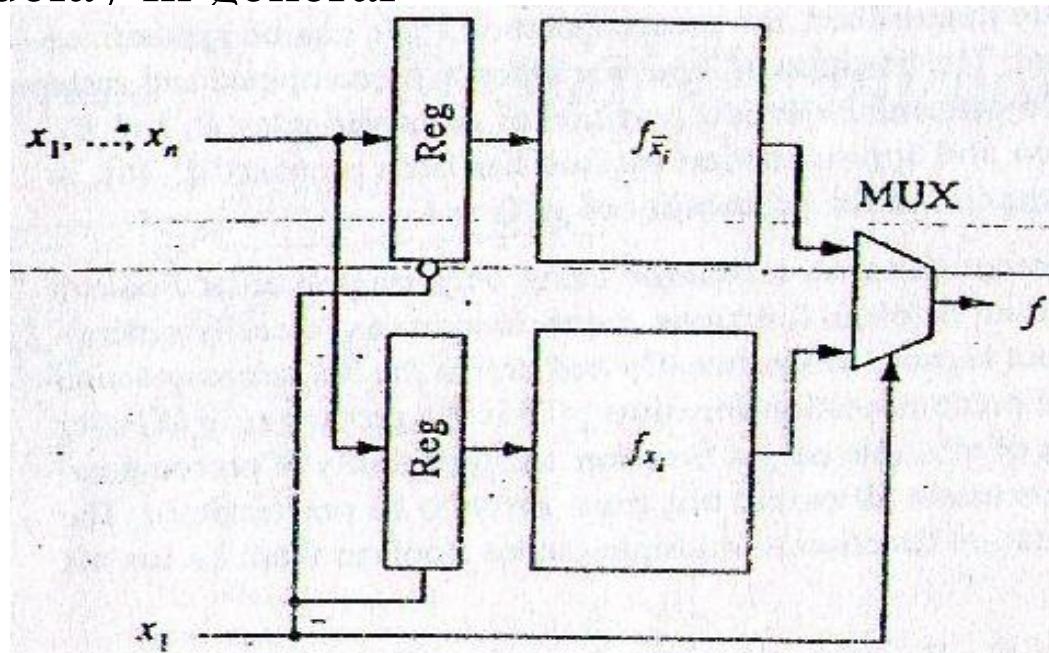
- The precomputation scheme based on Shannon's decomposition states that a Boolean function  $f(x_1, \dots, x_n)$  can be decomposed with respect to the variable  $x_i$  as follows

$$f(x_1, \dots, x_n) = x_i f_{x_i} + \bar{x}_i f_{\bar{x}_i}$$

- The equation allows us to use  $x_i$  as the load disable signal
- When  $x_i = 0$  ( $x_i = 1$ ) the inputs to the logic block  $f_{x_i}$  can be disabled

# Alternate Precomputation Architectures

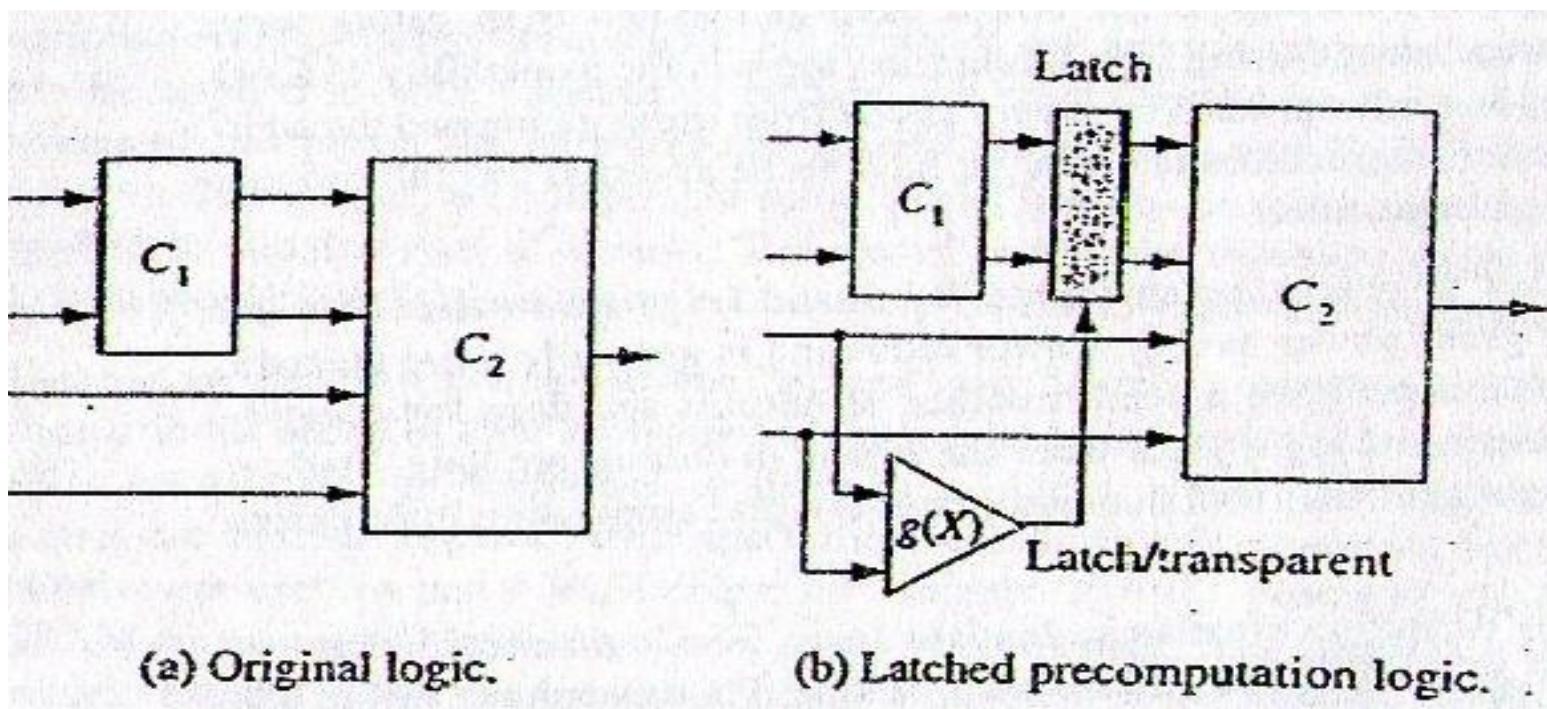
- The multiplexer selects the output of the combinational logic block that is active
- This means that only one combinational logic block is activated at any clock cycle
- Power saving is achieved if each of the two decomposed logic blocks consumes less power than a direct implementation
- However, the precomputation architecture consumes more area and delay in general



A precomputation architecture based on Shannon's decomposition

# Latch based Precomputation Architecture

- The latch based precomputation architecture is shown below
- This architecture is also called *guarded evaluation* because some inputs to the logic block  $C_2$  are isolated when the signals are not required, to avoid unnecessary transition
- Transmission gates may be used in place of the latches if the charge storage and noise immunity conditions permit



Latch based precomputation architecture

# Design Issues in Precomputation Logic Technique

- The basic design steps with precomputation logic are as follows:
  1. Select precomputation architecture
  2. Determine the precomputed inputs  $R_1$  and gated inputs  $R_2$  given the function  $f(x)$
  3. With  $R_1$  and  $R_2$  selected, find a precomputation *logic function*  $g(x)$ .
  4. Note that  $g(x)$  is not unique and the choice greatly affects the *power efficiency*. The function  $g(x)$  may also fail to exist for poor choices of  $R_1$  and  $R_2$
  5. Evaluate the probability of precomputation condition and the potential power savings. Make sure that the final circuit is not overwhelmed by the *additional logic circuitry* and power consumption required to compute  $g(x)$
  6. After  $R_1$ ,  $R_2$  and  $g(x)$  are determined, the precomputation logic can be synthesized using a *logic synthesis tool*

## **UNIT-III**

**Low power Architecture & Systems**

# Architecture & Systems

- The most common abstraction level for manual design is at the so called *register level transfer*
- At this level, all synchronous registers, latches and the combinational logic between the sequential elements are described in a hardware description language such as Verilog or VHDL
- The description is then transferred to logic gate implementation
- Hence, the designer can implement more complex design
- More than *register level transfer*, the analysis and synthesis of digital circuits are still being pushed toward even higher level of abstraction for manual implementation E.g. *macro block level* specification
- The *register level transfer* and *macro block level* will be discussed in architecture and systems
- Low power architecture techniques are important because the design analysis, verification and automated synthesis begin at this level
- The choice of clocking strategy, parallelism, pipelining, component organization, etc are the issues to be considered at this level

# Power & performance management

- Power management
  - It refers to a general class of low power techniques that carefully manage the performance and throughput of a system based on its computation needs to achieve low power goals
- Microprocessor sleep modes
  - The most important and successful use of power management is to deactivate some functional units when no computation is required
  - The tradeoff is to justify the additional hardware and design complexity in managing the various functional units
  - At architecture design, this is attractive technique because little hardware and design complexity is needed to achieve substantial power saving

# Power & performance management

- An example of power management schemes can be seen with Motorola's PowerPC 603 processor design
  - The CPU has three primary power save modes called DOZE, Nap and SLEEP controlled by software
  - DOZE mode – The most functional units of the processor are stopped except the on chip cache memory to maintain cache coherency
  - NAP mode – The cache is also turned off to conserve power and the processor wakes up after a fixed amount of time or upon external interrupt
  - SLEEP mode – The entire processor clock may be halted and only an external reset or interrupt can resume its operation

# Power & performance management

- The aim is to put more functional units in idle when the processor goes into a deeper sleep mode
- However, it requires more latency to resume computation from a deeper sleep mode
- This allows the software to choose the proper power saver mode based on the computation requirements of the processor
- As the computational requirements becomes less, the processor can be put into deeper sleep modes without any perceivable loss in performance
- The *dynamic power management* mode refers to the application of clock gating at the macro module level in which unused functional units are not clocked

# Portable computer Application

- When an application program is running and the processor is waiting for a key stroke, it can be put into DOZE mode
- If the processor is waiting for disk access, it may go into the NAP mode
- If the processor has been idle for a long time and the power saver is activated, the processor can be put into the SLEEP mode

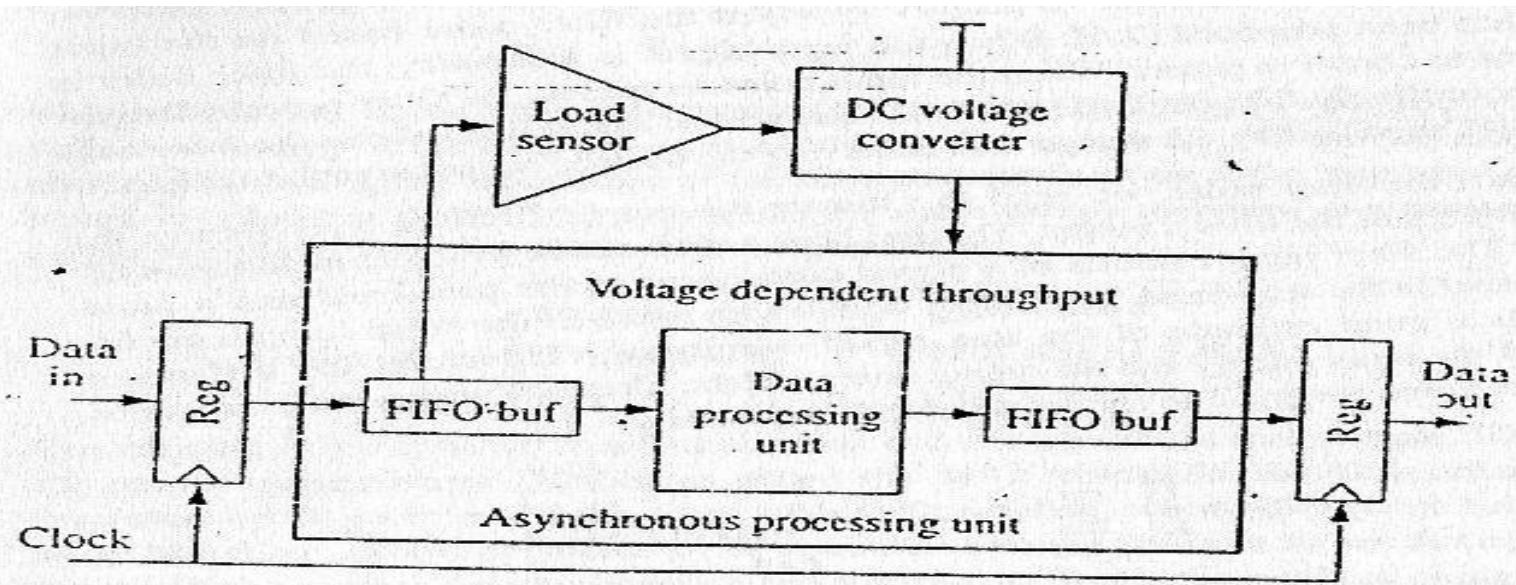
Mode	66Mhz	80Mhz
No power management	2.18W	2.54W
Dynamic power management	1.89W	2.20W
DOZE	397mW	366mW
NAP	113mW	135mW
SLEEP	89mW	105mW
SLEEP without PLL	18mW	19mW
SLEEP without system clock	2mW	2mW

**Power saver modes of  
Motorola's Power PC 603**

# Performance management

- Performance matching is another important issue in architecture level management
- The performance requirement of a system is typically prescribed from a high level perspective
- Since it's is a waste of power to design and operate a circuit beyond its performance requirements, it is important to balance the throughput of each subsystem
- One form of performance management is to scale down the operating voltage on parts of the system that are not performance critical
- The operating voltage of a subsystem can be chosen to match its speed requirement
- This allows the performance and power tradeoff to be applied to the subsystem level and increases the potential for power efficiency

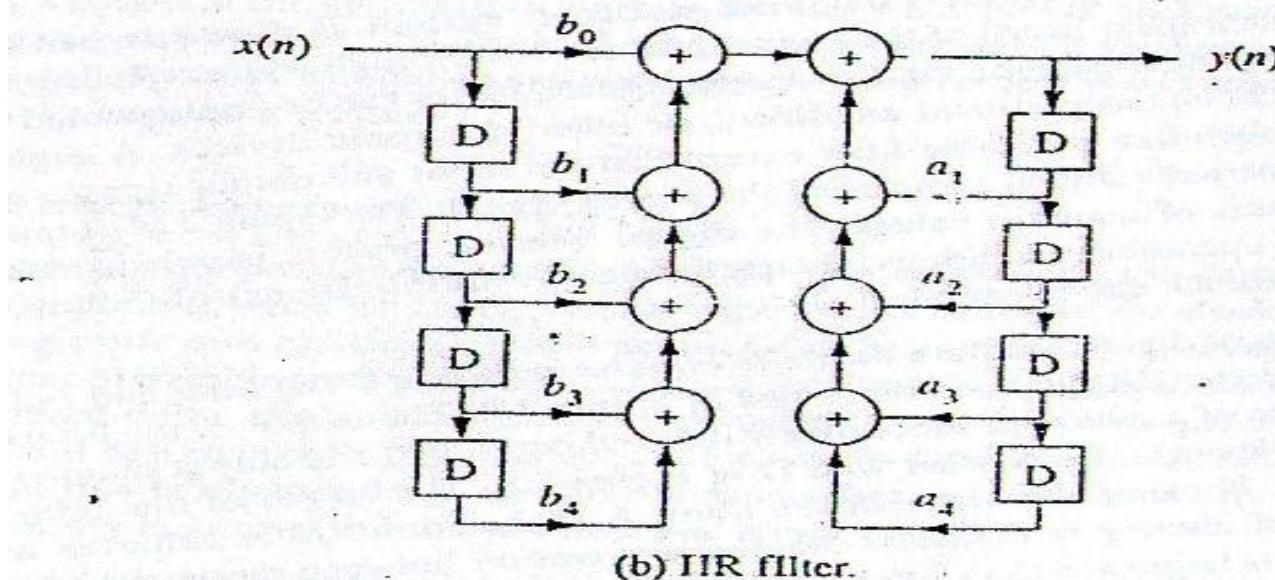
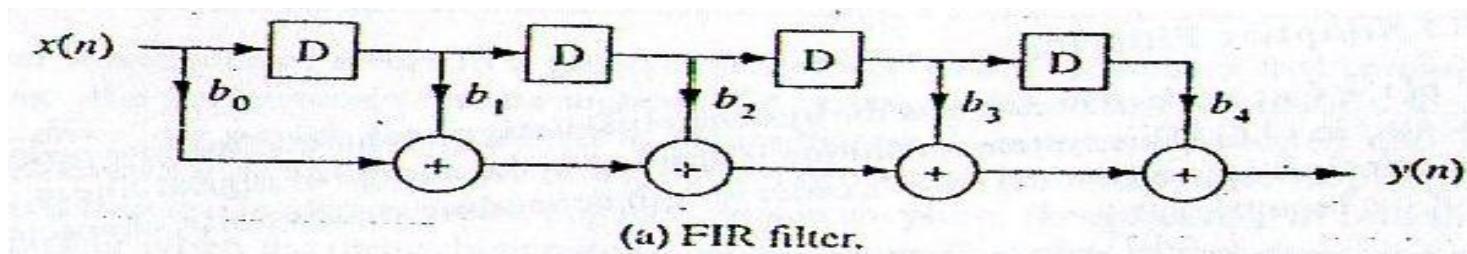
# Adaptive performance management by voltage control



- The throughput of the data processing unit is dependent on the operating voltage, which is typical of an asynchronous processing system
- The load sensor checks the FIFO buffer length to determine the workload of the system
- If the queue is long, the voltage of the system is increased so that the throughput is accelerated
- The voltage is scaled down during light loading to conserve power

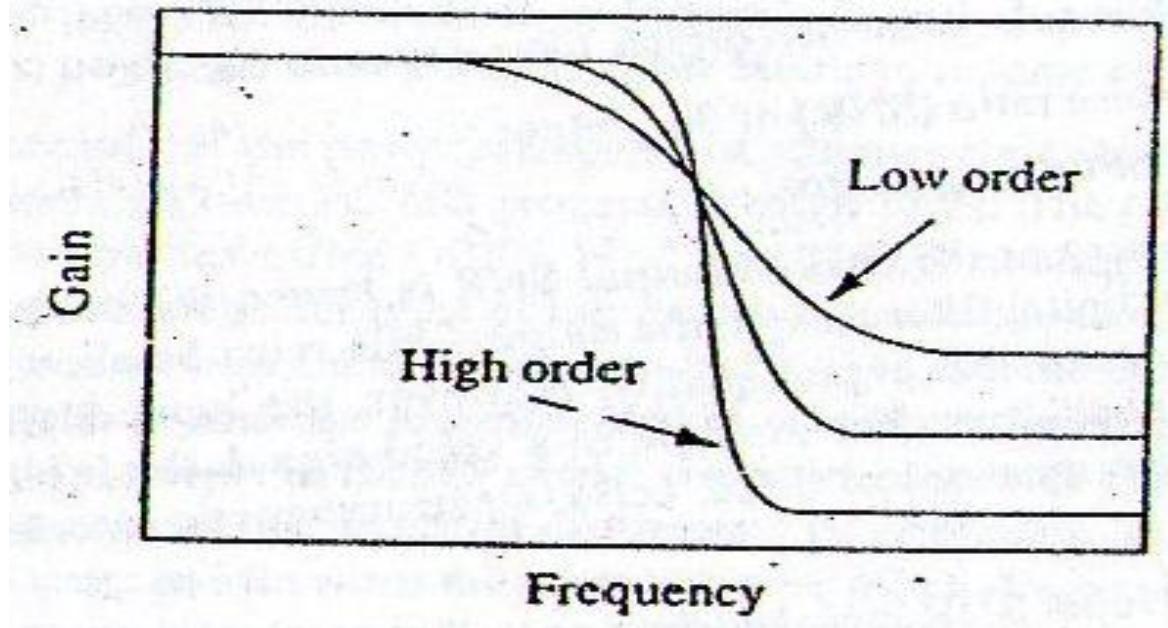
# Adaptive filtering

- The system level design technique for low power can be made through the design of digital filters
- The basic principle is to adjust the filters order length depending on the noise characteristics of the input signal
- Two basic classes of digital filter are FIR and IIR filter as shown below



# Adaptive filtering

- The length of the delay chain of the filter is called the order of the filter
- Higher order filter achieves a higher quality of filtering
  - Attenuation between the passband and the stopband is larger and the slope is steeper
- The filter responses of the three lowpass filters with increasing higher order is shown below



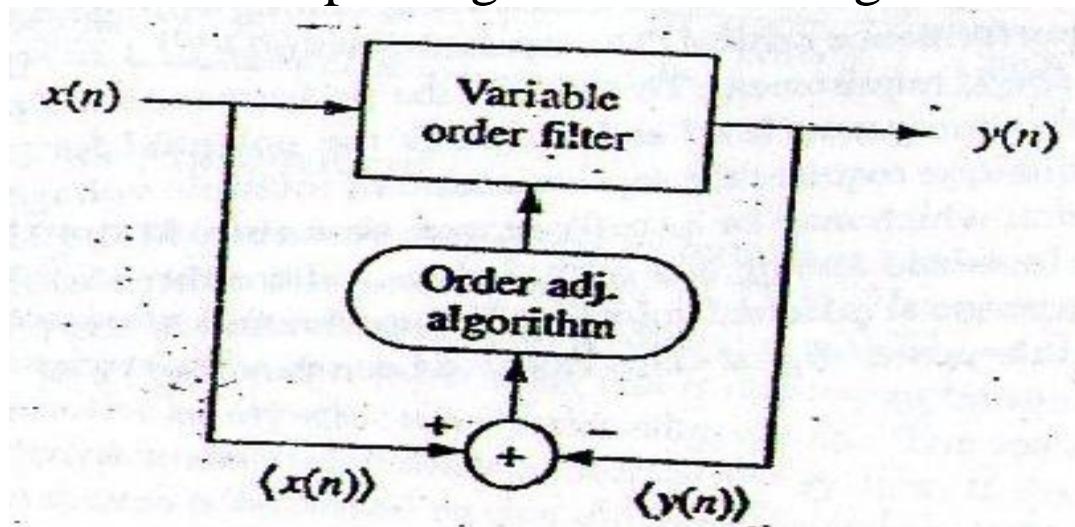
- The quality or order length requirement of a digital filter depends on
  - The desired SNR of the output
  - The noise energy level of the input signal

# Adaptive filtering

- For a fixed order filter, if the input signal is higher than normal (ie input is not noisy), the output SNR may be much higher than the required by the specification
- This means that lower order filtering should be sufficient to achieve the specified output SNR
- Also, the low order filter requires less power to operate because the amount of computation is less
- This motivates a low power technique to adaptively adjust the filter order based on the input SNR
- When the input signal is less noisy, the filter order is reduced to conserve power
- When the input noise increases, the filter order is lengthened accordingly
- The output SNR of the filter will be maintained above the minimum threshold specified by the design thus achieving the desired output signal quality

# Adaptive filtering

- A general framework for updating filter order length



- The filter order update algorithm is as follows

$$Order(n) = \begin{cases} Order(n-1) + N, & Q(n) > MAX \\ Order(n-1), & MIN \leq Q(n) \leq MAX \\ Order(n-1) - N, & Q(n) \leq MIN \end{cases}$$

$$Q(n) = \alpha d(n) E_{SB}[Order(n)]$$

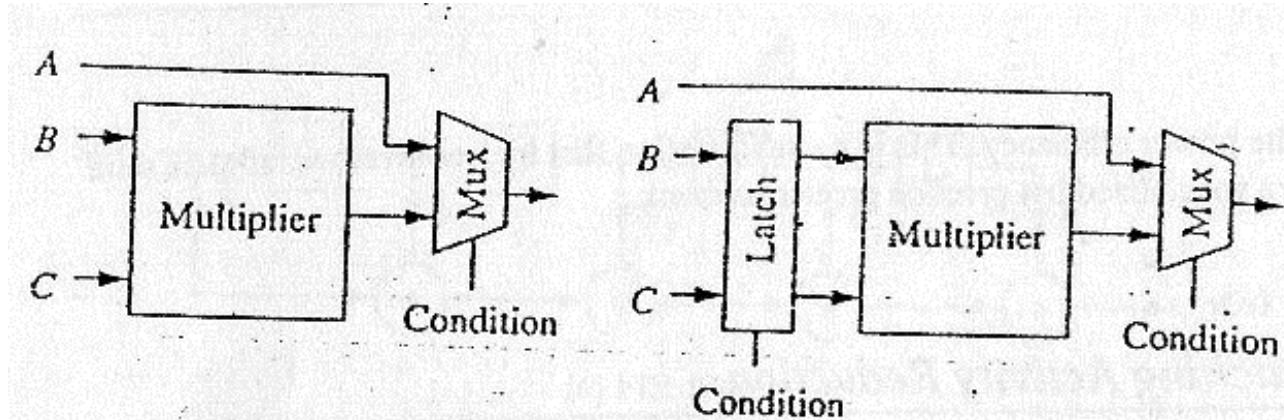
- Order(n) is the filter order,  $E_{SB}$  is the stopband energy of the filter and alpha is the const
- Variable N is a const in which the filter order is adjusted at each sample
- Q(n) represents the noise energy measured at a particular output sample n

# Switching Activity Reduction

- Switching activities are the biggest cause of power dissipation in most CMOS digital systems
- In order to do computation, switching activities cannot be avoided
- However, some switching activities do not contribute to the actual contribution and should be eliminated
- The suppression of switching activities always involves some tradeoff decisions
- In general hardware logic is required to suppress unwanted switching activities and the additional logic itself consumes power
- **Guarded Evaluation**
  - It is a technique to reduce switching activities by adding latches or blocking gates at the inputs of a combinational module if the inputs are not used

# Guarded Evaluation

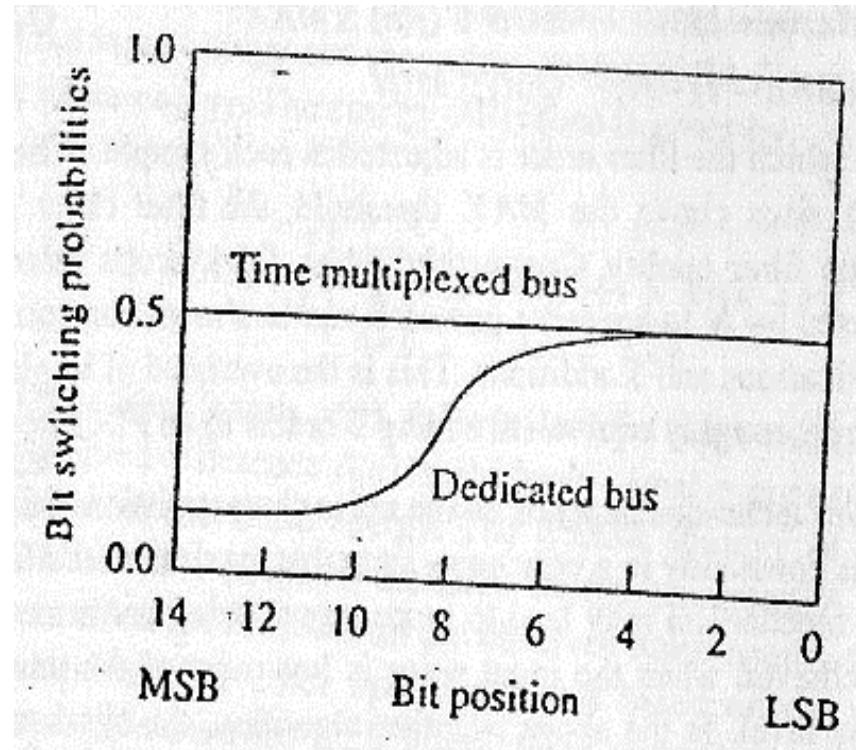
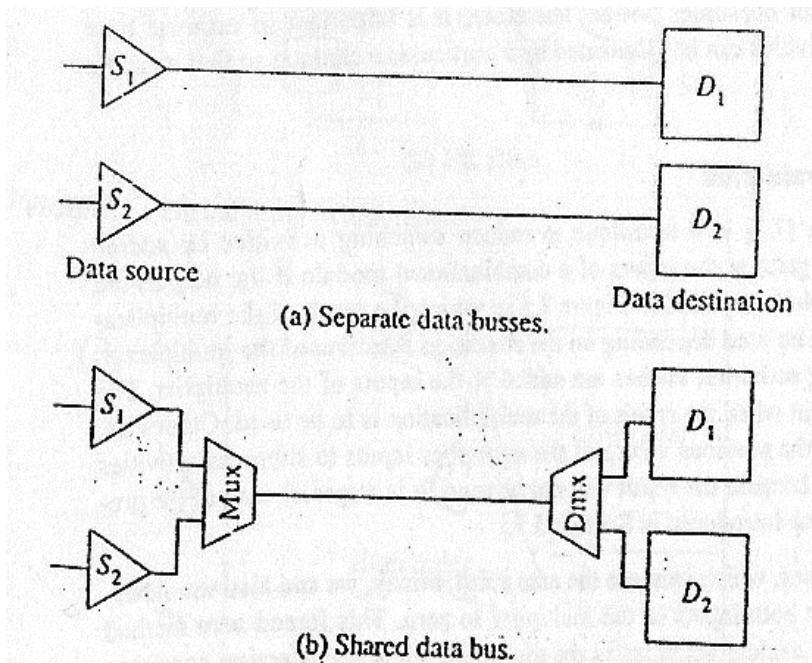
- As shown in figure below the result of multiplication may or may not be used depending on the condition selection of the multiplexer



- To reduce switching activities, latches are added at the input of the multiplexer
- The latches are transparent when the result of multiplication is to be used
- Instead of using latches, which increase the area substantially, we can also use AND gates to mask one or both inputs of the multiplier to zero

# Bus Multiplexing

- One common way to reduce hardware resources in a digital system is to share long data buses with time multiplexing, as shown in left Fig



- At even clock cycles,  $S_1$  uses the shared bus to send data to destination  $D_1$  while at odd cycles, source  $S_2$  send data to  $D_2$
- The switching activity characteristics of the shared and dedicated buses, as shown in right Fig

# Glitch Reduction by Pipelining

- Glitches are unwanted switching activities that occur before a signal settles to its intended value
- They are caused by transitions at multiple inputs of a gate within a clock cycle
- Glitches are undefined and unpredictable behavior of a circuit that cannot be analyzed deterministically
- Glitches of a signal node are dependent on the logic depth of the node
- A logically deep node is typically affected by more primary input switching and therefore more susceptible to glitches
- One way to reduce glitches is to shorten the depth of combinational logic by adding pipeline registers

# Parallel Architecture with Voltage Reduction

- Parallelism has been traditionally used to improve the computational throughput of high performance digital systems
- Parallelism essentially trades area off for a lower operating frequency or higher throughput
- The same tradeoff idea can also be applied to achieve power reduction
- In a uniprocessing system, the power dissipation is given by

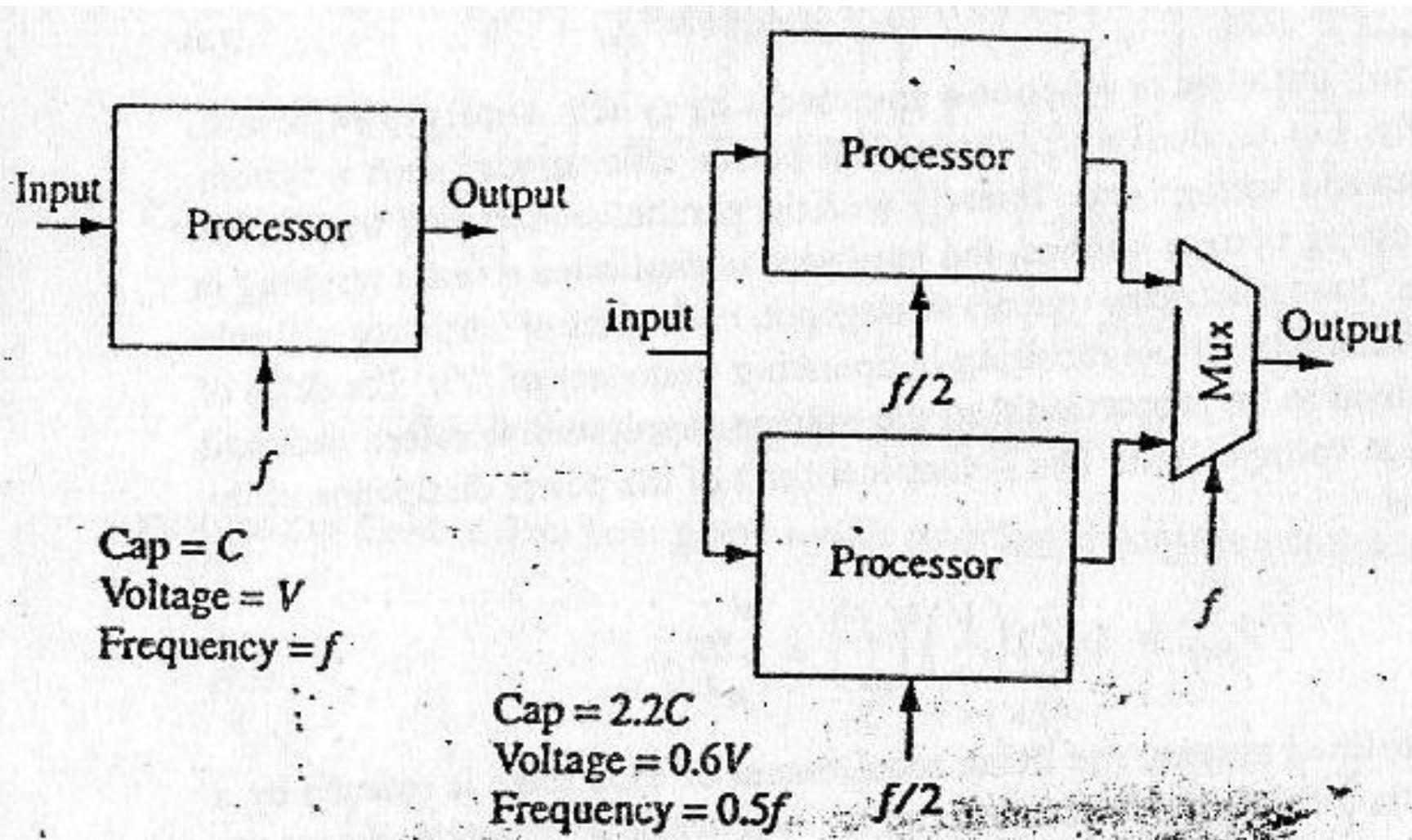
$$P_{uni} = CV^2f$$

- Where C is the average capacitance switched and V is the operating voltage of the uniprocessing system
- The power of parallel system is given by

$$P_{par} = (2.2C)(0.6V)^2(0.5f) = 0.396 P_{uni}$$

- 60% power reduction is achieved by parallelism

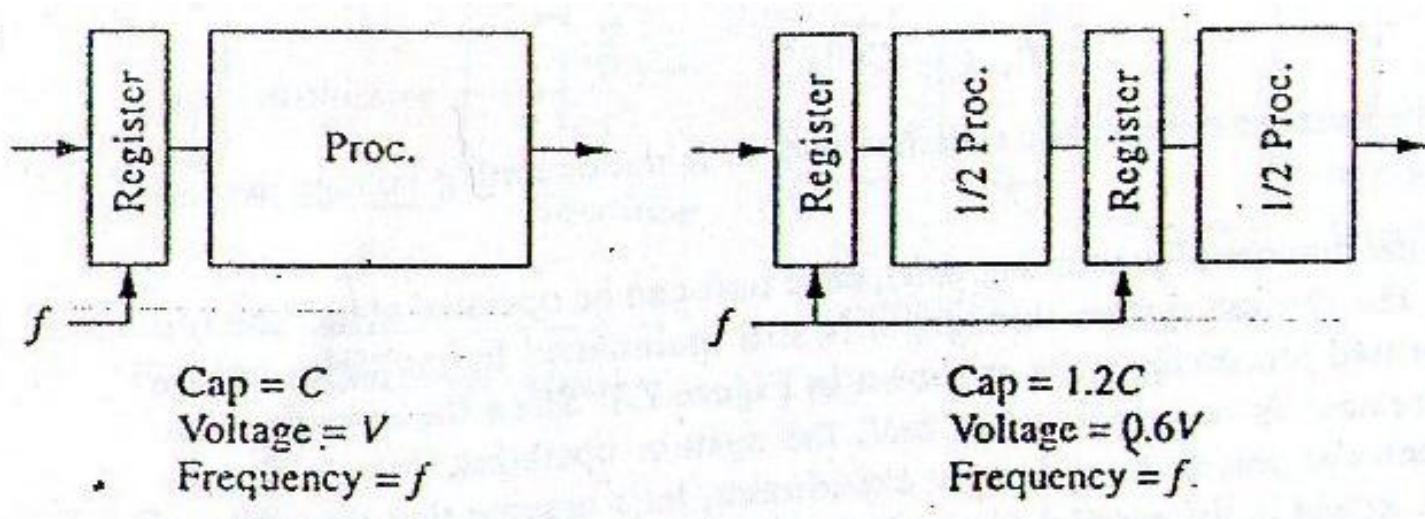
# Parallel Architecture with Voltage Reduction



Power dissipation of uniprocessing and parallel processing systems

# Power Efficiency by Pipelining

- The above parallel technique increases the chip area by at least twice
- If the area penalty of a parallel system is prohibitive, pipelining can offer similar tradeoff results with less area overhead but more complexity in controller design



$$P_{\text{pip}} = (1.2C)(0.6V)^2f = 0.432 P_{\text{uni}}$$

# Flow Graph Transformation

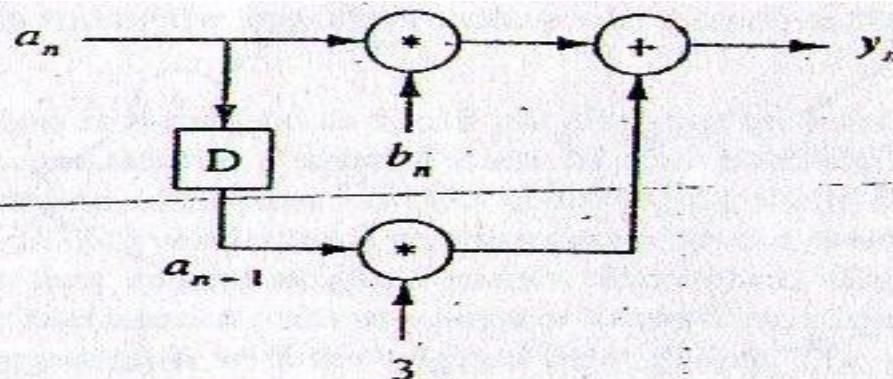
- Here, we focus on a system level technique for the design of special purpose architecture DSP systems
- Such systems are characterized by computation intensive data path operations with simple control structures
- The system architecture can be represented by a control data flow graph
- The graph consists of control nodes and data nodes connected by directed edges
- Control nodes change the flow of data that pass through it
- Data nodes provide computation operators for the input data streams such as addition, multiplication, shift etc
- The graph edges represent the data streams of the system

# Flow Graph Transformation

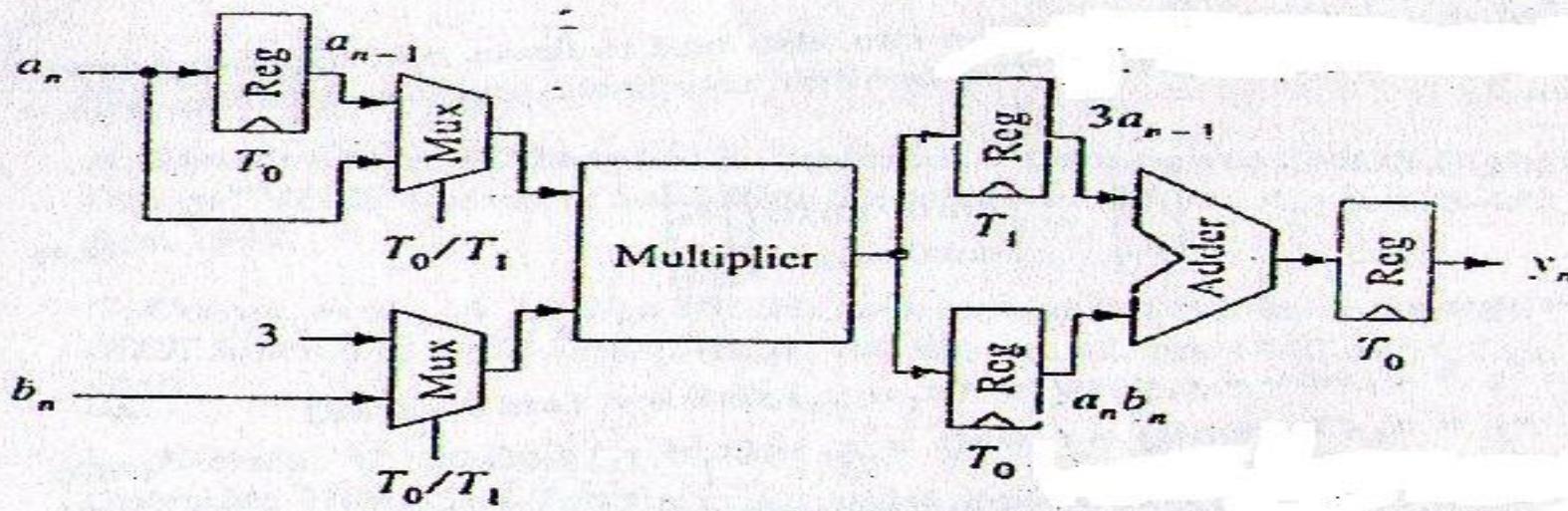
- A control data graph expresses the conceptual computation algorithm of the system
- It provides basic information such as the number of control and data operators, their ordering dependencies and the inherent parallelism that may exist
- The path from system input to output is a good estimate of the delay of the system
- The number of operator nodes is an estimate of the computation needs and the complexity of the system
- A control data flow graph is often the starting point to derive the actual hardware architecture of a system by mapping the operators and edges to actual hardware modules and busses respectively
- A controller schedules the operators to perform the desired computation in the proper order

# Flow Graph Transformation

- The below example illustrates a simple mapping from the control data flow graph to the system hardware architecture



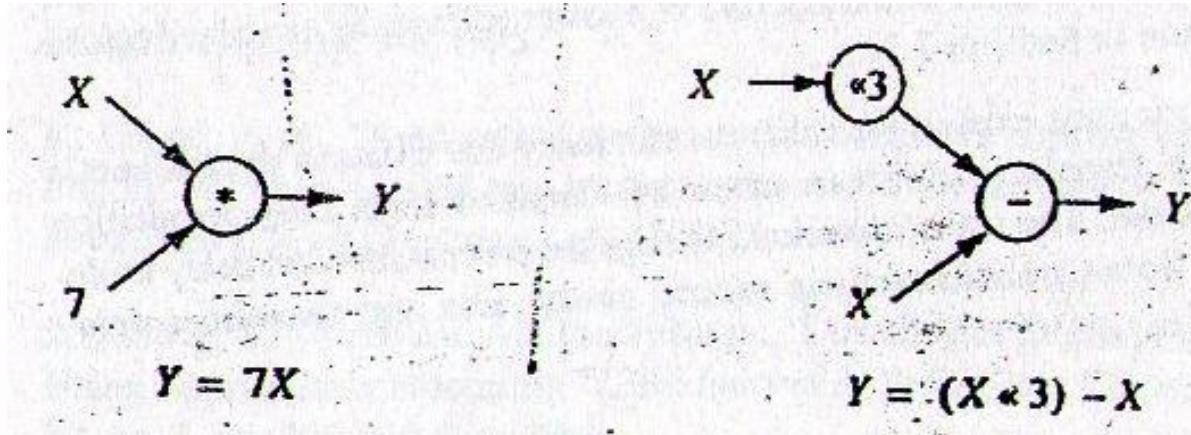
(a) Control data flow graph of  $y_n = a_n b_n + 3 a_{n-1}$ .



(b) Hardware architecture and scheduling.

# Operator Reduction

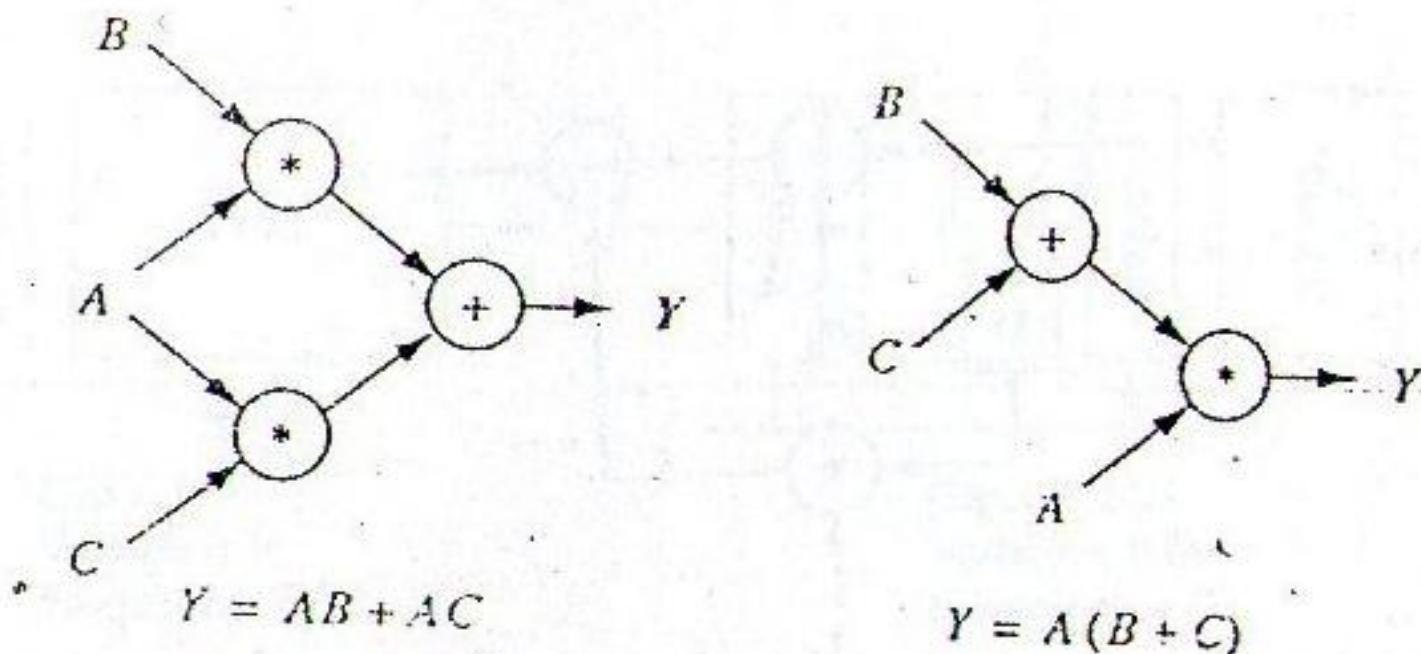
- The transformations preserve the functionality of the graph so that the resulting graph is computationally equivalent to the original work
- The transformed graph represents an alternate hardware implementation with a different tradeoff
- If the system operating voltage is variable, the transformed graph may offer a lower voltage implementation by reducing the worst case delay path
- An example of operator transformation is illustrated below



Constant multiplication versus shift and addition

# Operator Reduction

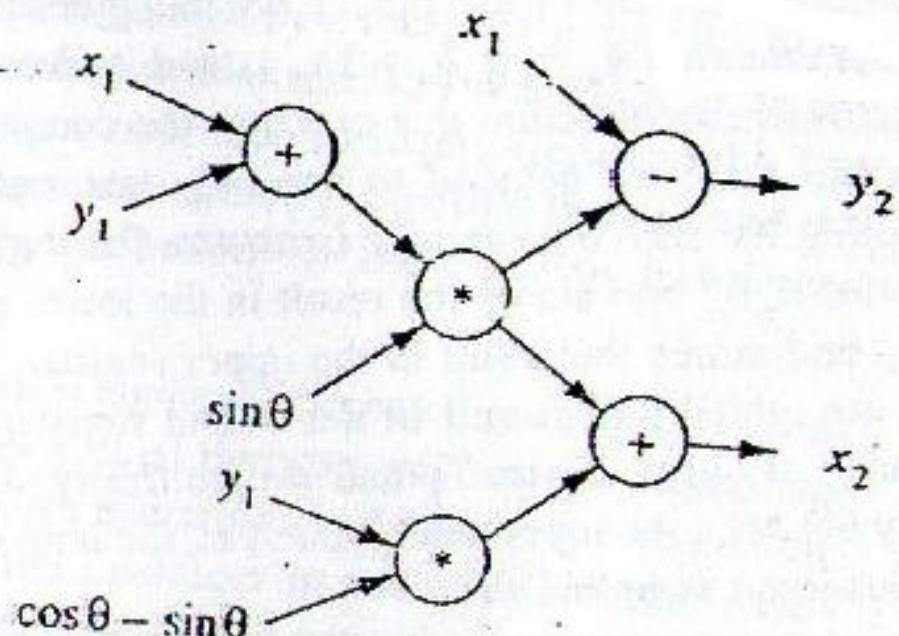
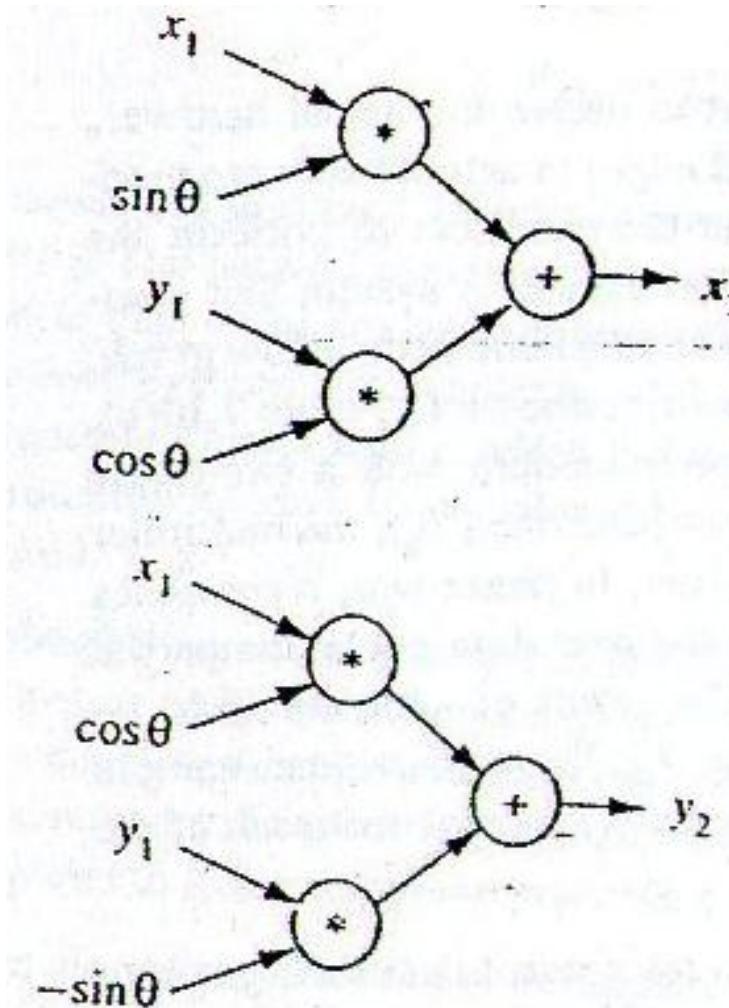
- The multiplication by an integer constant can be replaced by binary shift (multiplication by powers of 2) and add / subtract operations, which may be power efficient
- Rearranging the order of computation can also lead to fewer operations and lower power as shown below



Associative Transformation

# Operator Reduction

- Also the direct and transformed flow graphs of coordinate rotation operation is shown below



# Loop Unrolling

- The control data flow graphs of DSP systems often contain loops, as a result of recursive computations
- An important technique for flow graph transformation is to unroll the loop
- Loop unrolling is a method to apply parallelism to the computation
- Consider a simple recursive computation of an IIR filter

$$y_n = b_0 x_n + a_1 y_{n-1}$$

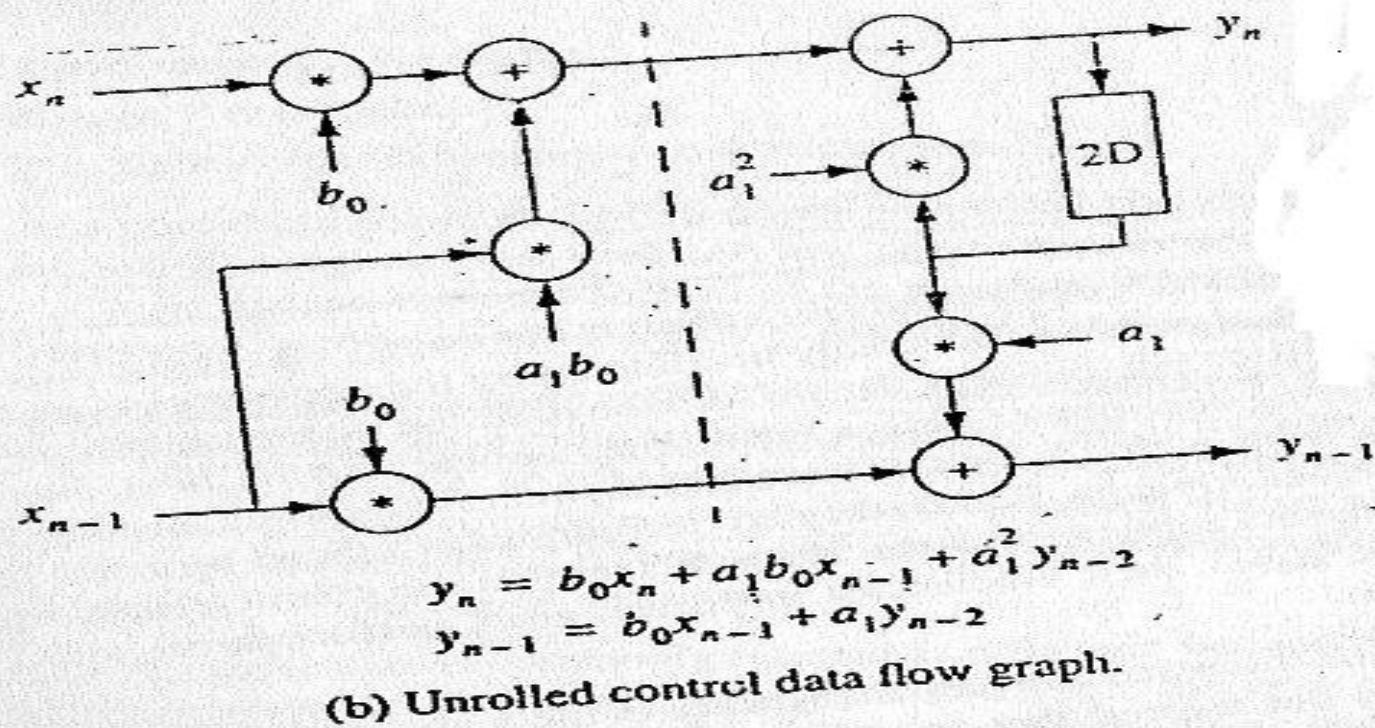
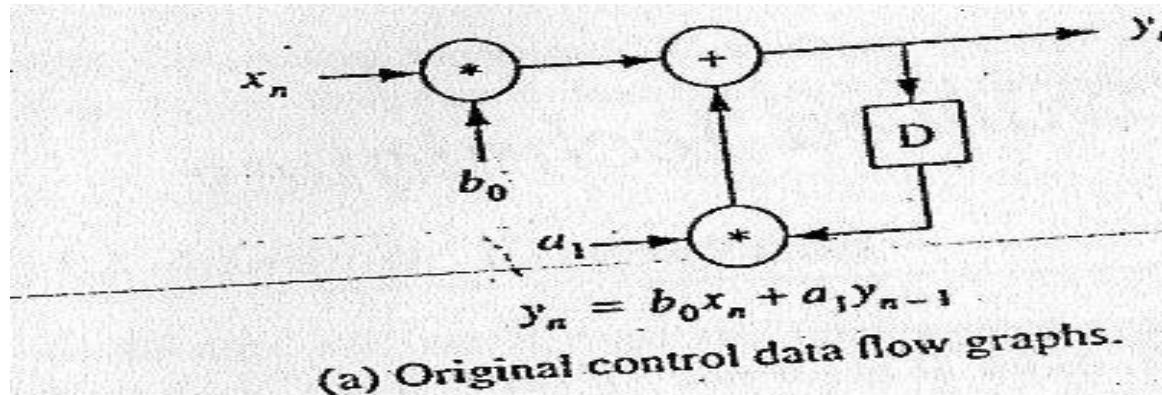
- We can unroll the loop once and obtain

$$\begin{aligned} y_n &= b_0 x_n + a_1 b_0 x_{n-1} + a_1^2 y_{n-2} \\ y_{n-1} &= b_0 x_{n-1} + a_1 y_{n-2} \end{aligned}$$

- After loop unrolling, two output values are produced using two input values in a single computation cycle
- The unrolled computation structure increases the computation by more than twice

# Loop Unrolling

- The data flow graphs of the original and unrolled computation are illustrated below



# Loop Unrolling

- If the unrolled computation structure is implemented directly, the power efficiency should be worse than the original implementation because of the increased computation
- However, the unrolled structure allows us to apply pipelining by adding pipeline registers on the graph edges crossing the vertical dashed line
- With pipelining , the longest delay path of the unrolled graph is identical to the original path (i.e. a multiplication followed by a addition)
- Since the unrolled graph produces two outputs simultaneously, it can be implemented with half the operating frequency of the original path
- Thus, its critical delay is identical but the operating frequency is halved
- This allows us to lower the operating voltage to improve the overall system power efficiency

# **UNIT-III**

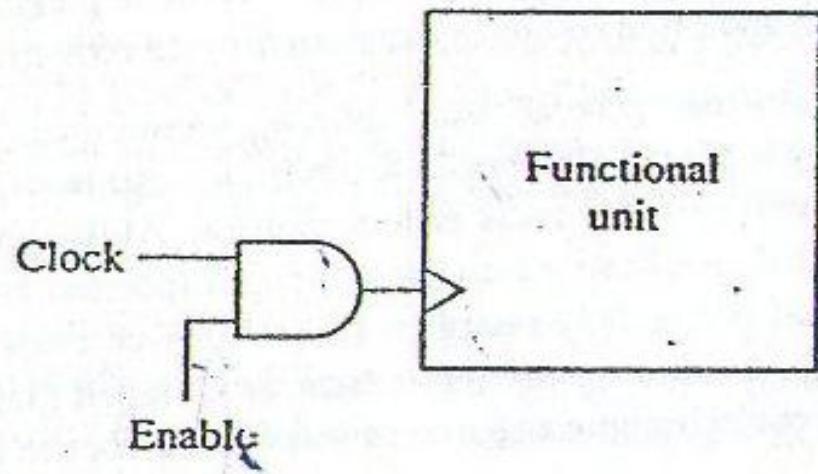
# **Power Reduction in Clock Networks**

# Introduction

- In a synchronous digital chip, the clock signal is generally one with the highest frequency
- The clock signal typically drives a large load because it has to reach many sequential elements distributed throughout the chip
- Therefore, clock signals have been a notorious source of power dissipation because of high frequency and load
- It has been observed that clock distribution can take up to 40% of the total power dissipation of a high performance microprocessor
- Many techniques have been devoted to the power efficiency of clock generation and distribution

# Clock Gating

- Clock gating is the most popular method for power reduction of clock signals
- When the clock signal of a fundamental module (ALUs, memories etc) is not required for some extended period, we use a gating function (NAND or NOR gate) to turn off the clock feeding the module
- The gating signal should be enabled or disabled at a much slower rate compared to the clock frequency
- Clock gating saves power by reducing unnecessary clock activities inside the gated module

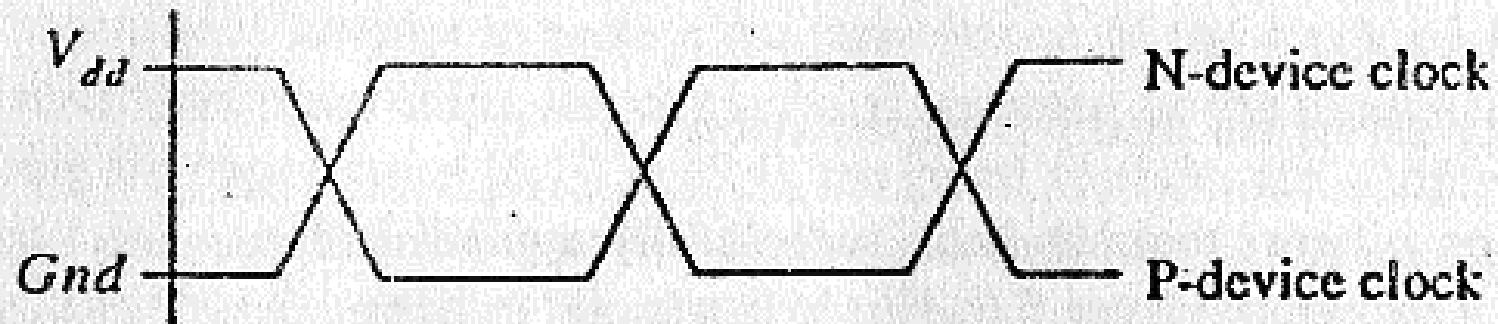


# Reduced Swing Clock

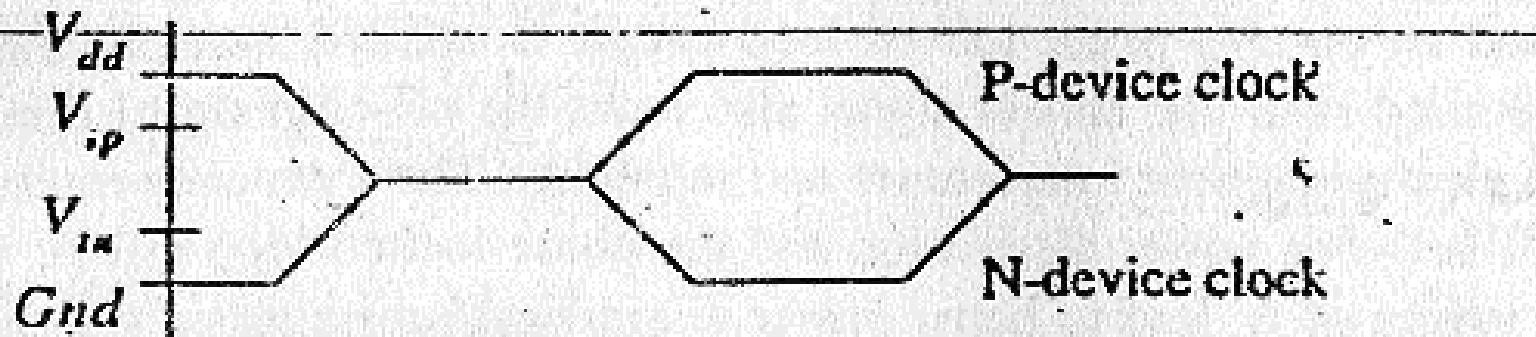
- Referring  $P = C V^2 f$  equation, the most attractive parameter for power reduction is the voltage swing  $V$  due to the quadratic effect
- Also, it is difficult to reduce the load capacitance or frequency of clock signals due to performance reasons
- Consider a 5 V digital CMOS chip with a N – transistor threshold voltage of 0.8 V
- N – transistor will turn on if the clock signal is above 0.8 V
- Hence, if we limit the swing of the N – transistor clock signal from 0 to 2.5 V (half swing)
- Further, the on – off characteristics of all N – transistor remains digitally identical
- Similar observation can be made for the clock signal feeding a P – transistor

# Reduced Swing Clock

- The figure below illustrates the clock waveform of the half swing clocking scheme



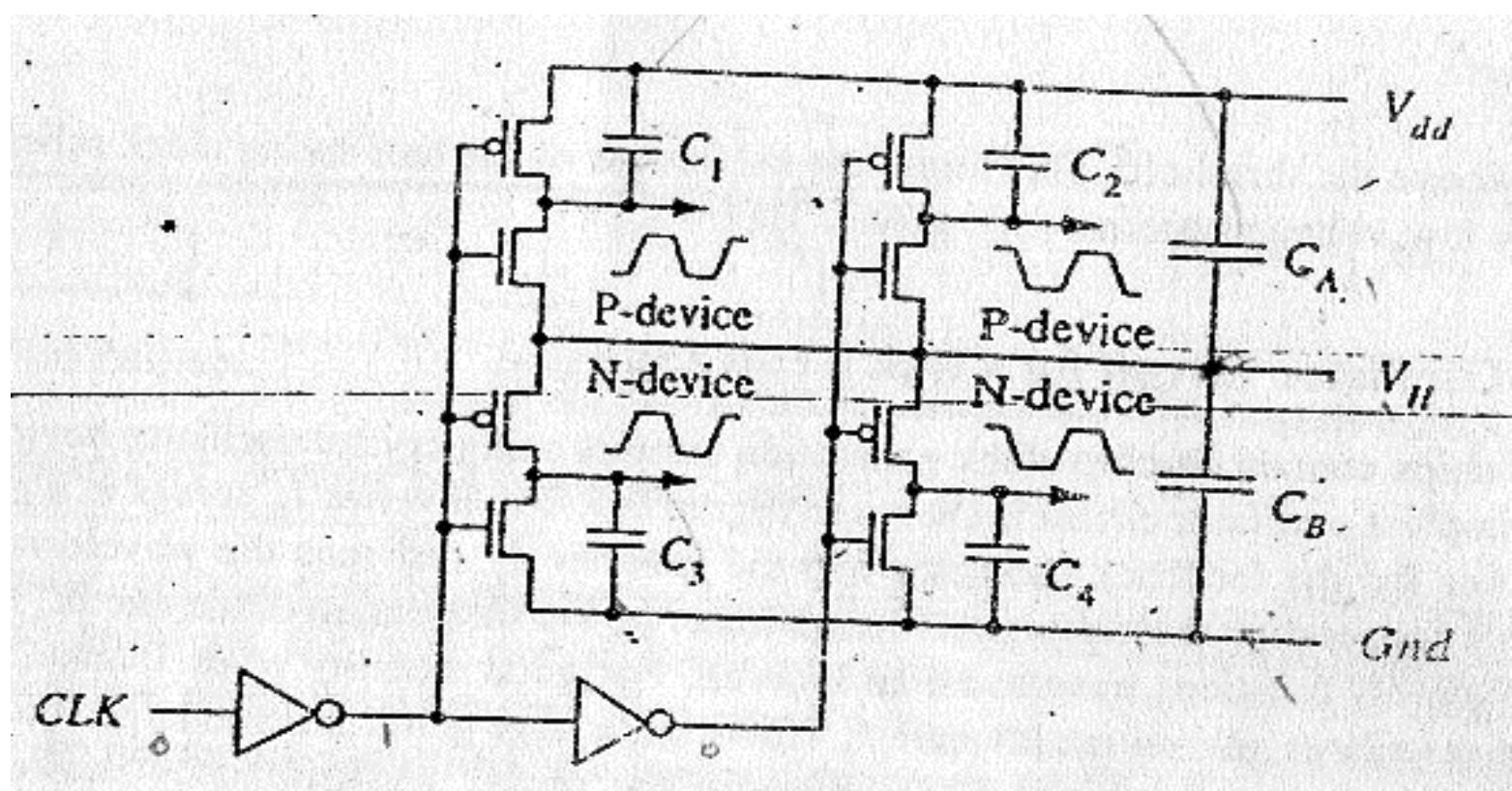
(a) Regular clock.



(b) Half-swing clock.

# Reduced Swing Clock

- Generating the half swing clock signal is also relatively simple using the charge sharing principle with stacked inverters
- The circuit to generate the clock waveform is shown below



# Reduced Swing Clock

- The capacitance  $C_1$ ,  $C_2$ ,  $C_3$  and  $C_4$  are the parasitic capacitances of the circuit
- $C_A$  and  $C_B$  are add on capacitance much larger than the parasitic capacitance
- From the principle of charge sharing, when CLK is low the voltage at  $V_H$  is given by

$$V_H = \frac{C_1 + C_A}{C_1 + C_4 + C_A + C_B} V_{dd}$$

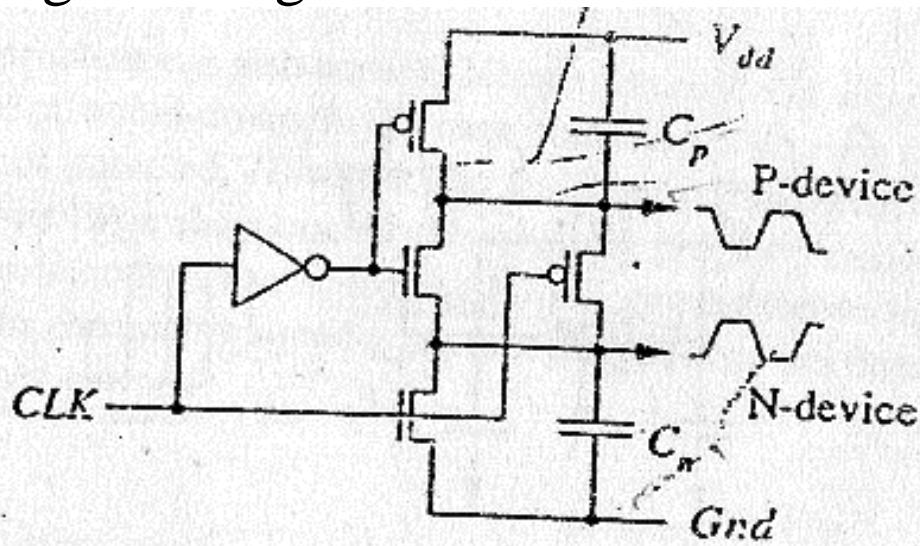
- When CLK is high, the voltage is

$$V_H = \frac{C_2 + C_A}{C_2 + C_3 + C_A + C_B} V_{dd}$$

- The circuit generates two-phase, half-swing waveforms

# Reduced Swing Clock

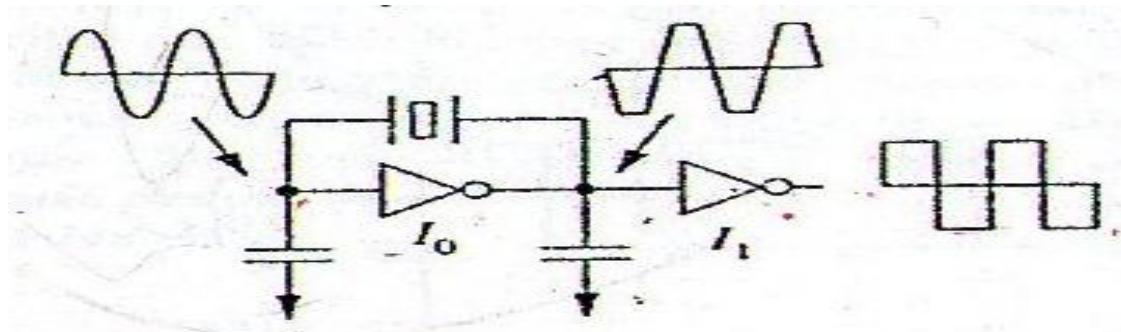
- Creating large on chip capacitance  $C_A$  and  $C_B$  is difficult and may require an external pin for the node  $V_H$
- A simpler charge sharing circuit is shown below



- The circuit relies on the parasitic loading of the clock lines  $C_P$  and  $C_N$  to achieve the charge sharing effect
- The capacitance  $C_P$  and  $C_N$  should be equalized to obtain the proper half swing waveform

# Oscillator Circuit for Clock Generation

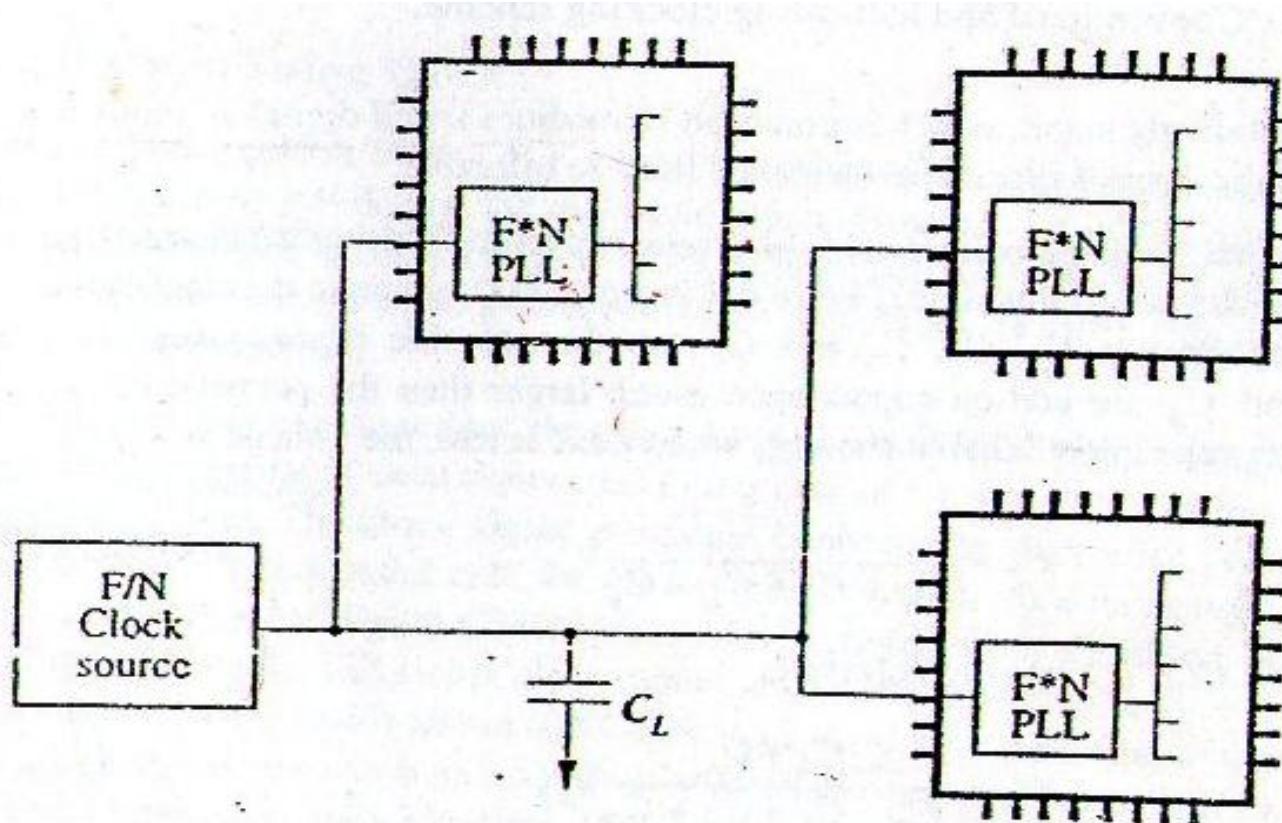
- The simplest oscillator circuit is shown below



- The inverter  $I_0$  serves as a gain amplifier for the feedback oscillator and the inverter  $I_1$  reshapes the waveform to obtain a proper digital clock signal
- For good stability of oscillation, we like to set a large gain on the inverter  $I_0$
- However, a large gain results in large voltage swing and short circuit current that increase the power dissipation
- Hence, it is important to tune the transistor sizes of the inverters to achieve a trade off between quality of oscillation and power dissipation of the circuit

# Frequency Division and Multiplication

- A power reduction scheme using frequency division and multiplexing is shown below



- This is common for off chip clock signals because they drive very large capacitance

# Frequency Division and Multiplication

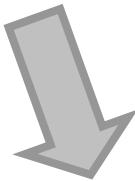
- Off chip clock signal runs at a slower speed and on chip PLL circuit is used to multiply the frequency to the desired rate
- The slower signal also eases the off chip signal distribution in terms of electromagnetic interference and reliability
- The frequency multiplier  $N$  is a trade off between power dissipation and the PLL circuit complexity
- Larger values of  $N$  lead to better power dissipation but increases the design complexity and performance of the PLL circuit

# Algorithm & Architectural Level Methodologies

# Agenda

- Recap
- Power reduction on
  - Gate level
  - Architecture level
  - Algorithm level
  - System level

# Recap: Problems of Power Dissipation



- Continuously increasing performance demands
  - ➔ Increasing power dissipation of technical devices
  - ➔ Today: power dissipation is a main problem
- **High Power dissipation** leads to:

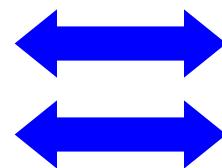
- (悲剧) Reduced time of operation
- (悲剧) Higher weight (batteries)
- (悲剧) Reduced mobility



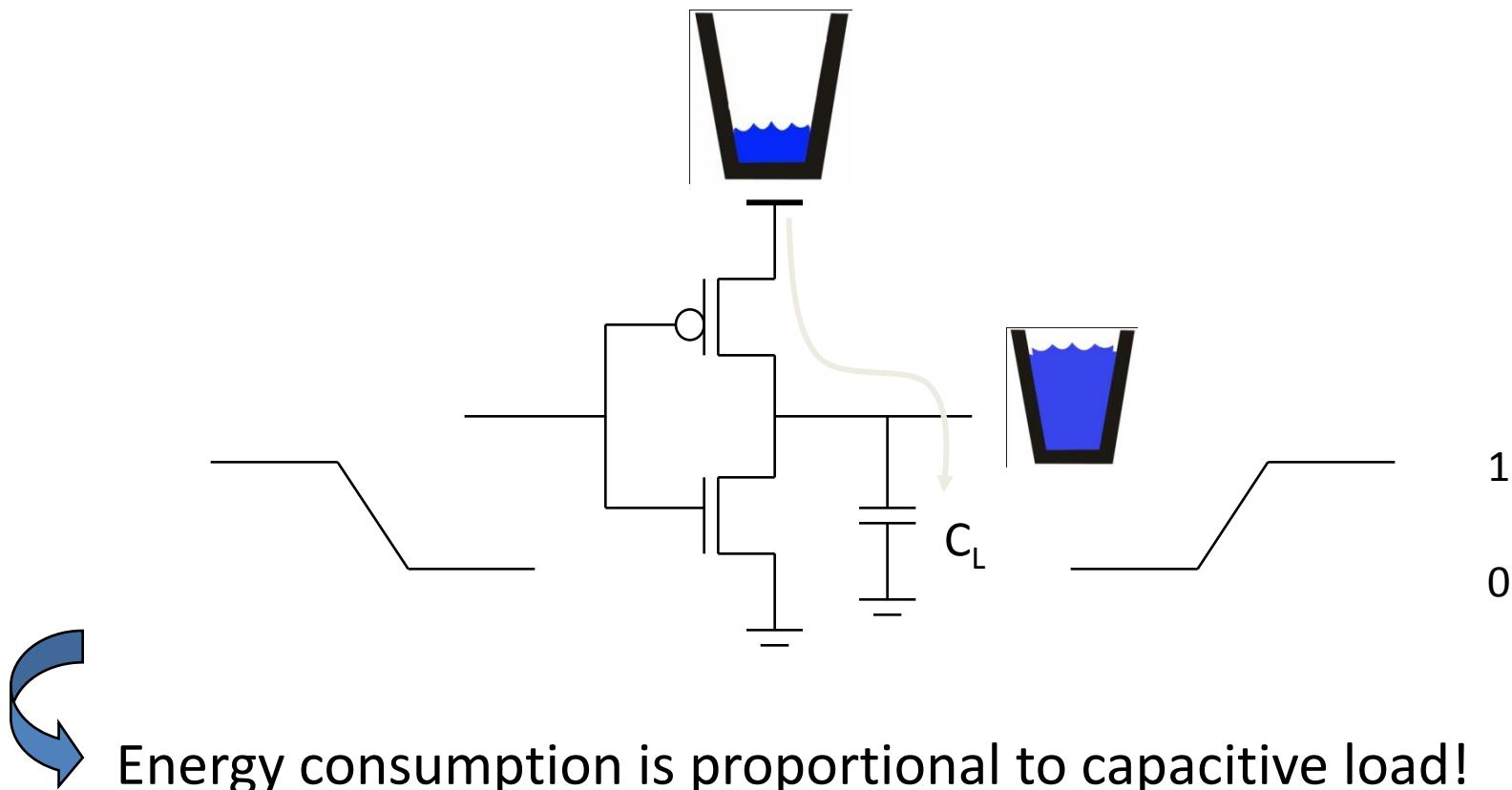
- (悲剧) High efforts for cooling
- (悲剧) Increasing operational costs
- (悲剧) Reduced reliability

# Recap: Consumption in CMOS

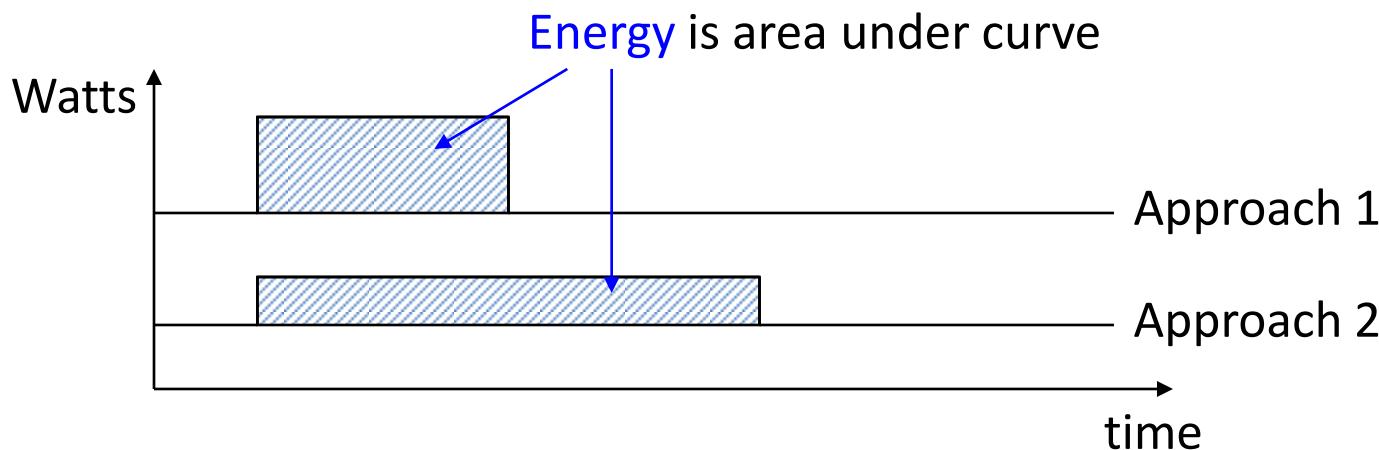
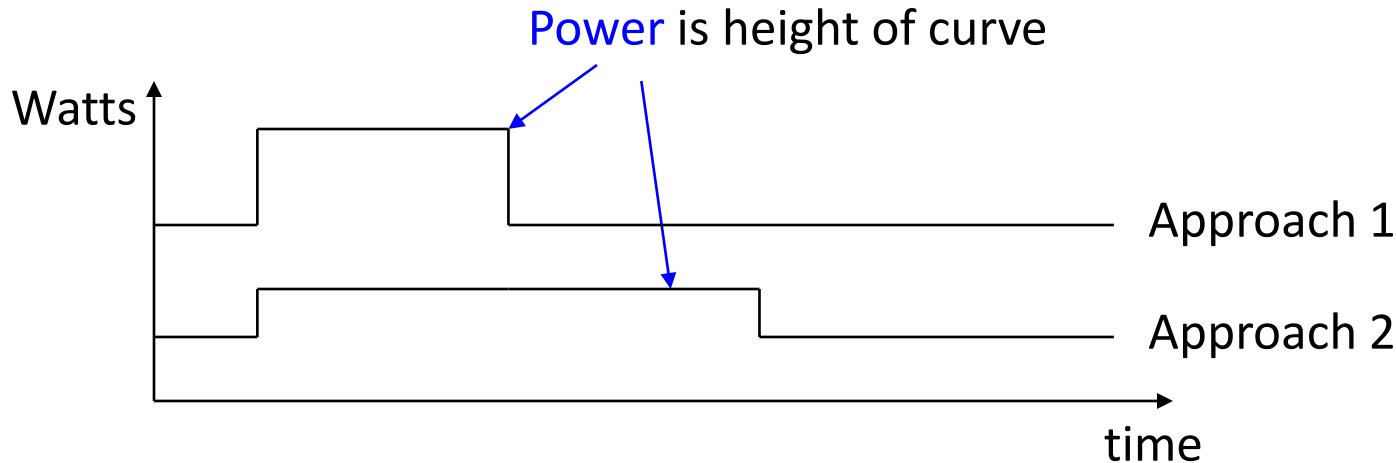
- ❑ Voltage (Volt, V)
- ❑ Current (Ampere, A)
- ❑ Energy



- Water pressure (bar)
- Water quantity per second (liter/s)
- Amount of Water



# Recap: Energy and Power



Energy = Power \* time for calculation = Power \* Delay



# Recap: Power Equations in CMOS

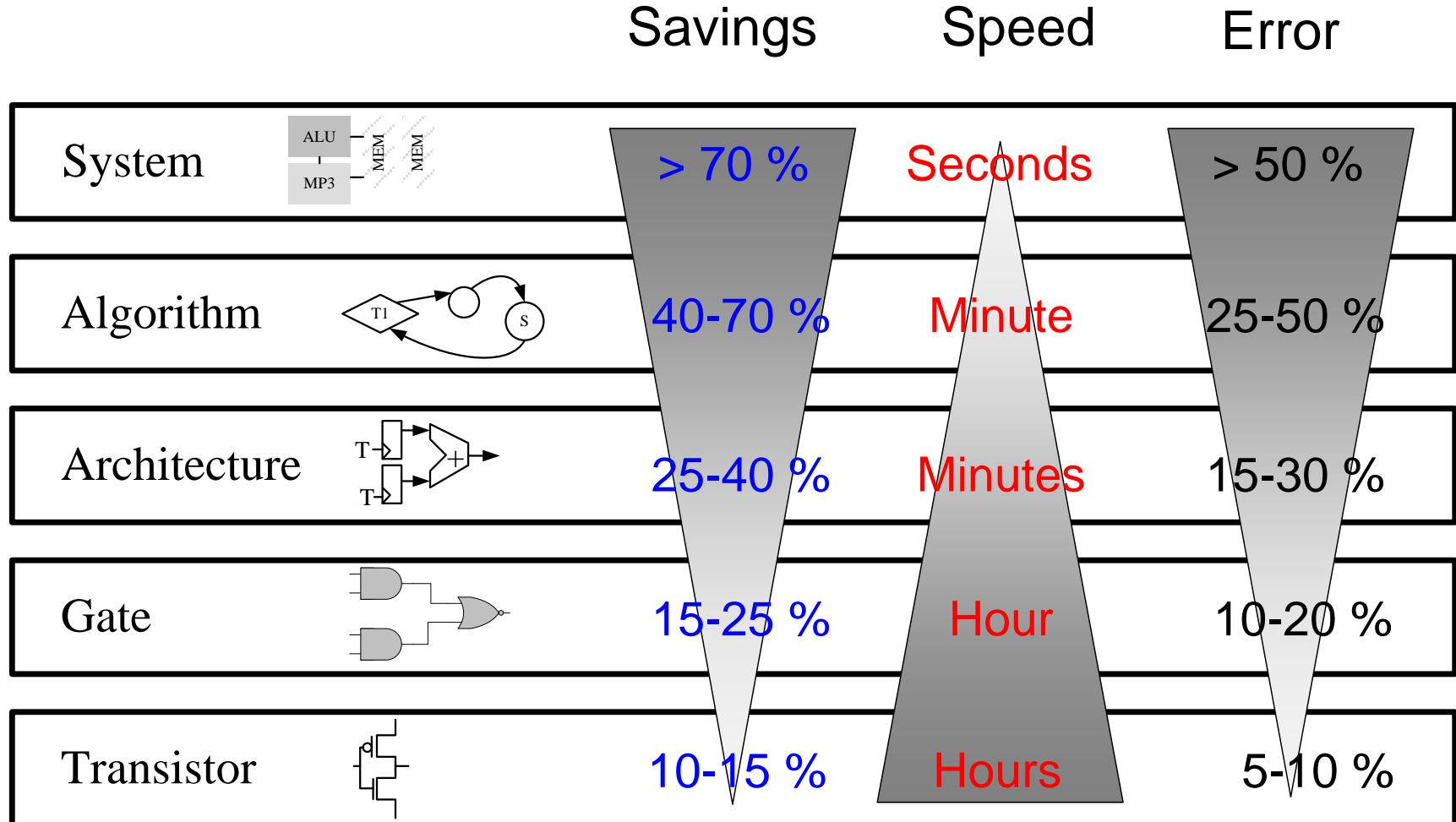
$$P = \alpha f C_L V_{DD}^2 + V_{DD} I_{peak} (P_{0 \rightarrow 1} + P_{1 \rightarrow 0}) + V_{DD} I_{leak}$$

Dynamic power  
( $\approx 40 - 70\%$  today  
and decreasing  
relatively)

Short-circuit power  
( $\approx 10\%$  today and  
decreasing absolutely)

Leakage power  
( $\approx 20 - 50\%$  today  
and increasing)

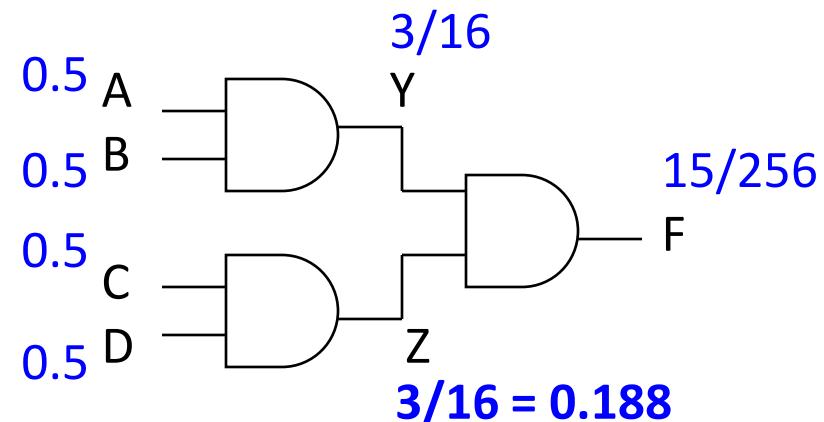
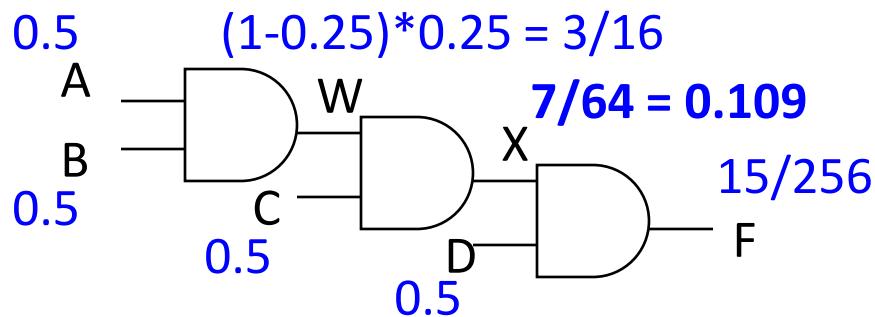
# Recap: Levels of Optimization



# Recap: Logic Restructuring

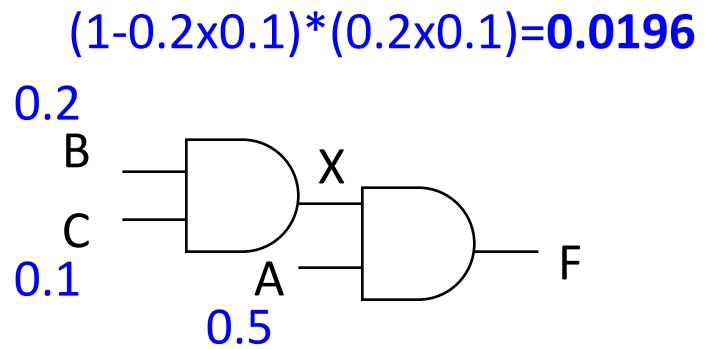
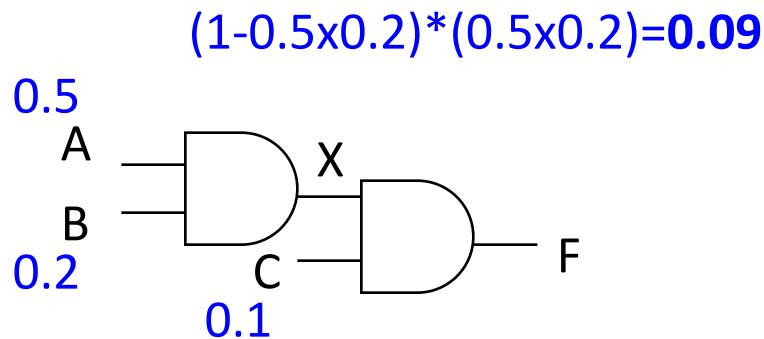
- Logic restructuring: changing the topology of a logic network to reduce transitions

$$\text{AND: } P_{0 \rightarrow 1} = P_0 * P_1 = (1 - P_A P_B) * P_A P_B$$



- Chain implementation has a lower overall switching activity than tree implementation for random inputs
- BUT: Ignores glitching effects

# Recap: Input Ordering

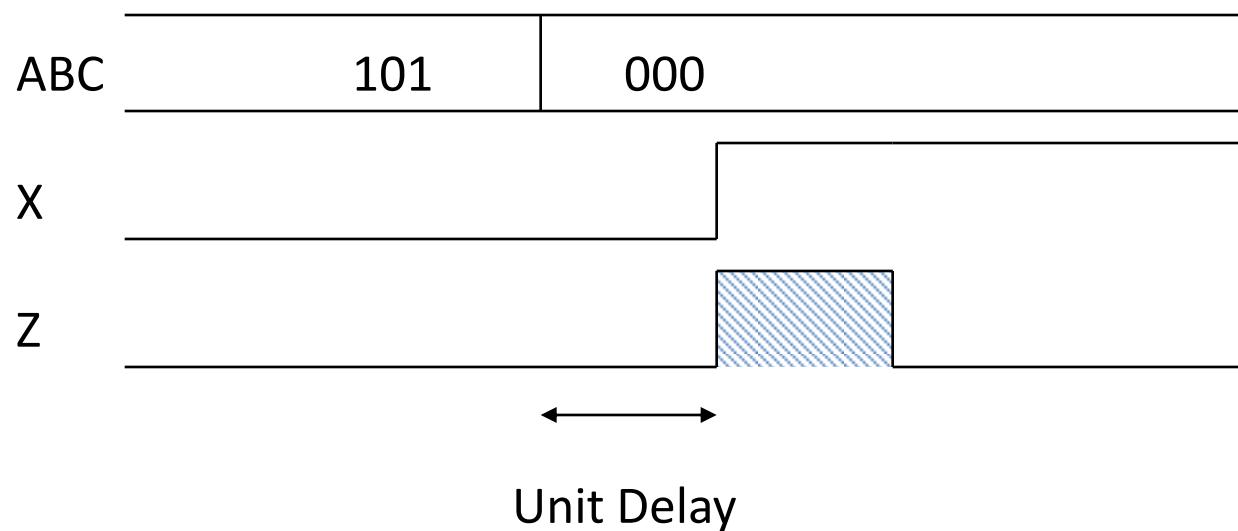
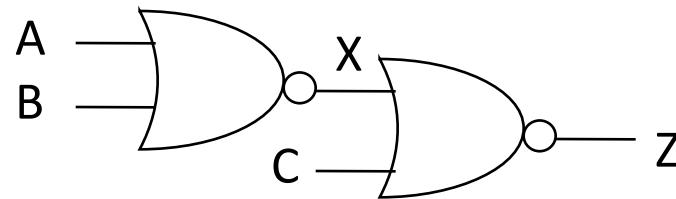


$$\text{AND: } P_{0 \rightarrow 1} = (1 - P_A P_B) * P_A P_B$$



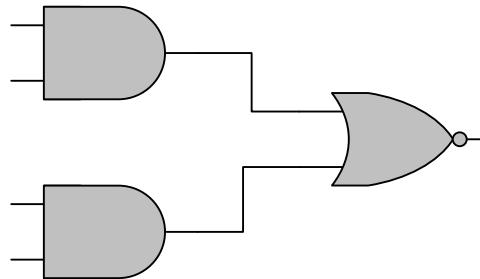
Beneficial: postponing introduction of signals with a **high** transition rate (signals with signal probability close to 0.5)

# Recap: Glitching



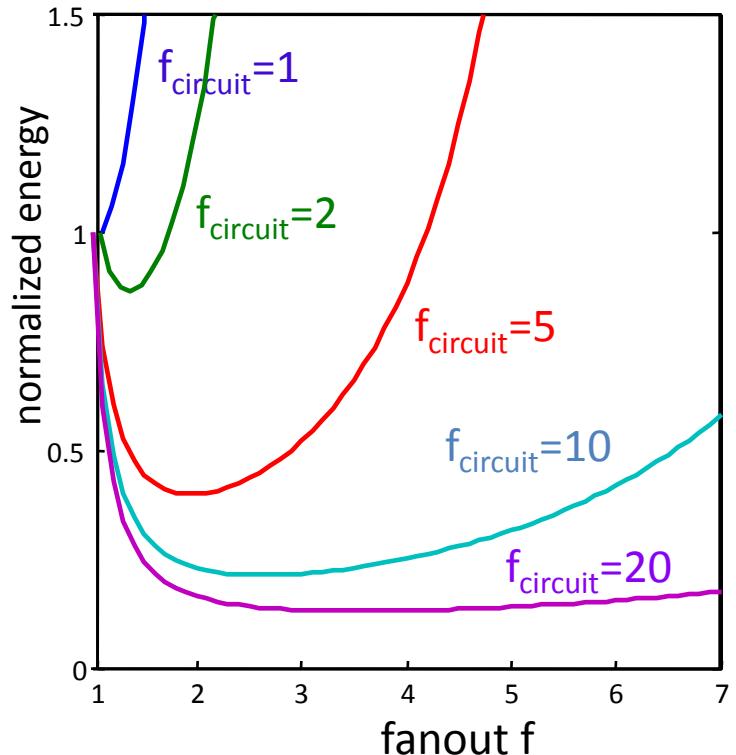
# Design Layer: Gate Level

- Basic elements:
  - Logic gates
  - Sequential elements (flipflops, latches)
- Behavior of elements is described in libraries

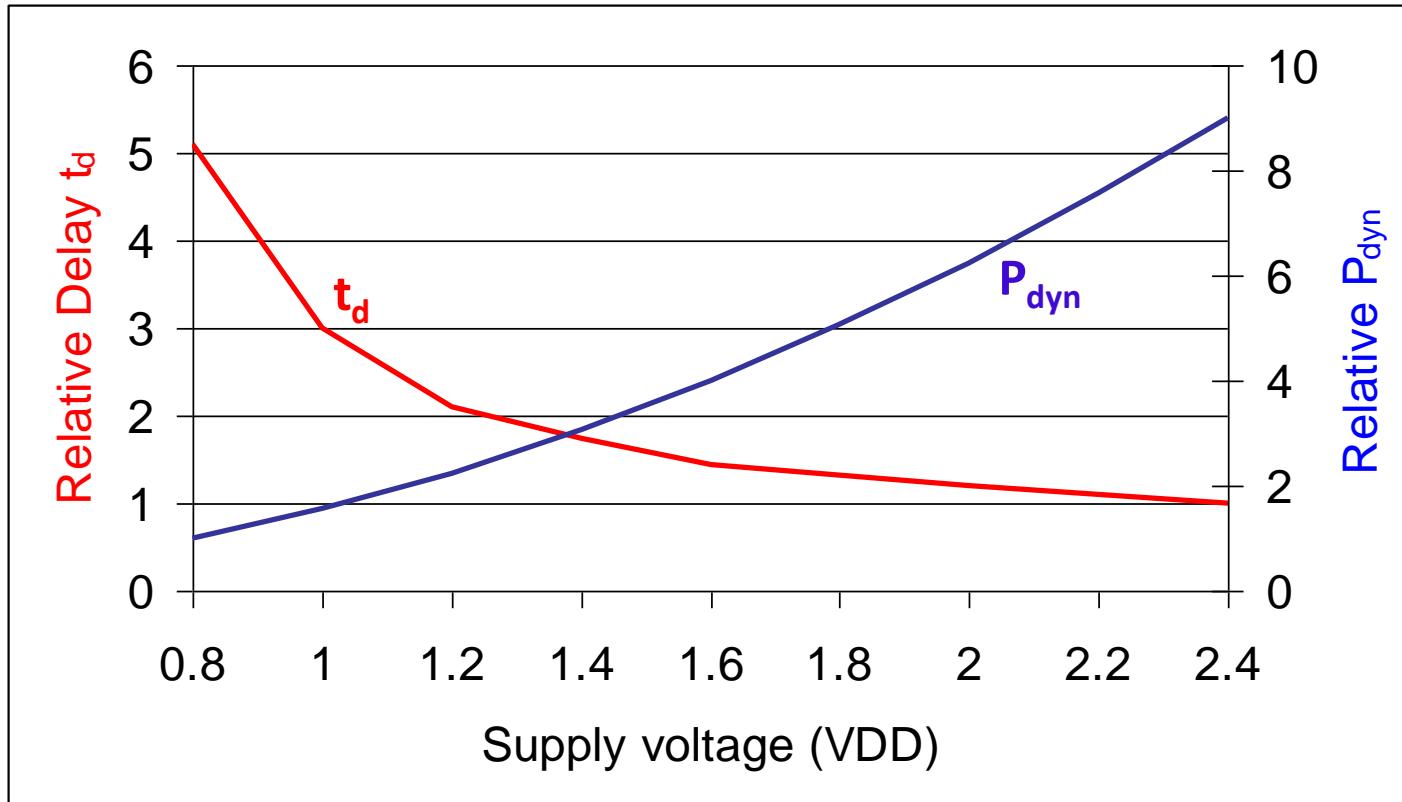


# Dynamic Power and Device Size

- Device Sizing (= changing gate width)
  - ➔ Affects input capacitance  $C_{in}$
  - ➔ Affects load capacitance  $C_{load}$
  - ➔ Affects dynamic power consumption  $P_{dyn}$
- Optimal fanout factor  $f$  for  $P_{dyn}$  is smaller than for performance (especially for large loads)
  - e.g., for  $C_{load}=20$ ,  $C_{in}=1$ 
    - ➔  $f_{circuit} = 20$
    - ➔  $f_{opt\_energy} = 3.53$
    - ➔  $f_{opt\_performance} = 4.47$
- For Low Power: avoid oversizing ( $f$  too big) beyond the optimal



# $V_{DD}$ versus Delay and Power



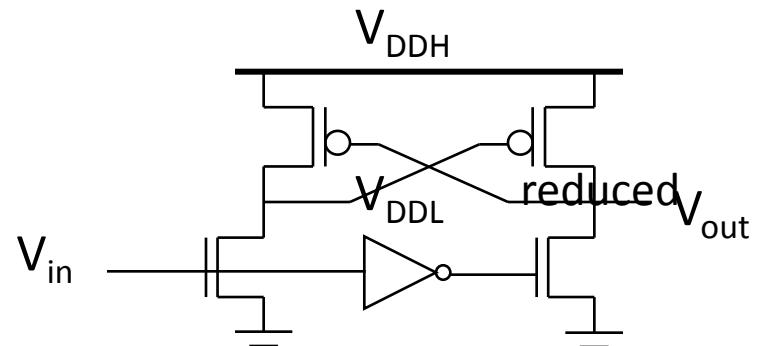
- Delay ( $t_d$ ) and dynamic power consumption ( $P_{dyn}$ ) are functions of  $V_{DD}$

# Multiple $V_{DD}$

- Main ideas:
  - Use of different supply voltages within the same design
  - High  $V_{DD}$  for critical parts (high performance needed)
  - Low  $V_{DD}$  for non-critical parts (only low performance demands)
- At design phase:
  - Determine **critical path(s)** (see upper next slide)
  - High  $V_{DD}$  for gates on those paths
  - Lower  $V_{DD}$  on the other gates (in non-critical paths)
  - For low  $V_{DD}$ : prefer gates that drive large capacitances (yields the largest energy benefits)
- Usually two different  $V_{DD}$  (but more are possible)

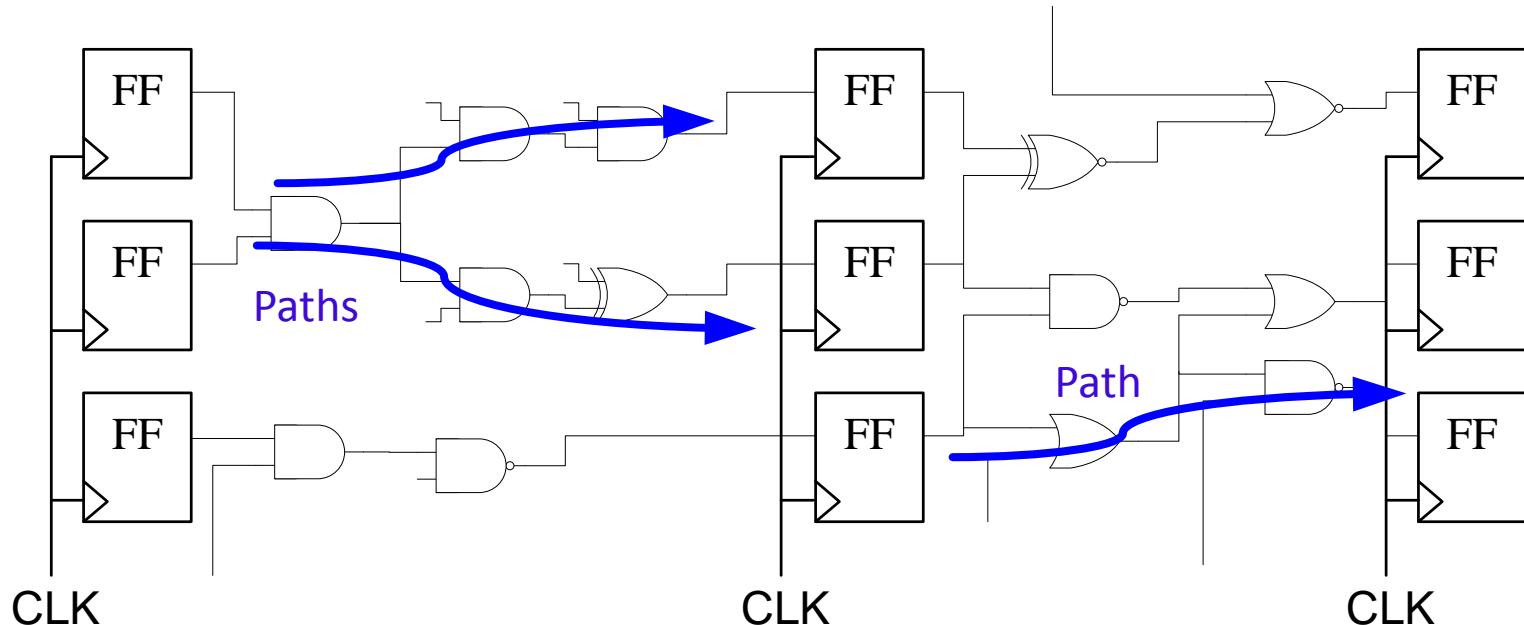
# Multiple $V_{DD}$ cont'd

- Level converters:
  - Necessary, when module at lower supply drives gate at higher supply (step-up)
  - If gate supplied with  $V_{DDL}$  drives a gate supplied with  $V_{DDH}$ 
    - then PMOS never turns off
  - Possible implementation:
    - Cross-coupled PMOS transistors
    - NMOS transistor operate on supply
  - No need of level converters for step-down change in voltage
  - Reducing of overhead:
    - Conversions at register boundaries
    - Embedding of inside flipflop

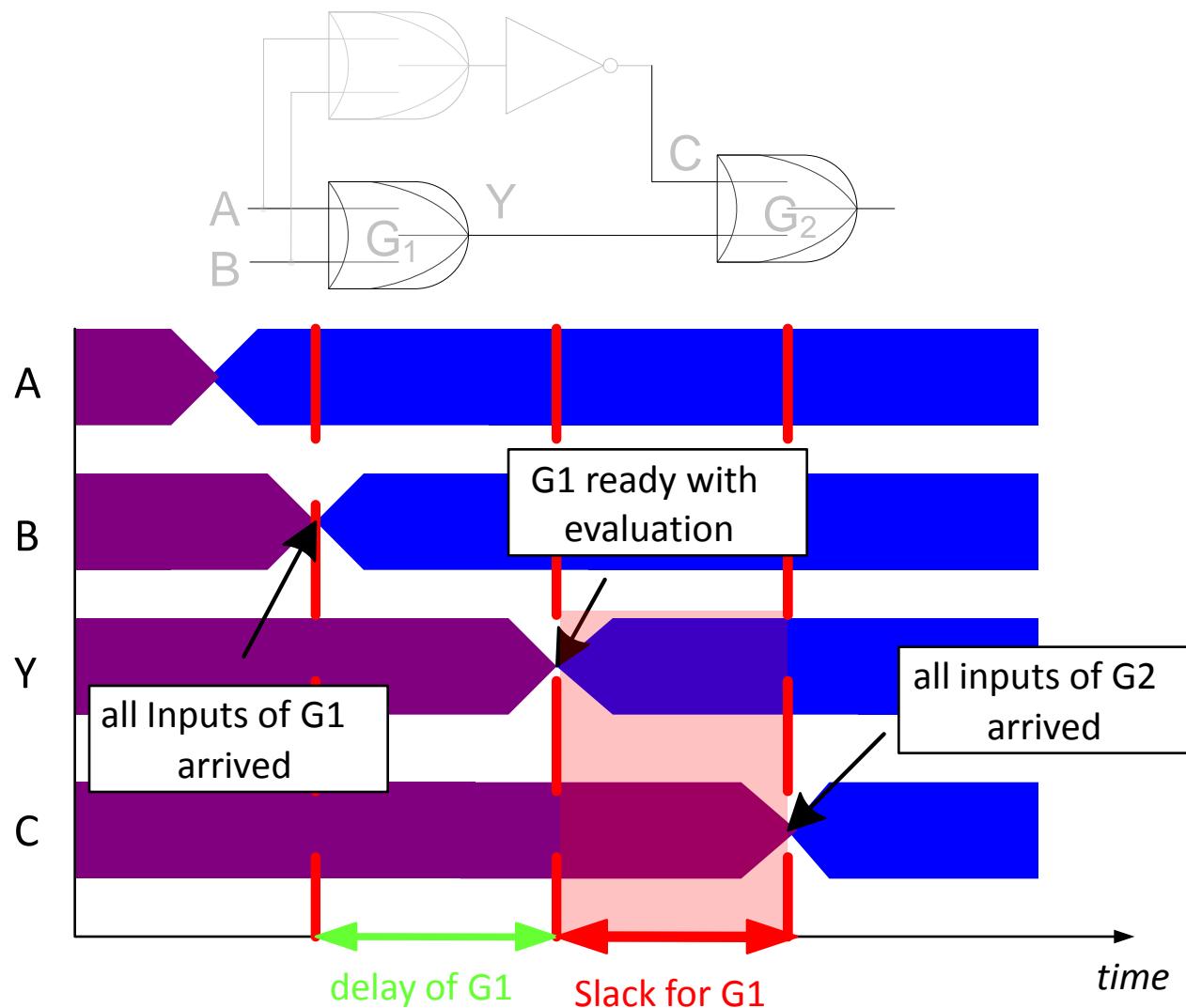


# Data Paths

- Data propagate through different data paths between registers (flipflops - FF)
- Paths mostly differ in propagation delay times
- Frequency of clock signal (CLK) depends on path with longest delay → **critical path**

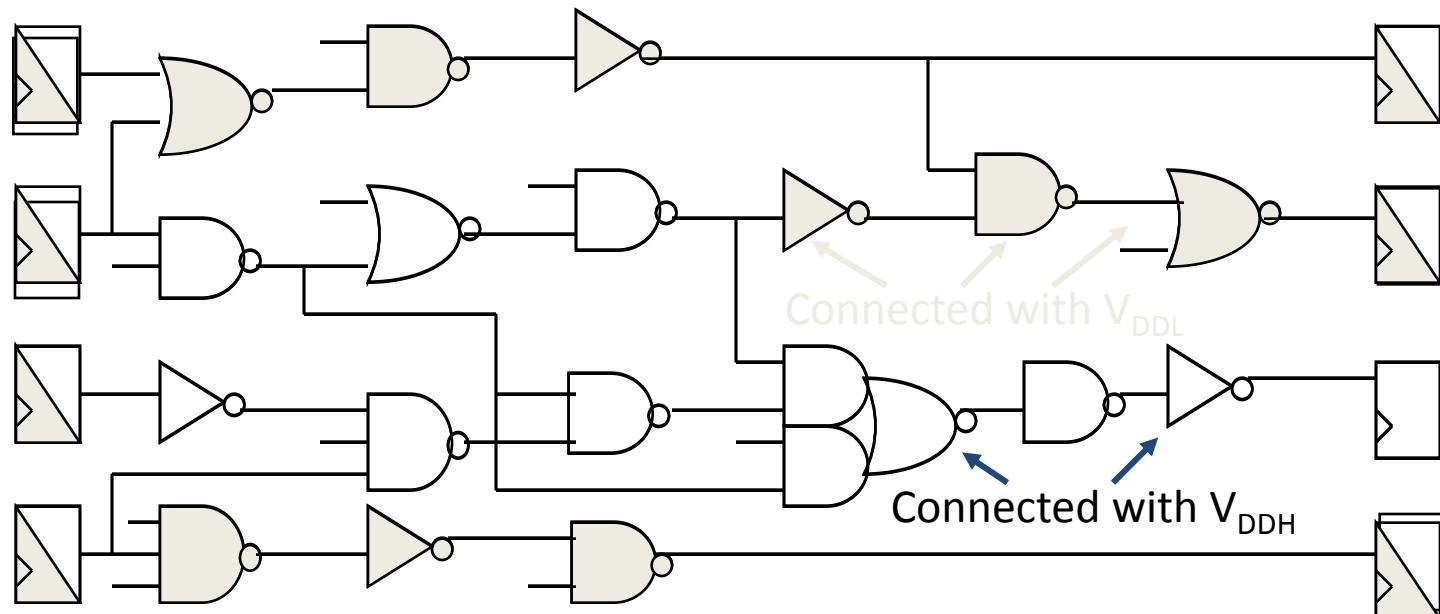


# Data Paths: Slack



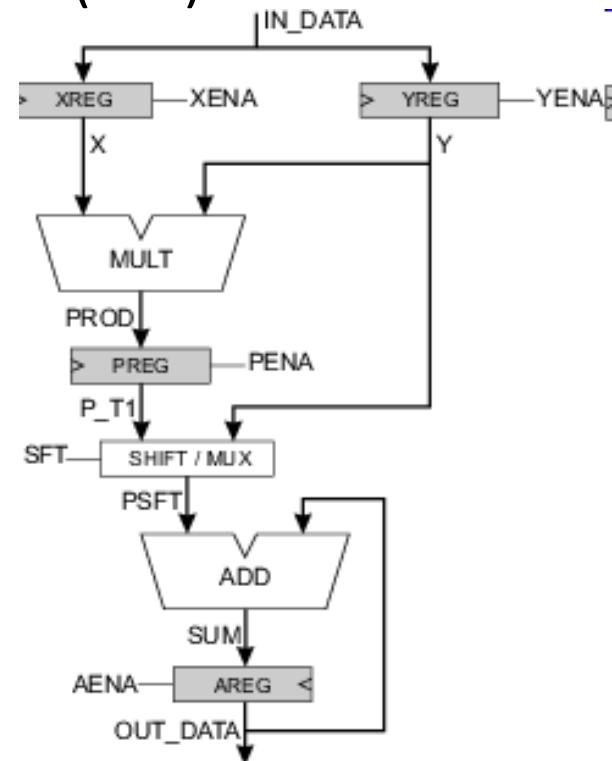
# Multiple $V_{DD}$ in Data Paths

- Minimum energy consumption when **all** logic paths are critical (same delay)
- Possible Algorithm: clustered voltage-scaling
  - Each path starts with  $V_{DDH}$  and switches to  $V_{DDL}$  (blue gates) when **slack** is available
  - Level conversion in flipflops at end of paths



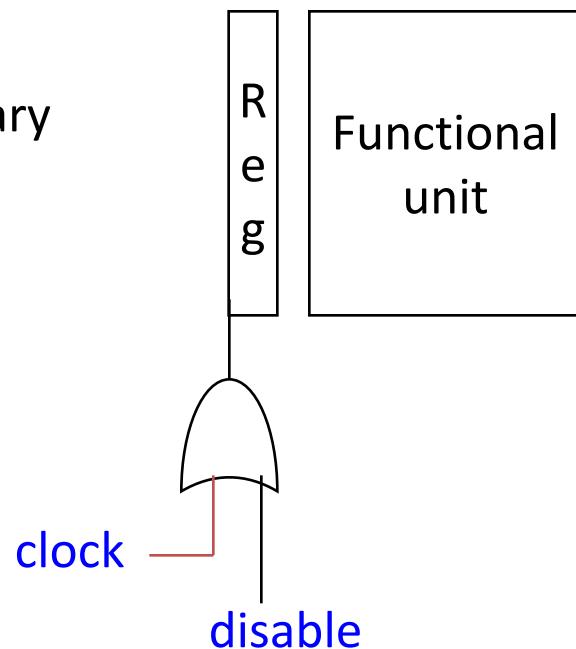
# Design Layer: Architecture Level

- Also known as Register transfer level (RTL)
- Base elements:
  - Register structures
  - Arithmetic logic units (ALU)
  - Memory elements
- Only behavior is described  
(no inner structure)



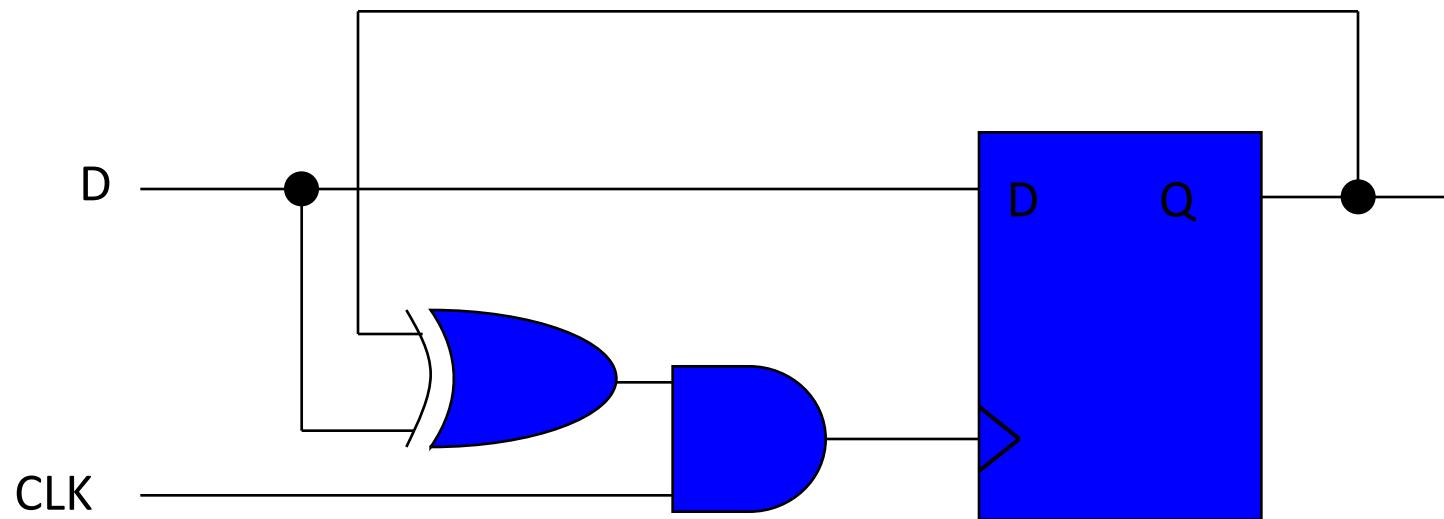
# Clock Gating

- Most popular method for power reduction of clock signals and functional units
- Gate off clock to idle functional units
- Logic for generation of **disable** signal necessary
  - 👎 Higher complexity of control logic
  - 👎 Higher power consumption
  - 👎 Critical timing critical for avoiding of clock glitches at OR gate output
  - 👎 Additional gate delay on clock signal



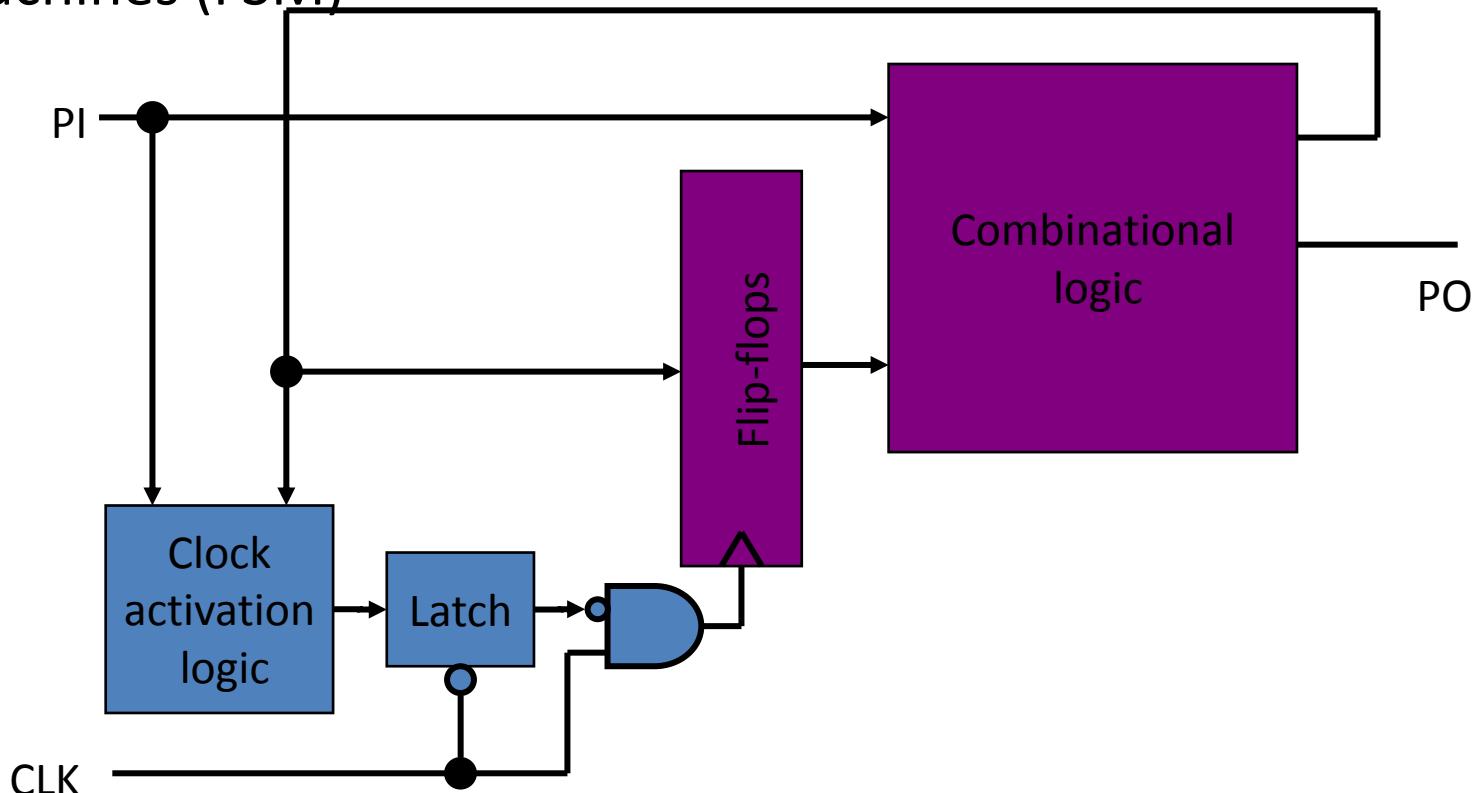
# Clock Gating cont'd

- Clock-Gating in Low-Power Flip-Flop

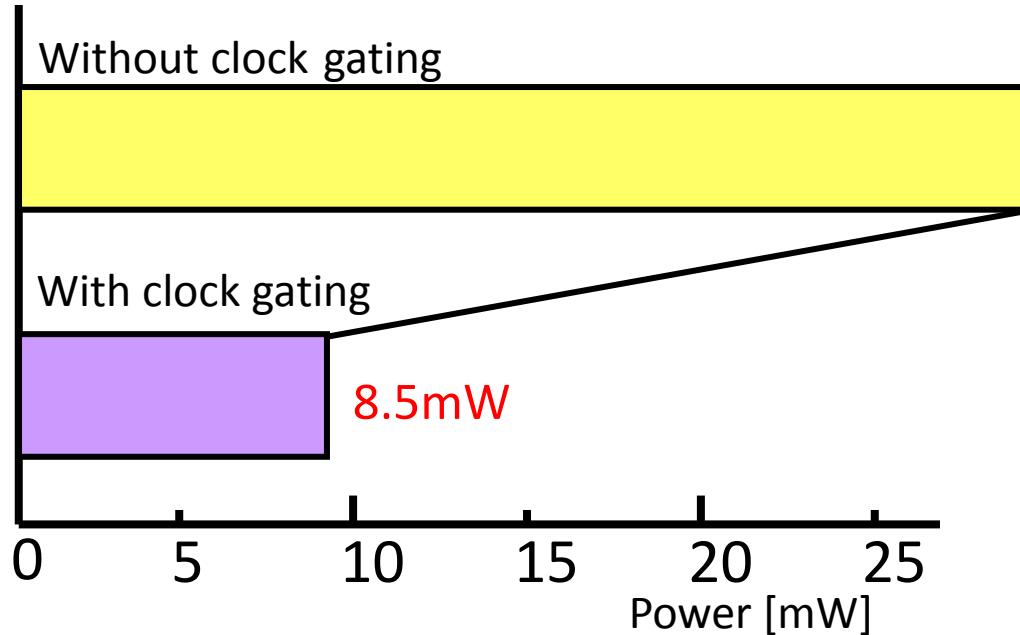


# Clock Gating cont'd

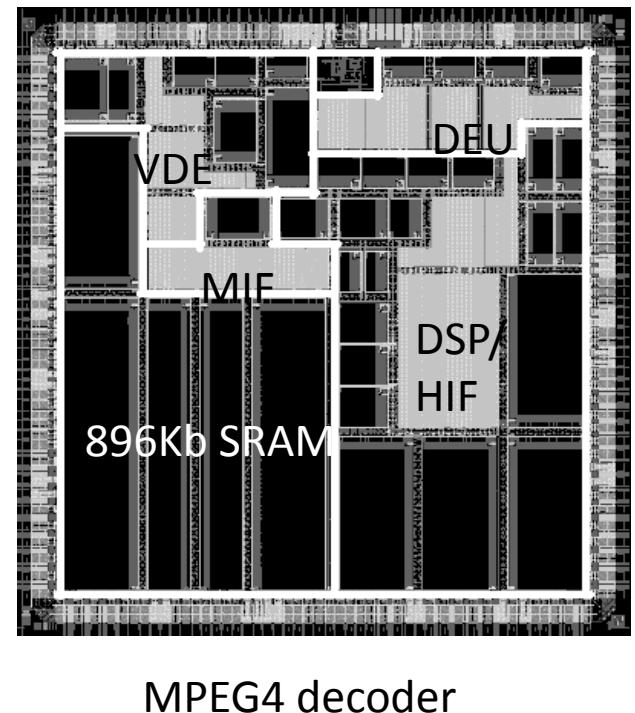
- Clock gating over consideration of state in Finite-State-Machines (FSM)



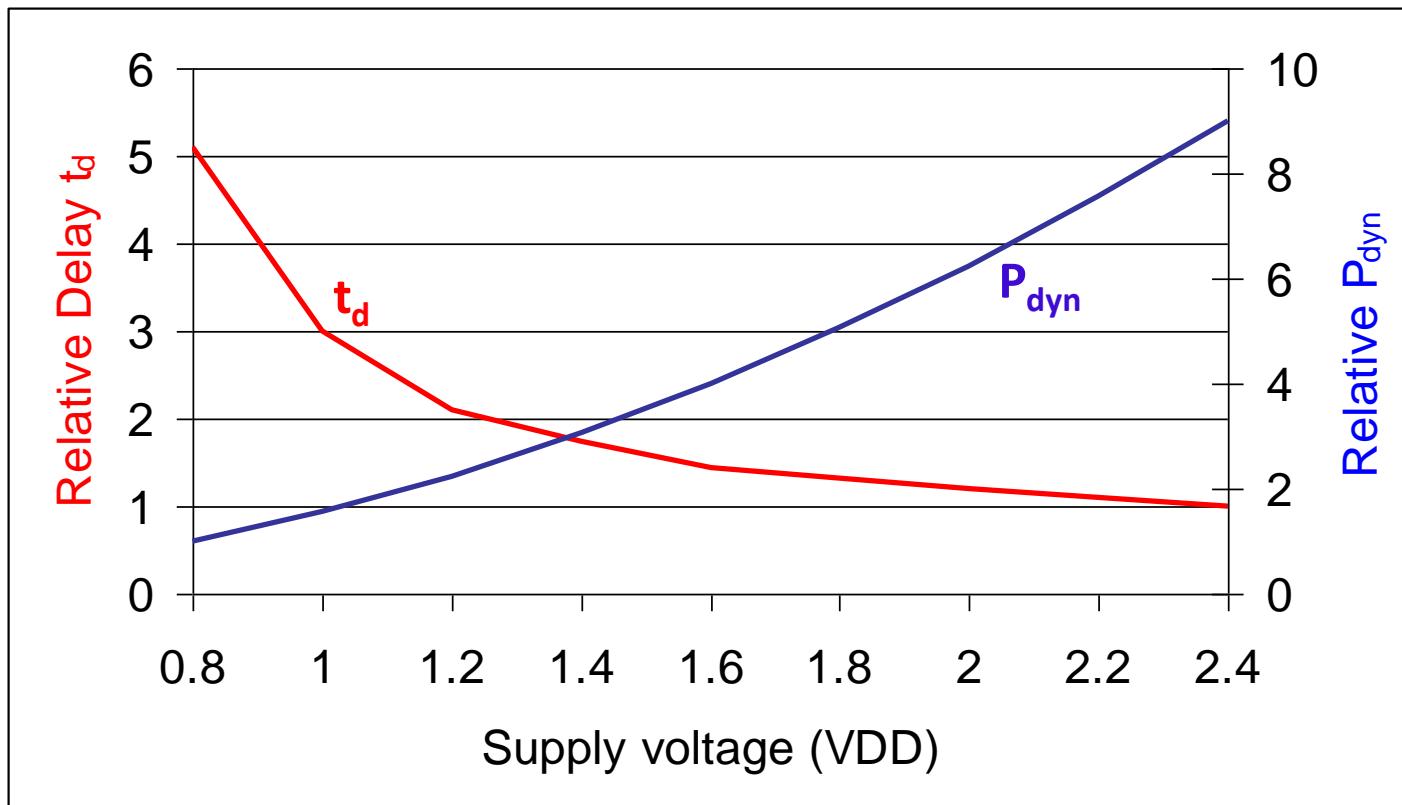
# Clock Gating: Example



- 90% of FlipFlops clock-gated
- 70% power reduction by clock-gating

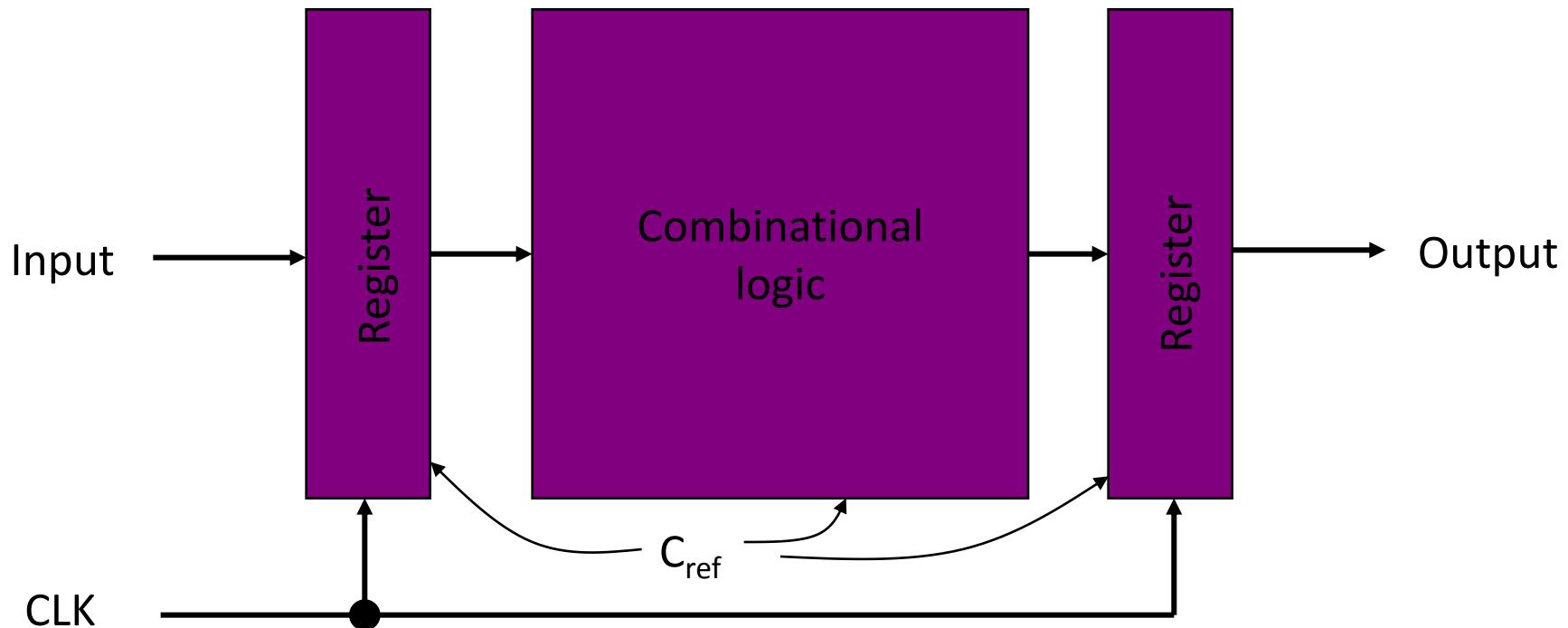


# Recap: $V_{DD}$ versus Delay and Power



Dynamic Power can be traded by delay

# A Reference Datapath



Supply voltage

Total capacitance switched per cycle

Clock frequency

Power consumption:

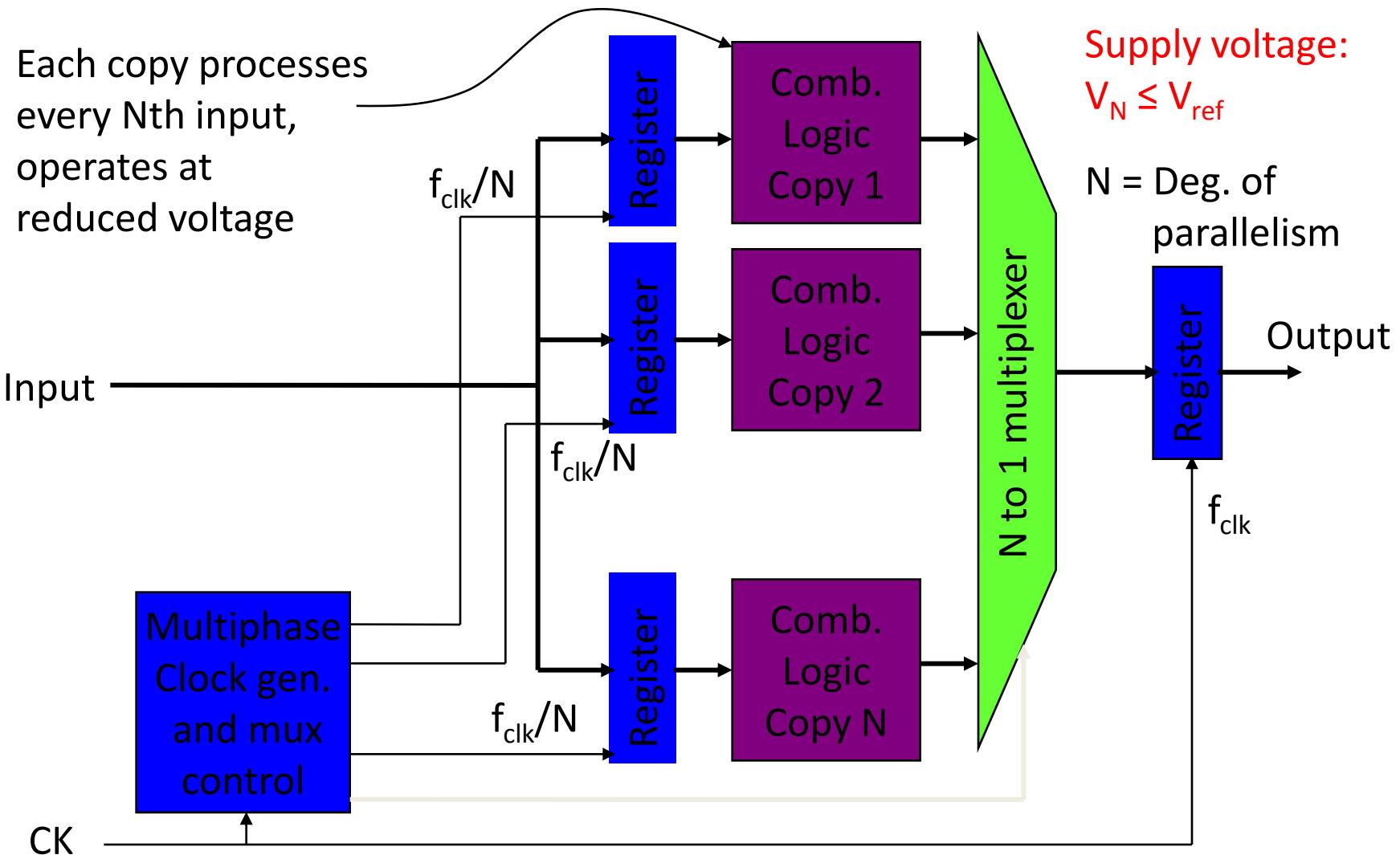
$$= V_{ref}$$

$$= C_{ref}$$

$$= f_{Clk}$$

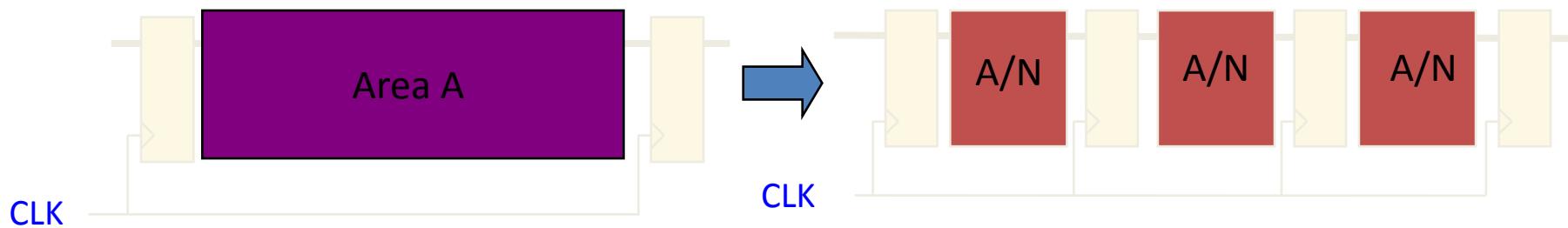
$$P_{ref} = C_{ref} V_{ref}^2 f_{clk}$$

# Parallel Architecture

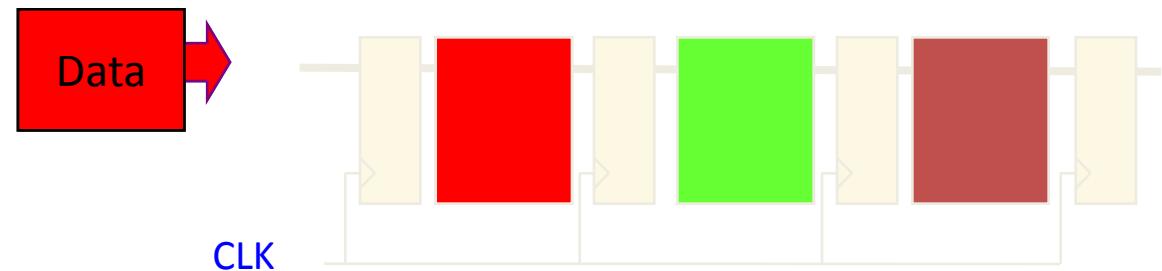


# Pipelined Architecture

- Reduces the propagation time of a block by factor N  
→ Voltage can be reduced at constant clock frequency
- Constant throughput

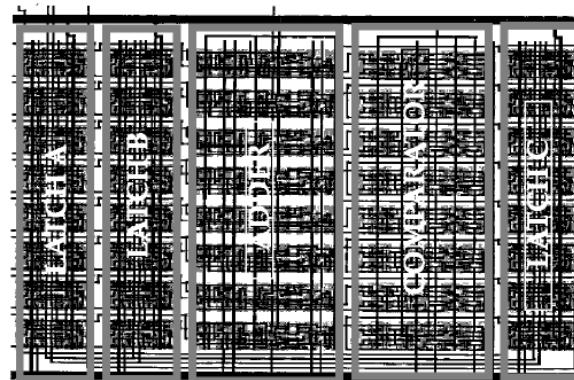
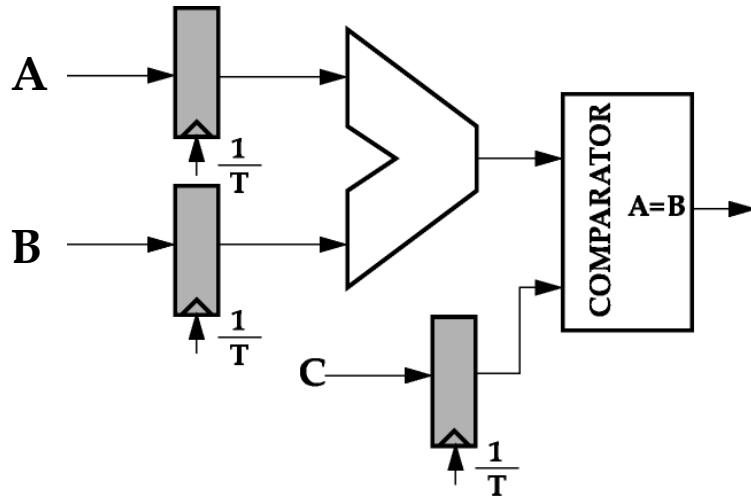


- Functionality:



# Parallel Architecture: Example

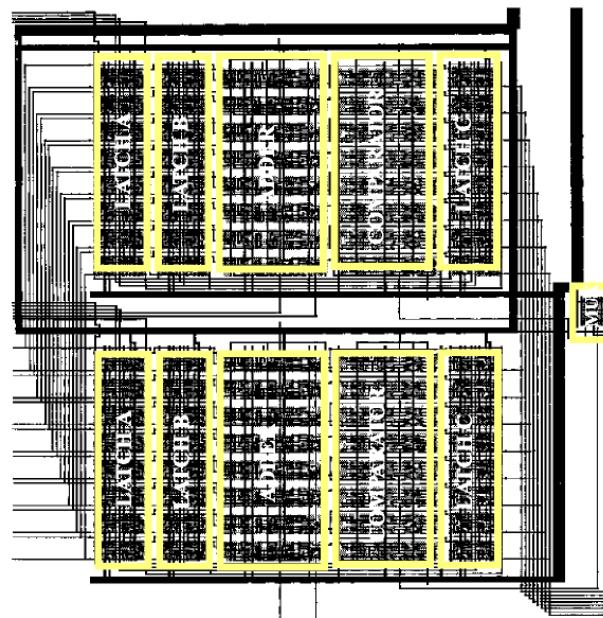
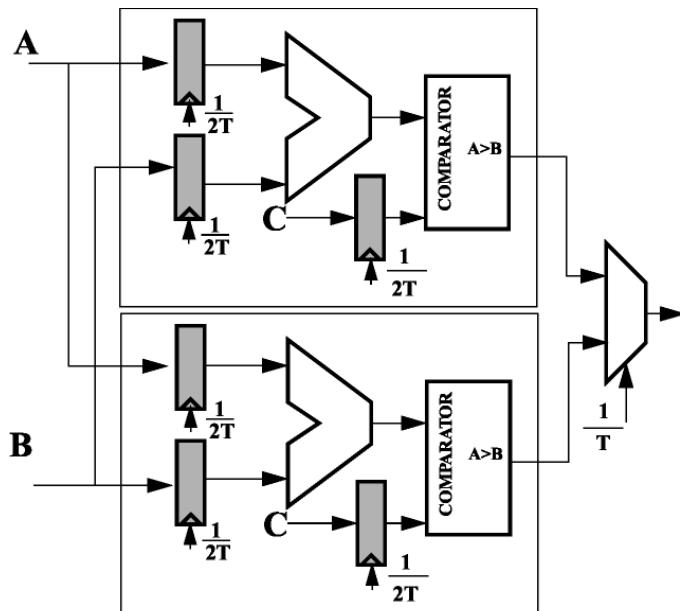
- Reference Data path (for example)



**Area =  $636 \times 833 \mu\text{m}^2$**

- Critical path delay  $T_{\text{adder}} + T_{\text{comparator}}$  ( $= 25 \text{ ns}$ )  
→  $f_{\text{ref}} = 40 \text{ MHz}$
- Total capacitance being switched =  $C_{\text{ref}}$
- $V_{\text{DD}} = V_{\text{ref}} = 5\text{V}$
- Power for reference datapath =  $P_{\text{ref}} = C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}}$

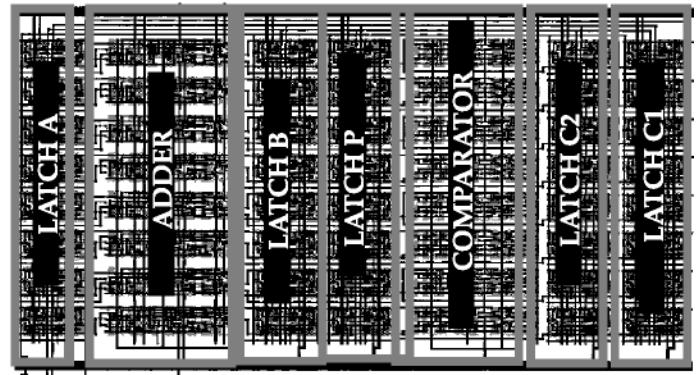
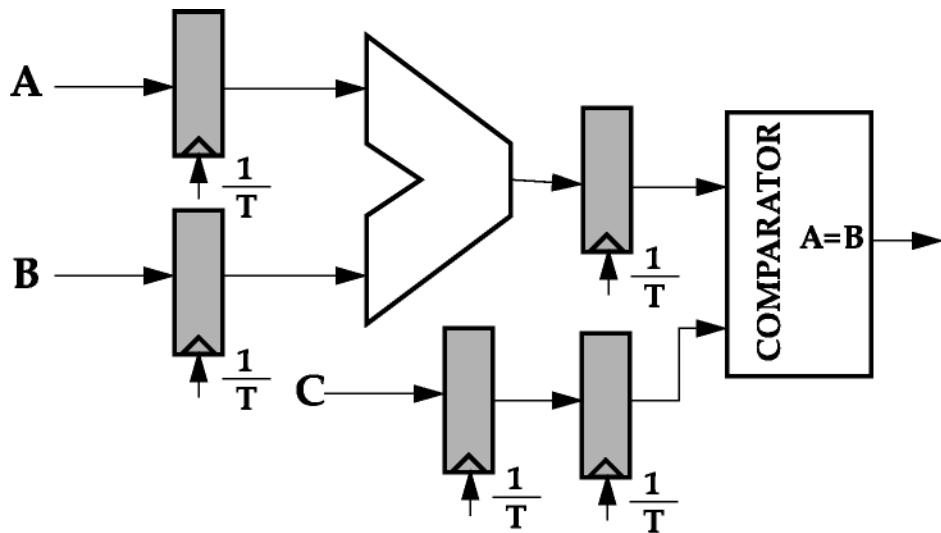
# Parallel Architecture: Example cont'd



$\text{Area} = 1476 \times 1219 \mu^2$

- The clock rate can be reduced by half with the same throughput  $f_{\text{par}} = f_{\text{ref}} / 2$
- $V_{\text{par}} = V_{\text{ref}} / 1.7$ ,  $C_{\text{par}} = 2.15 C_{\text{ref}}$
- $P_{\text{par}} = (2.15 C_{\text{ref}}) (V_{\text{ref}} / 1.7)^2 (f_{\text{ref}} / 2) = 0.36 P_{\text{ref}}$

# Pipelined Architecture: Example



**Area =  $640 \times 1081 \mu\text{m}^2$**

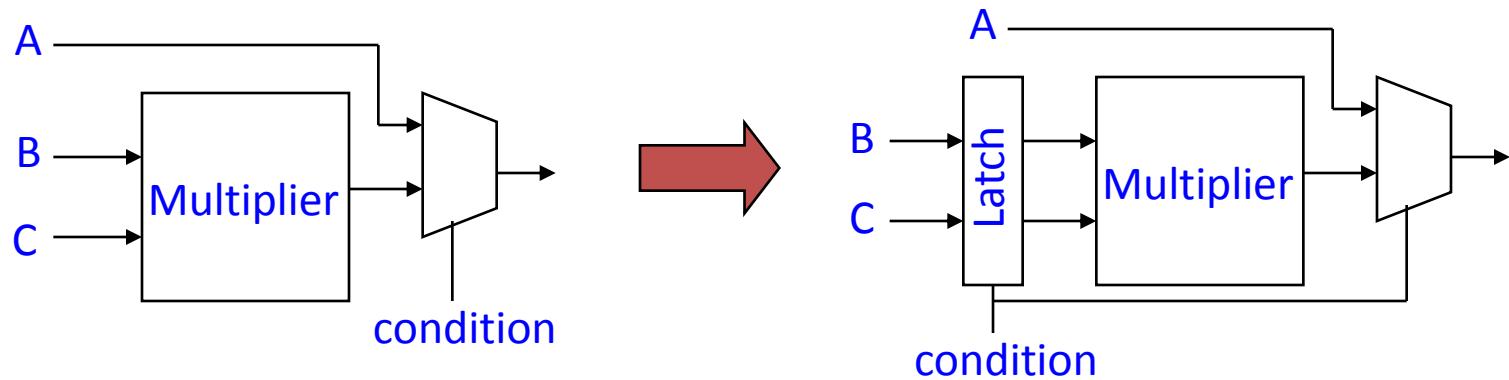
- $f_{\text{pipe}} = f_{\text{ref}}$ ,  $C_{\text{pipe}} = 1.1 C_{\text{ref}}$ ,  $V_{\text{pipe}} = V_{\text{ref}} / 1.7$
- Voltage can be dropped while maintaining the original throughput
- $P_{\text{pipe}} = C_{\text{pipe}} V_{\text{pipe}}^2 f_{\text{pipe}} = (1.1 C_{\text{ref}}) (V_{\text{ref}}/1.7)^2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

# Approximate Trend

	N-parallel proc.	N-stage pipeline proc.
Capacitance	$N^*C_{ref}$	$C_{ref}$
Voltage	$V_{ref}/N$	$V_{ref}/N$
Frequency	$f_{ref}/N$	$f_{ref}$
Dynamic Power	$C_{ref}V_{ref}^2f_{ref}/N^2$	$C_{ref}V_{ref}^2f_{ref}/N^2$
Chip area	N times	10-20% increase

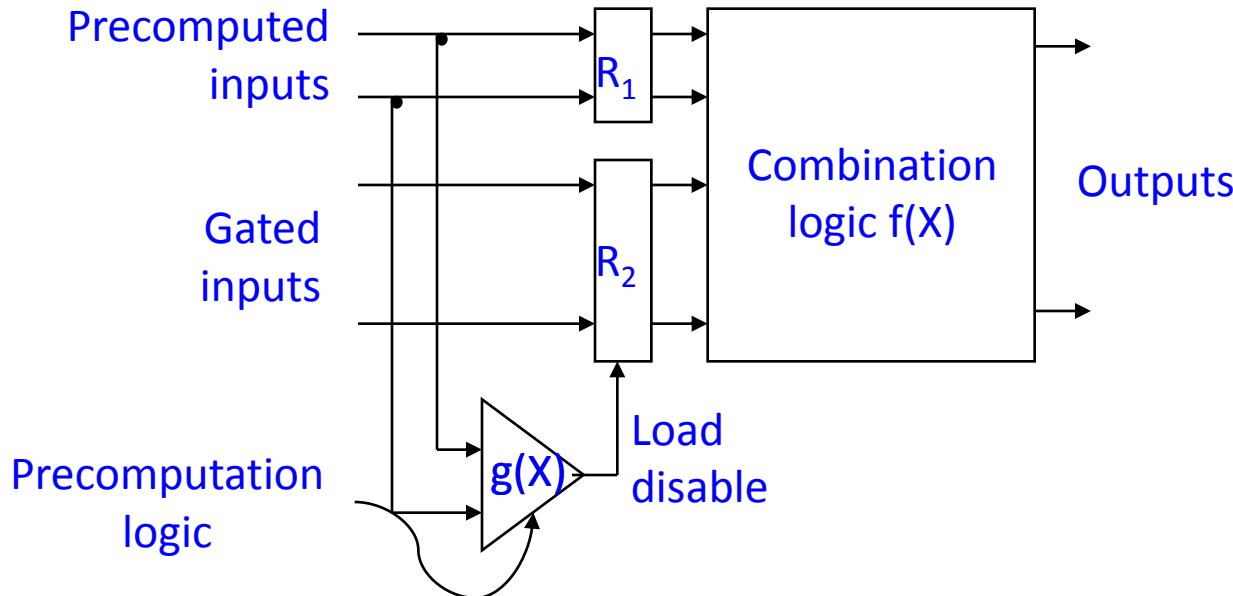
# Guarded Evaluation

- Reduction of switching activity by adding latches at inputs



- Latch preserves previous value of inputs to suppress activity
- Could also use AND gates to mask inputs to zero  
= forced zero

# Precomputation



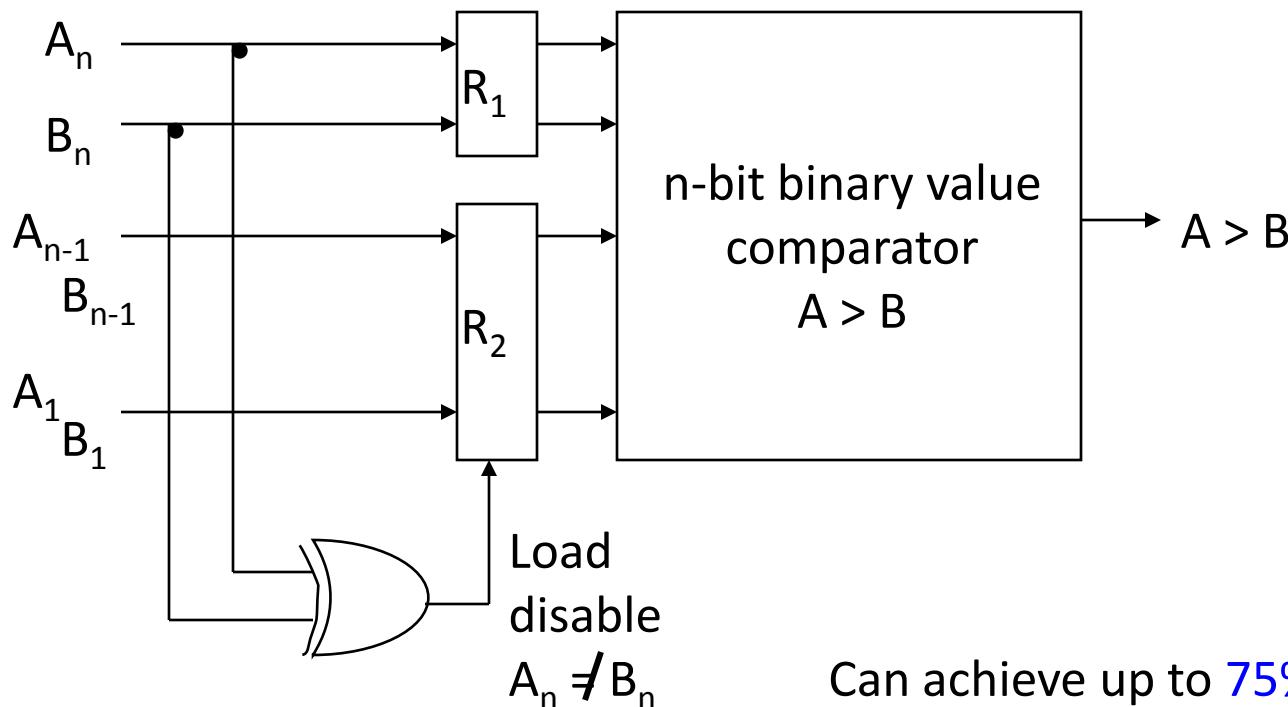
- Identify logical conditions at inputs that are invariant to the output
  - Since those inputs don't affect output, disable input transitions
  - Trade area for energy

# Precomputation: Design Issues

- Design steps
  1. Selection of precomputation architecture
  2. Determination of precomputed and gated inputs (Register R1 should be much smaller than R2)
  3. Search good implementation for  $g(X)$
  4. Evaluation of potential energy savings based on input statistics (if savings not sufficient go to step 2 or 3 and try again)
- Also works for multiple output functions where  $g(X)$  is the product of  $g_j(X)$  over all  $j$

# Precomputation: Example

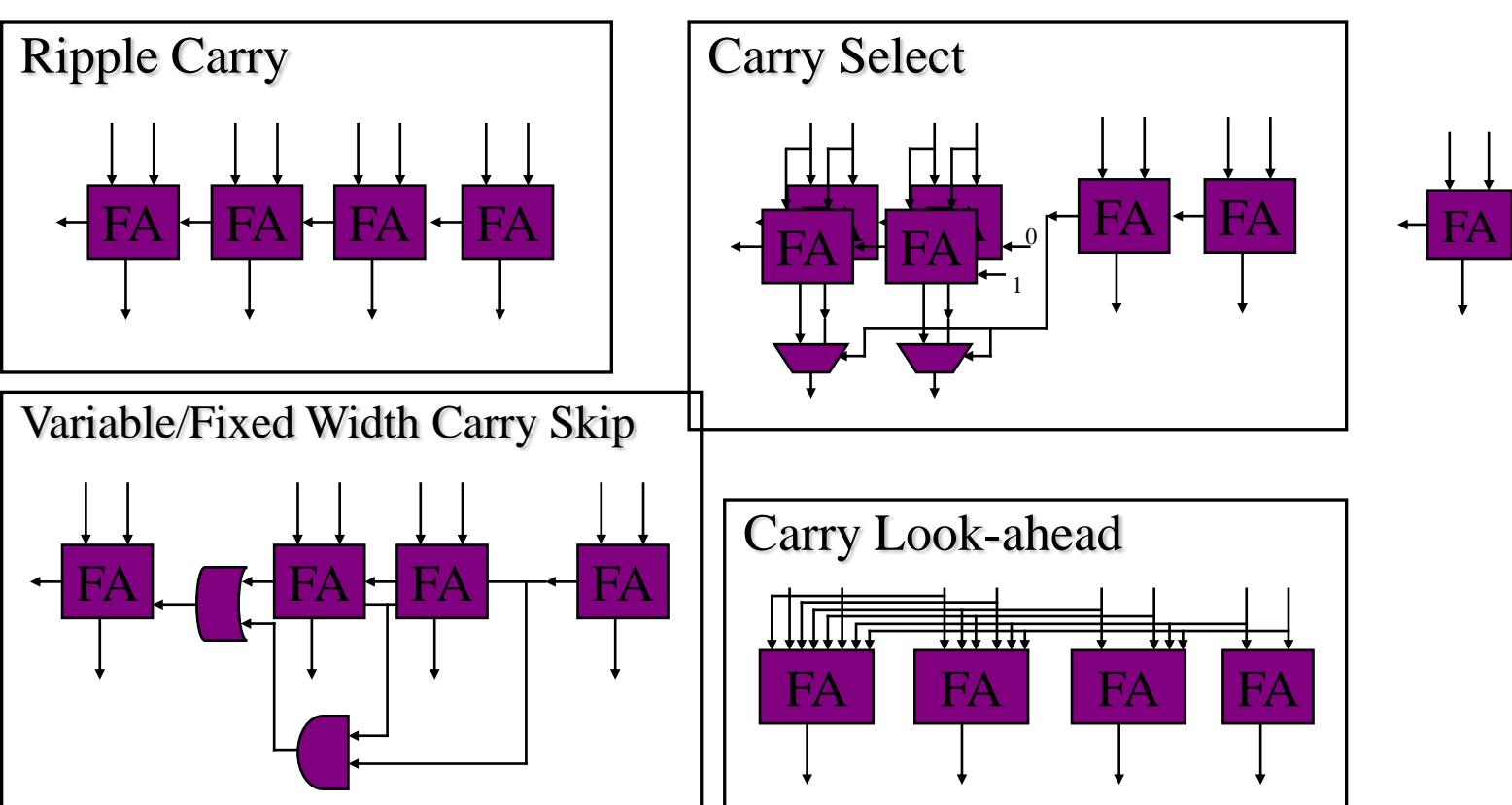
- Binary Comparator



Can achieve up to 75% power reduction with 3% area overhead and 1 to 5 additional gate delays in worst case path

# Adder Design

- Various algorithms exist to implement an integer adder
  - Ripple, select, skip (x2), Look-ahead, conditional-sum.
  - Each with its own characteristics of timing and power consumption.



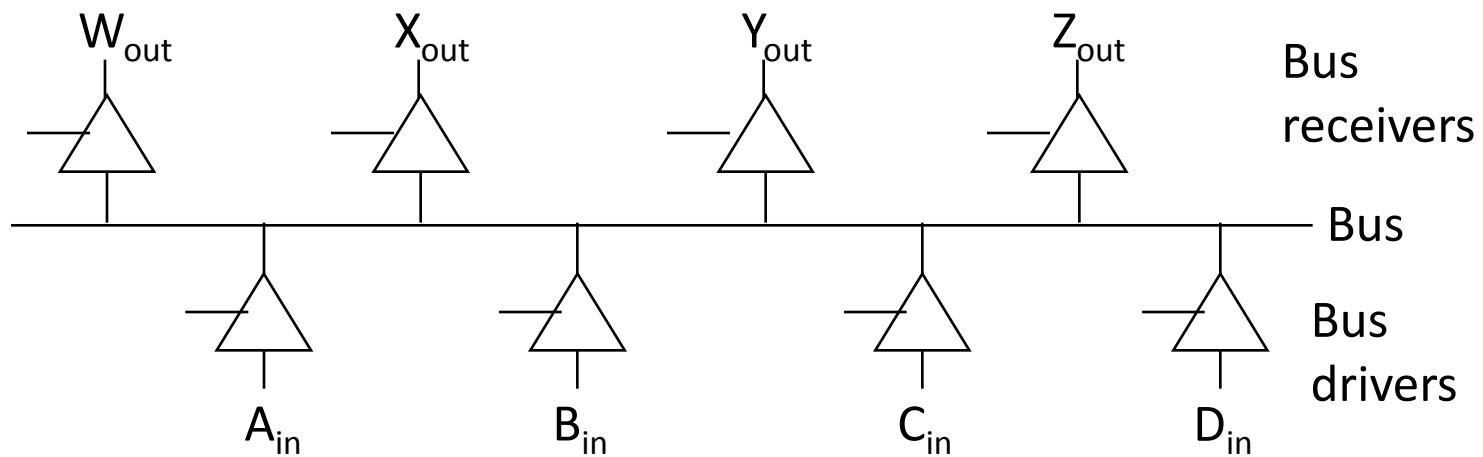
# Adder Design

	Energy (pJ )	Delay (nSec )
Ripple Carry	117	54.27
Constant Width Carry Skip	109	28.38
Variable Width Carry Skip	126	21.84
Carry Lookahead	171	17.13
Carry Select	216	19.56
Conditional Sum	304	20.05

- Adders differ in Energy and delay
- ➔ Different adders for different applications
- ➔ Also true for other units (multiplier, counter, ...)

# Bus Power

- Buses are significant source of power dissipation
  - 50% of dynamic power for interconnect switching (Magen, SLIP 04)
  - MIT Raw processor's on-chip network consumes 36% of total chip power (Wang et al. 2003)
- Caused by:
  - High switching activities
  - Large capacitive loading

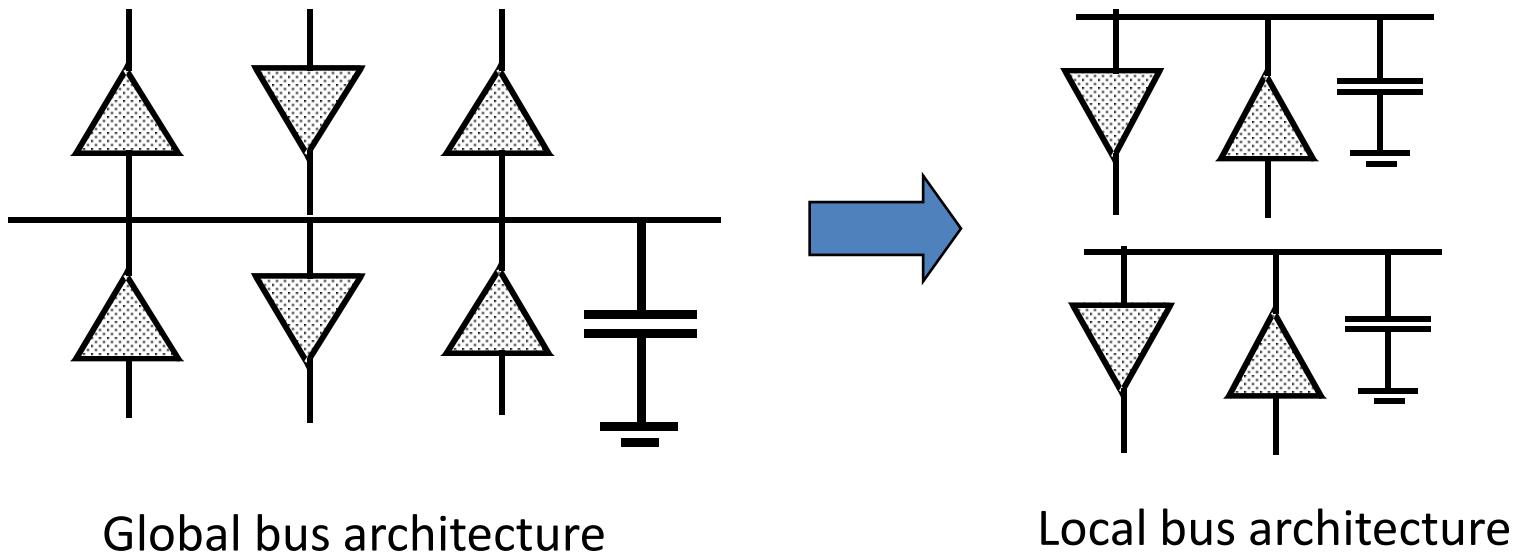


# Bus Power Reduction

- For an n-bit bus:  $P_{bus} = n * \alpha f_{Clk} C_{load} V_{DD}^2$
- Alternative bus structures
  - Segmented buses (lower  $C_{load}$ )
  - Charge recovery buses
  - Bus multiplexing (lower  $f_{Clk}$  possible)
- Minimizing bus traffic (n)
  - Code compression
  - Instruction loop buffers
- Minimization of bit switching activity ( $f_{clk}$ ) by data encoding
- Minimize voltage swing ( $V_{DD}^2$ ) using differential signaling

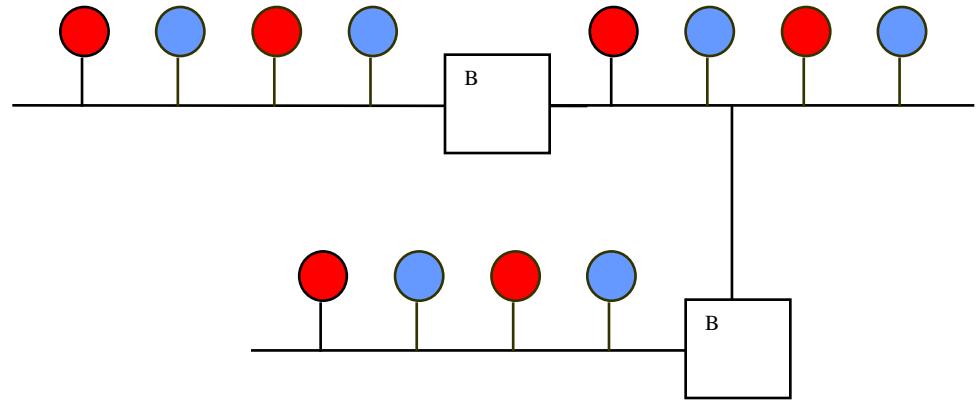
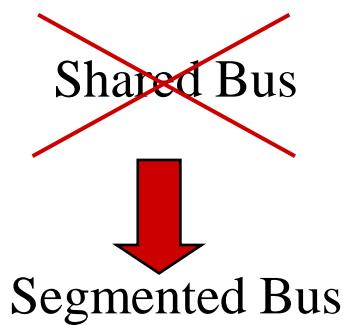
# Reducing Shared Resources

- Shared resources incur switching overhead
- Local bus structures reduce overhead



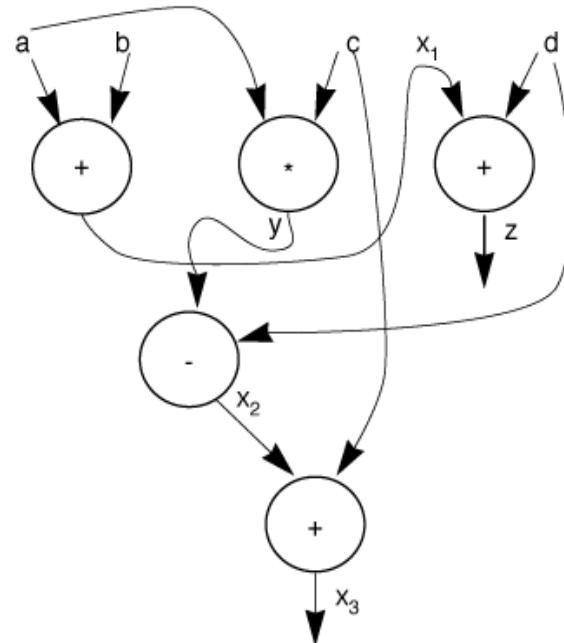
# Reducing Shared Resources cont'd

- Bus segmentation
  - Another way to reduce shared buses
  - Control of bus segment by controller blocks (B)



# Design Layer: Algorithm Level

- Base elements:
  - Functions
  - Procedures
  - Processes
  - Control structures
- Description of design behavior

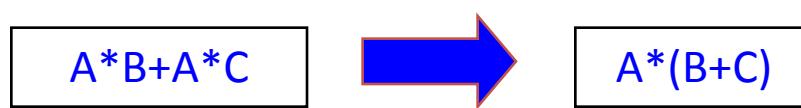


# Coding styles

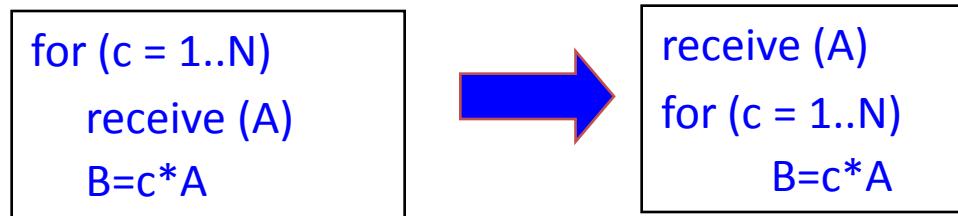
- Use processor-specific instruction style:
  - Variable types
  - Function calls style
  - Conditionalized instructions (for ARM)
- Follow general guidelines for software coding
  - Use table look-up instead of conditionals
  - Make local copies of global variables so that they can be assigned to registers
  - Avoid multiple memory look-ups with pointer chains

# Source-code Transformations

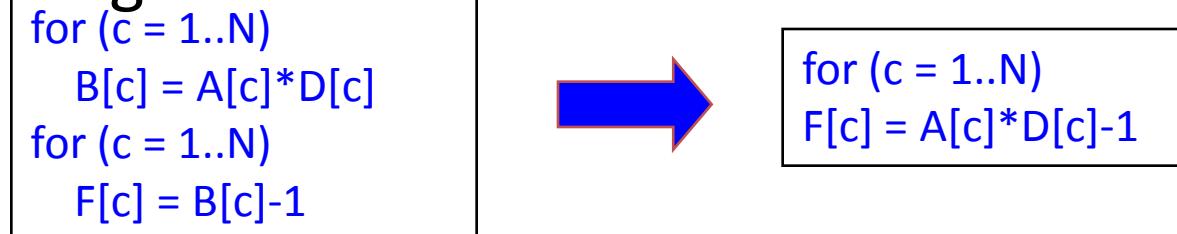
- Minimize power-consuming activity:
  - Computation



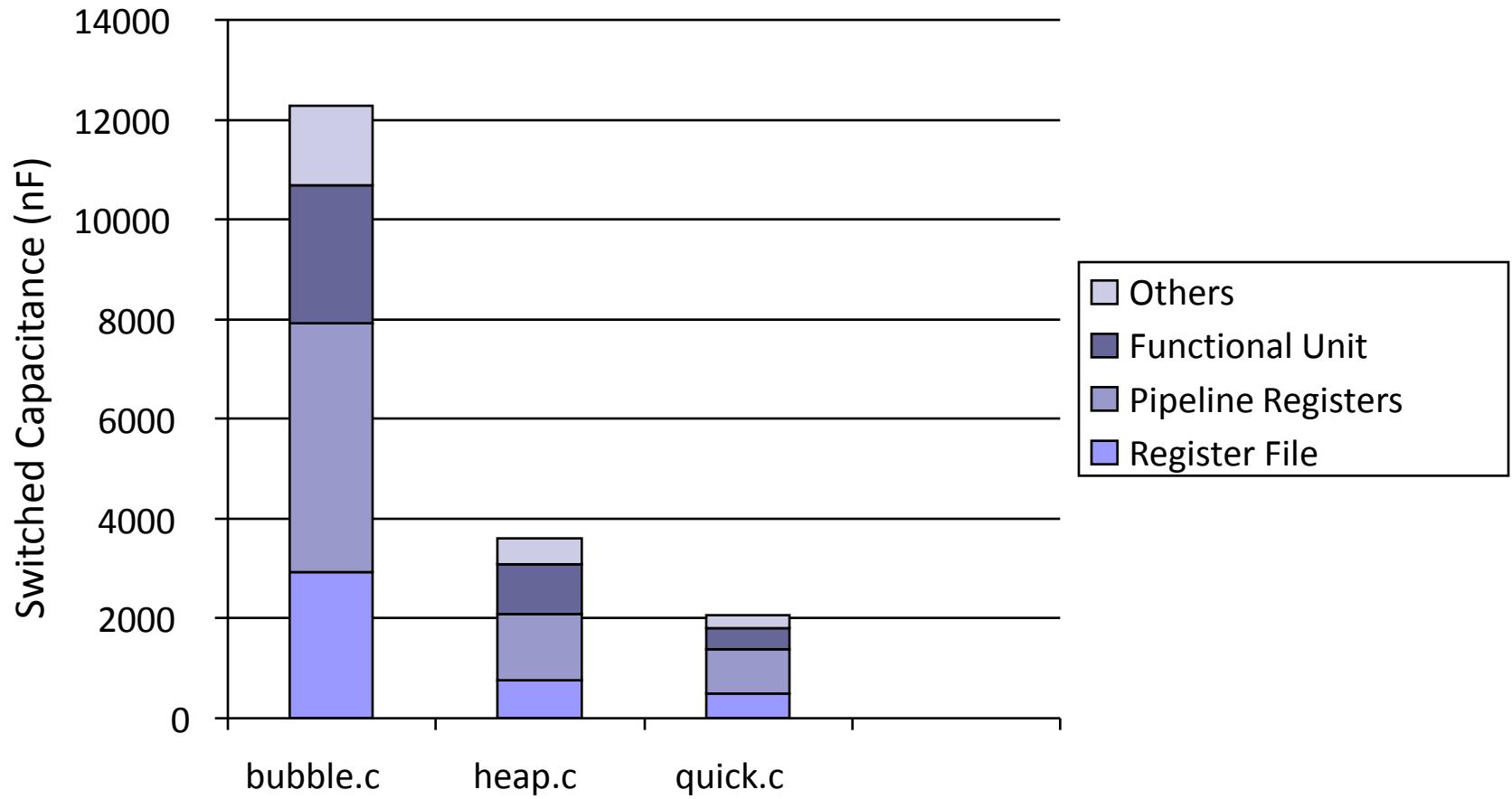
- Communication



- Storage



# Datapath Energy Consumption



→ Algorithms can differ in power dissipation

# Adaptive Dynamic Voltage Scaling (DVS)

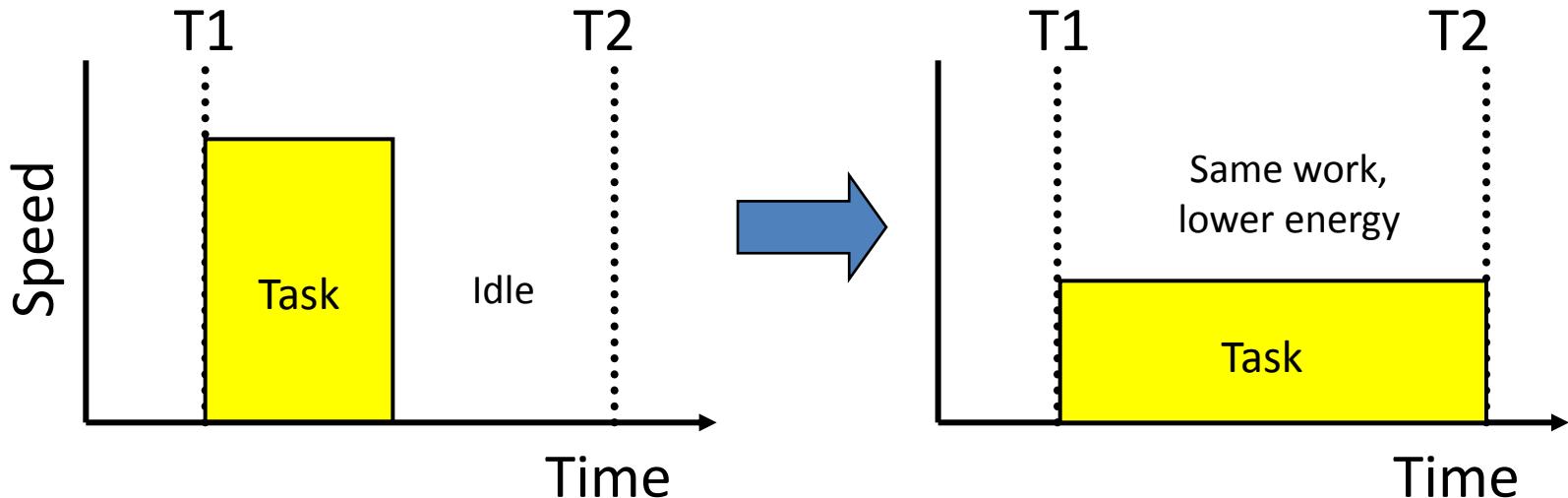
- Slow down processor to fill idle time
- More Delay → lower operational voltage



- Runtime Scheduler determines processor speed and selects appropriate voltage
- Transitions delay for frequencies  $\sim 150\mu\text{s}$
- Potential to realize 10x energy savings

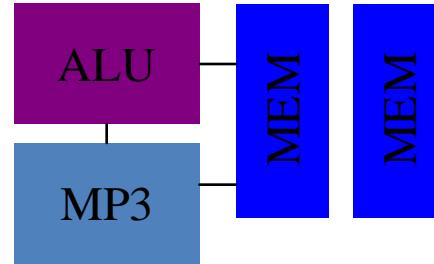
# Adaptive DVS: Example

- Task with 100 ms deadline, requires 50 ms CPU time at full speed
  - Normal system gives 50 ms computation, 50 ms idle/stopped time
  - Half speed/voltage system gives 100 ms computation, 0 ms idle
  - Same number of CPU cycles but:  $E = C (V_{DD}/2)^2 = E_{ref} / 4$
  - Dynamic Voltage Scaling adapts voltage to workload



# Design Layer: System Level

- Basic Elements:
  - Complex modules
  - Processors
  - Calculation and control units
  - Sensors

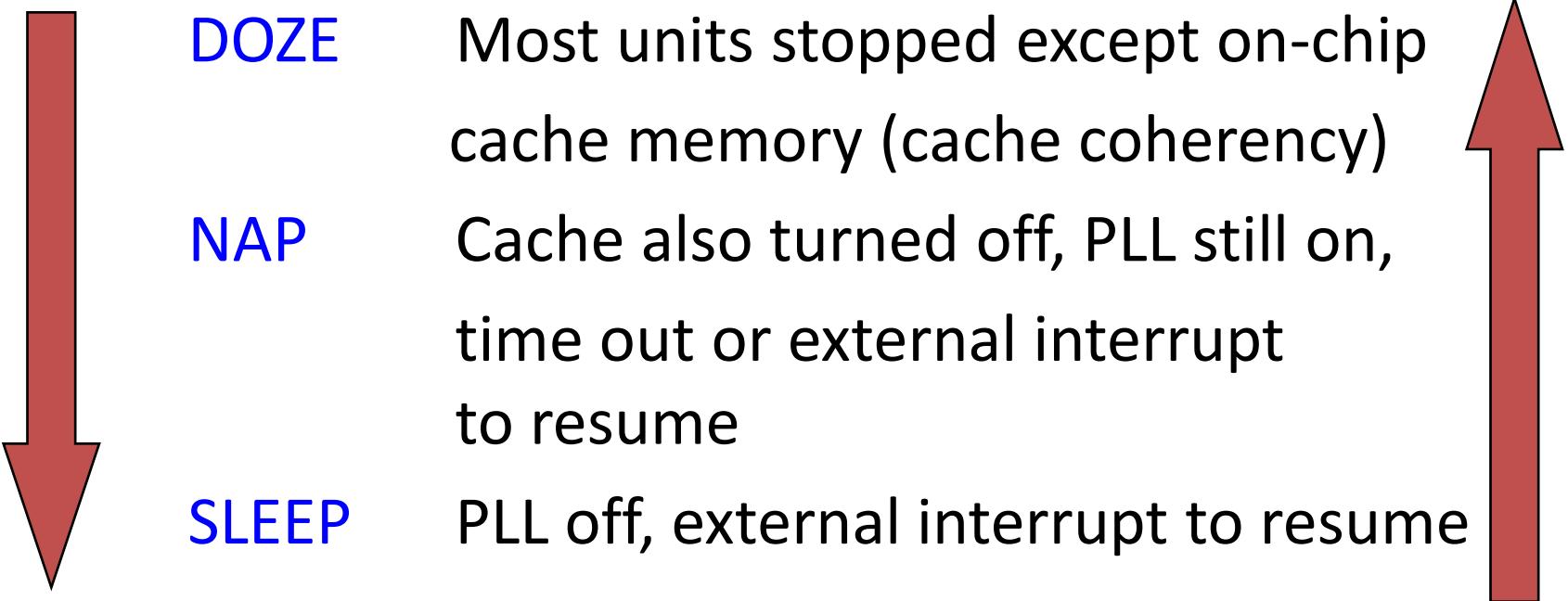


# Dynamic Power Management

- Systems are:
  - Designed to deliver peak performance, but ...
  - Not needing peak performance most of the time
- Components are idle sometimes
- Dynamic power management (DPM):
  - Puts idle components in low-power non-operational states when idle
- Power manager:
  - Observes and controls the system
  - Power consumption of power manager is negligible

# Processor Sleep Modes

- Software power control - power management



Deeper sleep mode consumes less **power**

Deeper sleep mode requires more **latency** to resume

# Processor Sleep Modes: Example

- PowerPC sleep modes

Mode	66Mhz	80Mhz
No power mgmt	2.18W	2.54W
Dynamic power mgmt	1.89W	2.20W
DOZE	307mW	366mW
NAP	113mW	135mW
SLEEP	89mW	105mW
SLEEP without PLL	18mW	19mW
SLEEP without clock	2mW	2mW

10 cycles to wake up from SLEEP

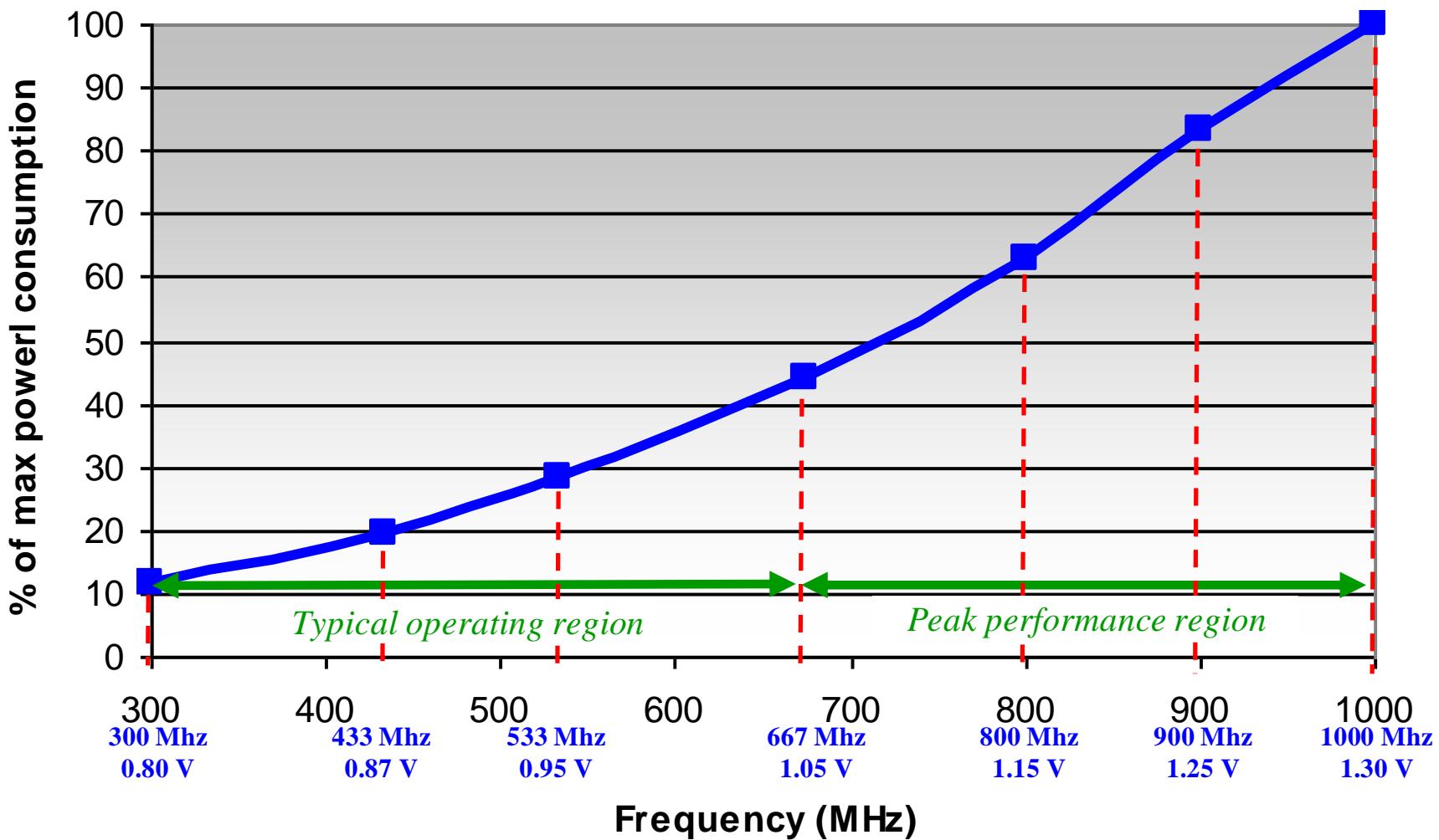
100us to wake up from SLEEP+



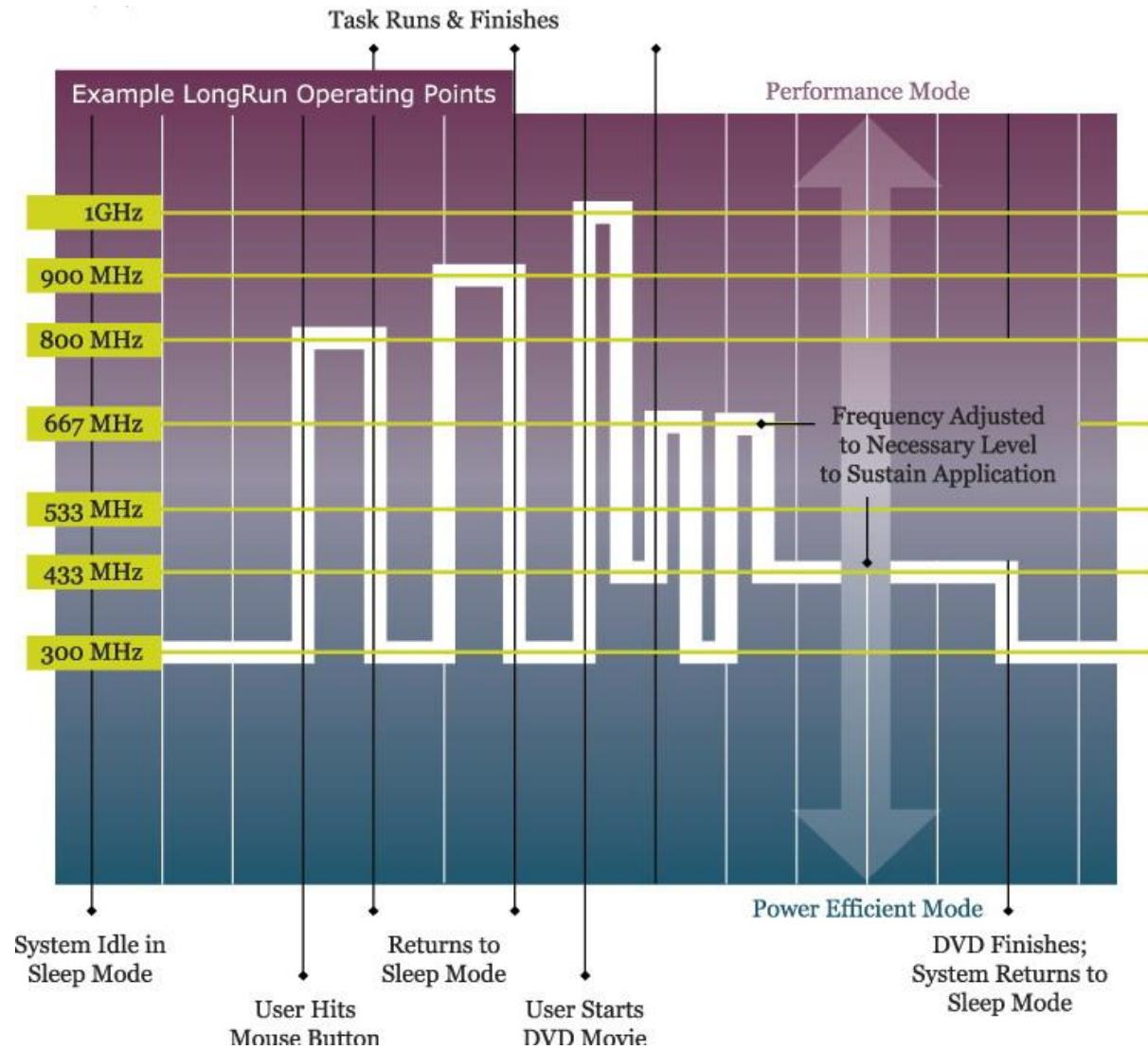
# Transmeta LongRun

- Applies adaptive DVS
  - LongRun policies:
    - Detection of different workload scenarios
    - Based on runtime performance information
  - After detection → accordingly adaptation of:
    - Processor supply voltage
    - Processor frequency
    - Clock frequency always within limits required by supply voltage to avoid clock skew problems
  - Use of core frequency/voltage hard coded operating points
- Best trade-off between performance and power possible

# Transmeta LongRun cont'd

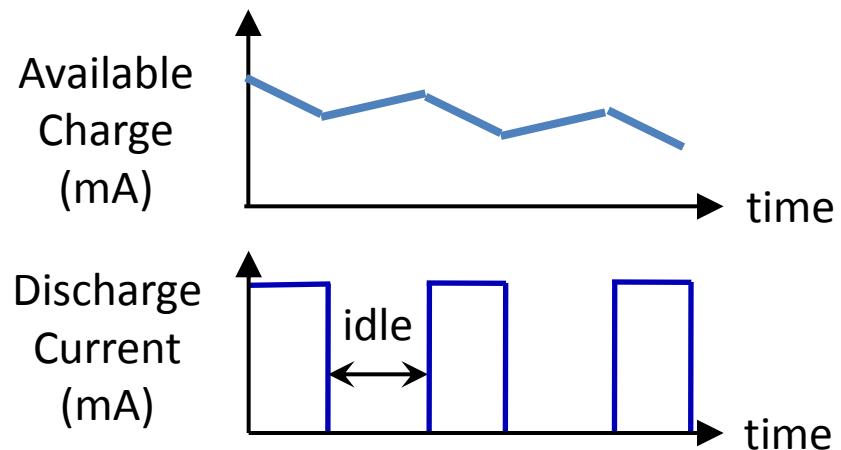
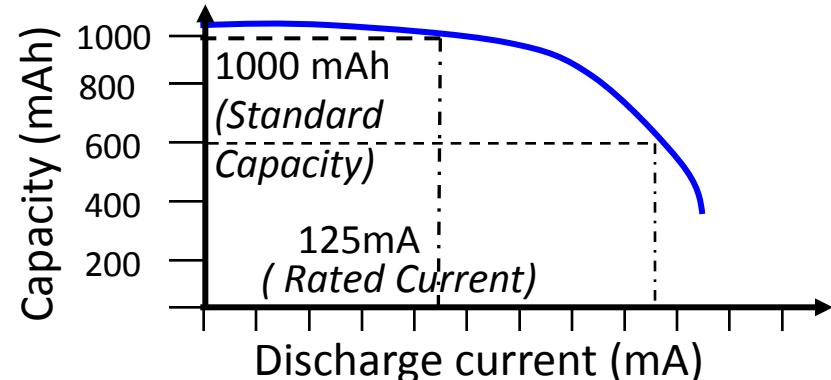


# Transmeta LongRun: Example



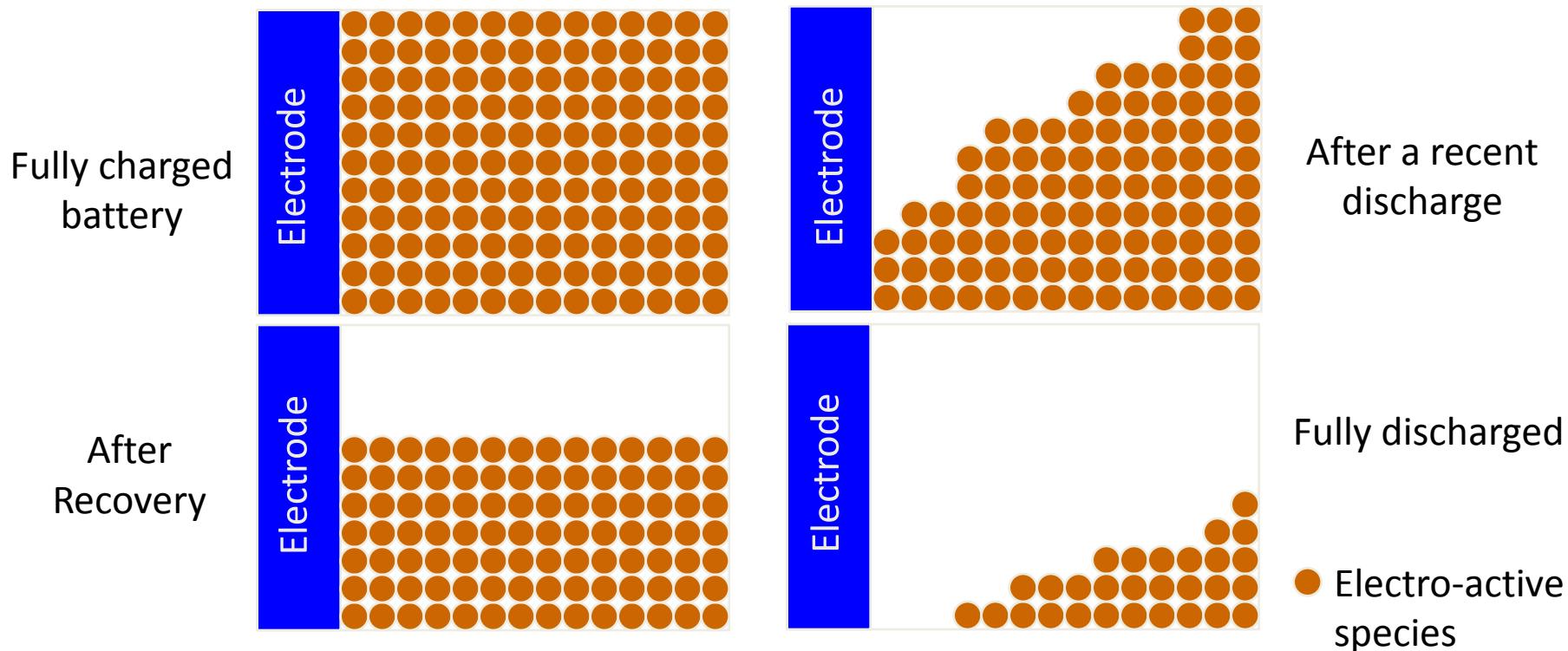
# Battery aware design

- Non-linear effects influence life time of batteries
- “Rate Capacity”
  - If discharging currents higher than allowed  
→ real capacity goes under nominal capacity
- “Battery Recovery”
  - Pulsed discharge increases nominal capacity
  - Based on recovery times
  - (as long there is no rate capacity effect)



# Battery aware design cont'd

Diffusion Model from - Rakhmatov, Vrudula et al.

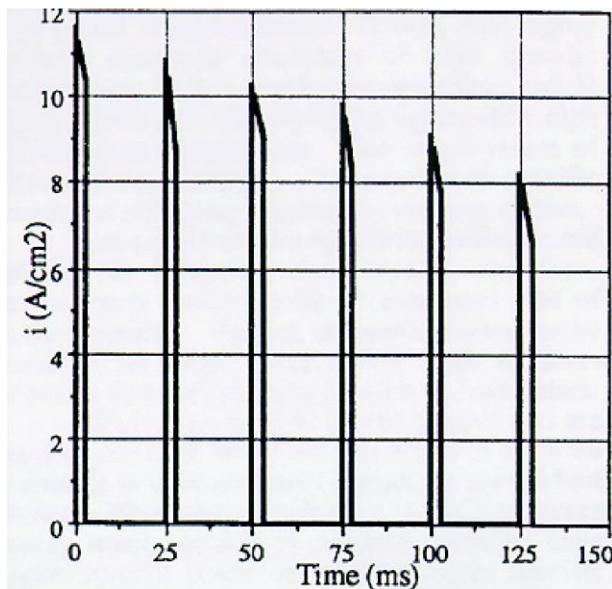


- Analytically very sound but computationally intensive
- Cannot be used for online scheduling decisions.

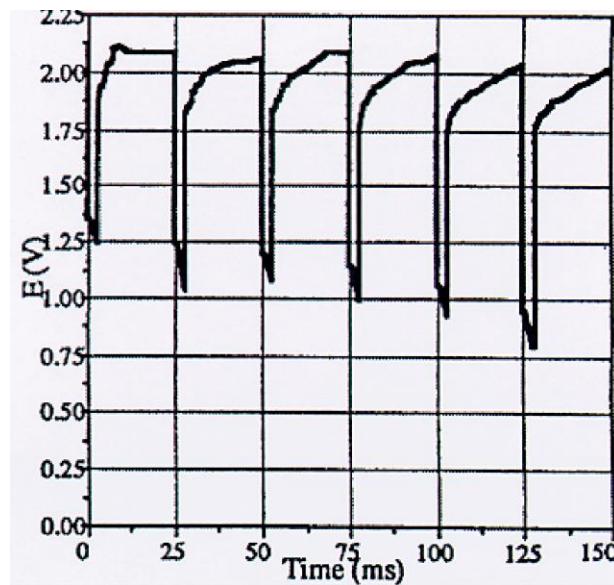
# Battery aware design: Example 1

- Performance of a bipolar lead-acid battery subjected to six current impulses. Pulse length=3 ms, rest period=22 ms.

**Current**

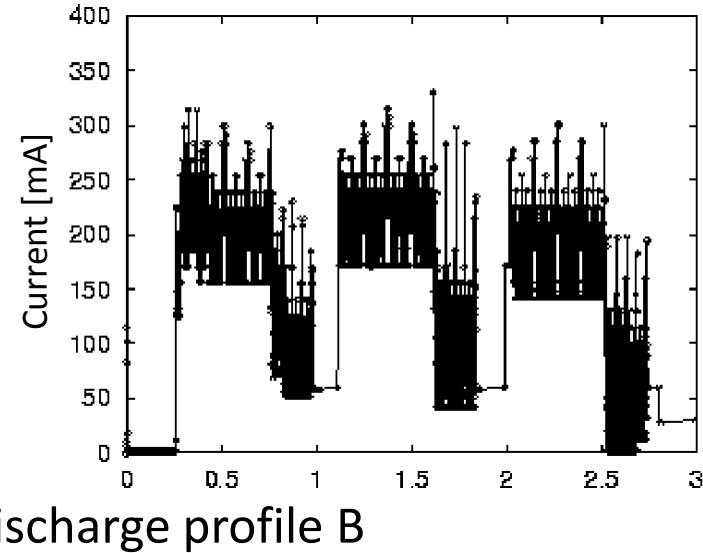
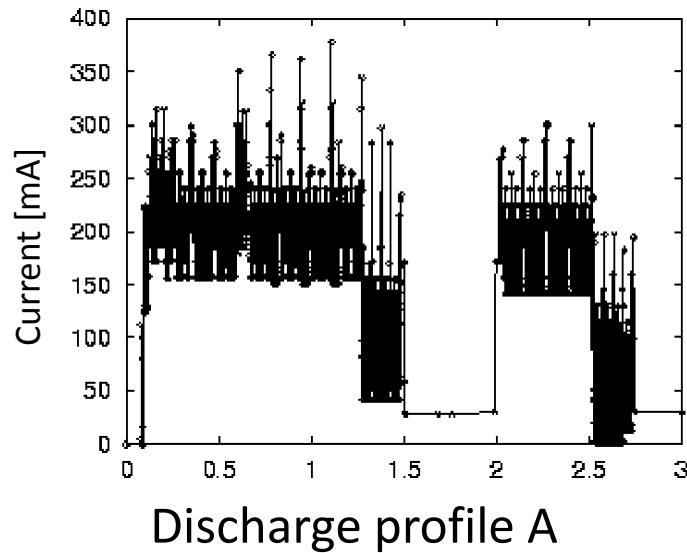


**Battery Voltage**



Source: LaFollette, "Design and performance of high specific power, pulsed discharge, bipolar lead acid batteries", *10th Annual Battery Conference on Applications and Advances*, Long Beach, pp. 43–47, January 1995.

# Battery aware design: Example 2



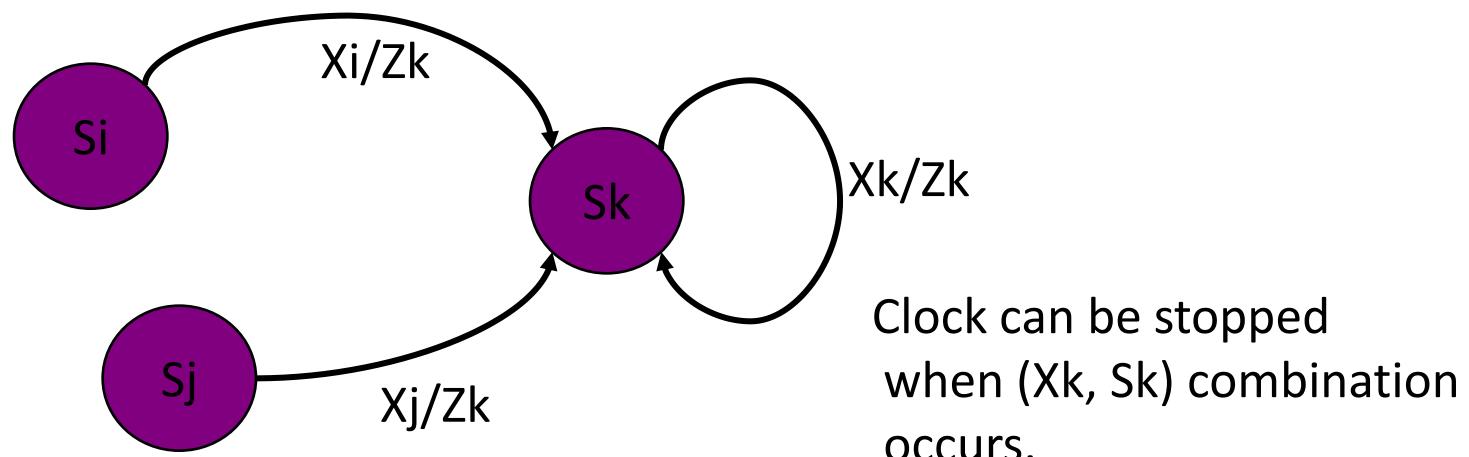
Profile	Aver. Current [mA]	Battery lifetime [ms]	Specif. energy [Wh/Kg]
A	<b>123.8</b>	357053	15.12
B	<b>124.2</b>	536484	18.58

→ Minimum average current ≠ Maximum battery life time

# Backup

# FSM: Clock-Gating

- Moore machine: Outputs depend only on the state variables.
  - If a state has a self-loop in the state transition graph (STG), then clock can be stopped whenever a self-loop is to be executed.



# Trend: Interconnects

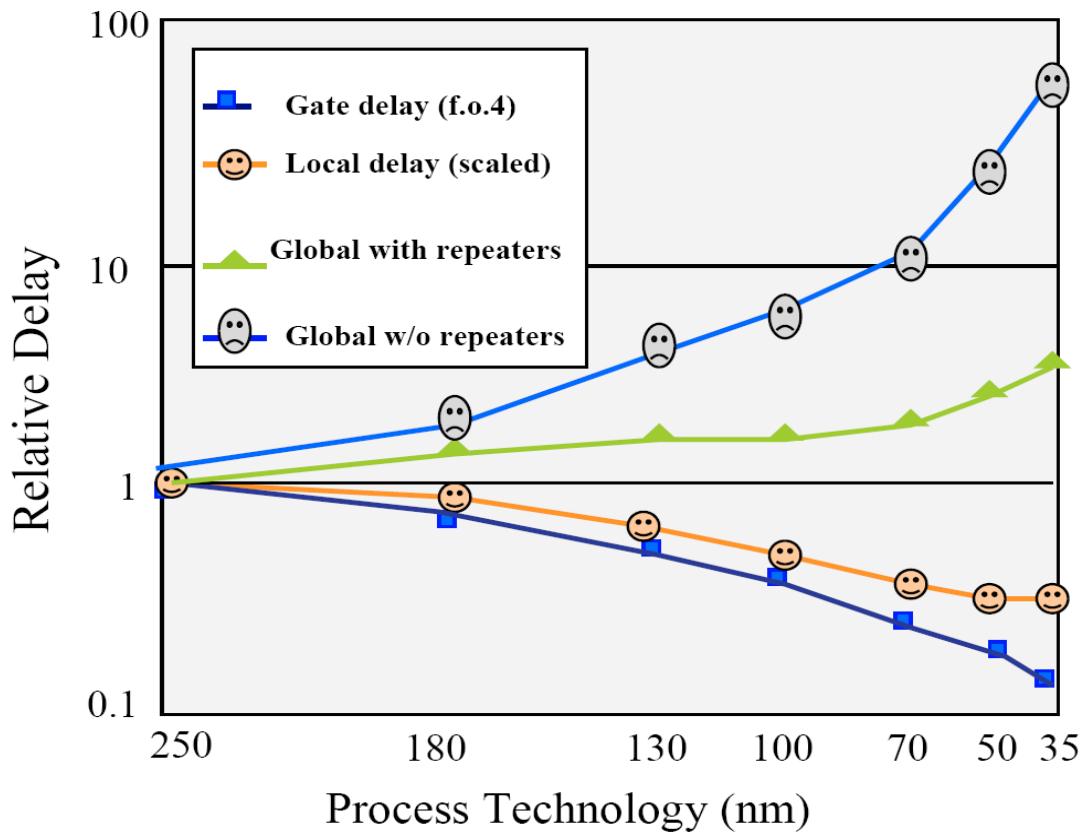
## Interconnects

Propagation delays of global wires will be a multiple of the clock cycle.

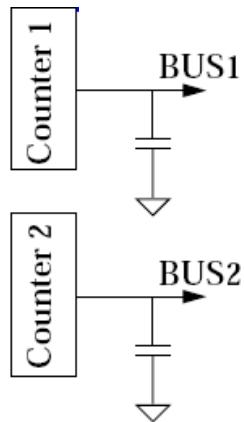
Example (very optimistic):

**6–10 clock cycles in 50nm technology**

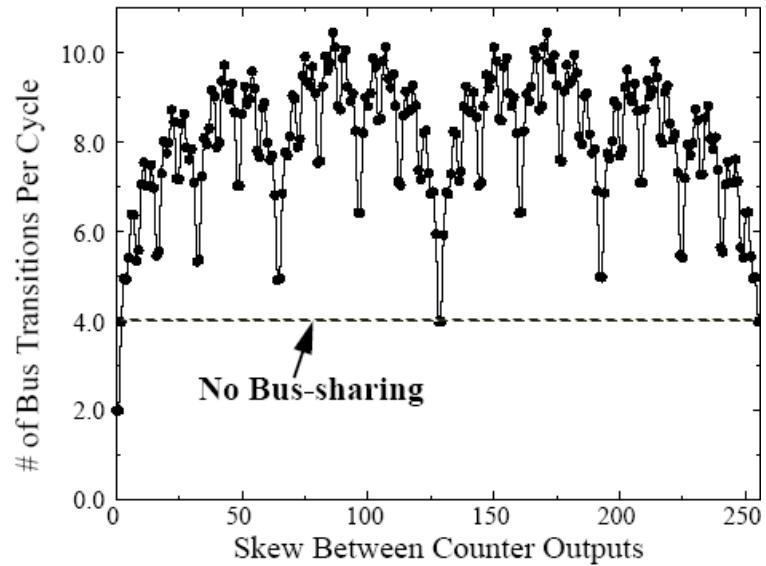
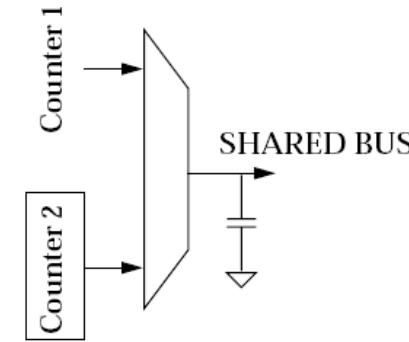
[Benini, 2002]



# Bus Multiplexing

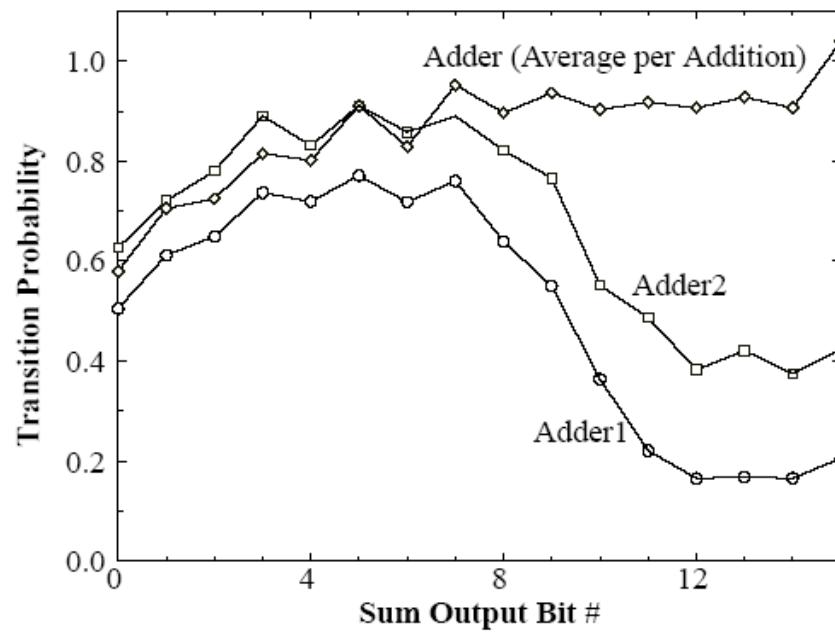
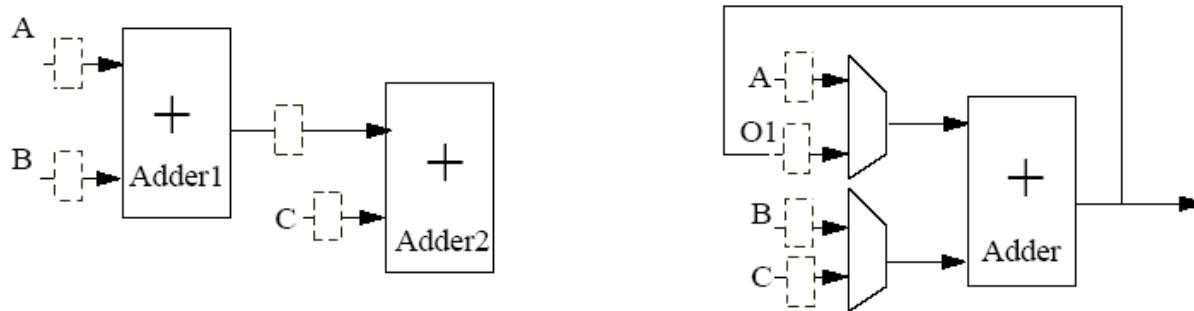


or



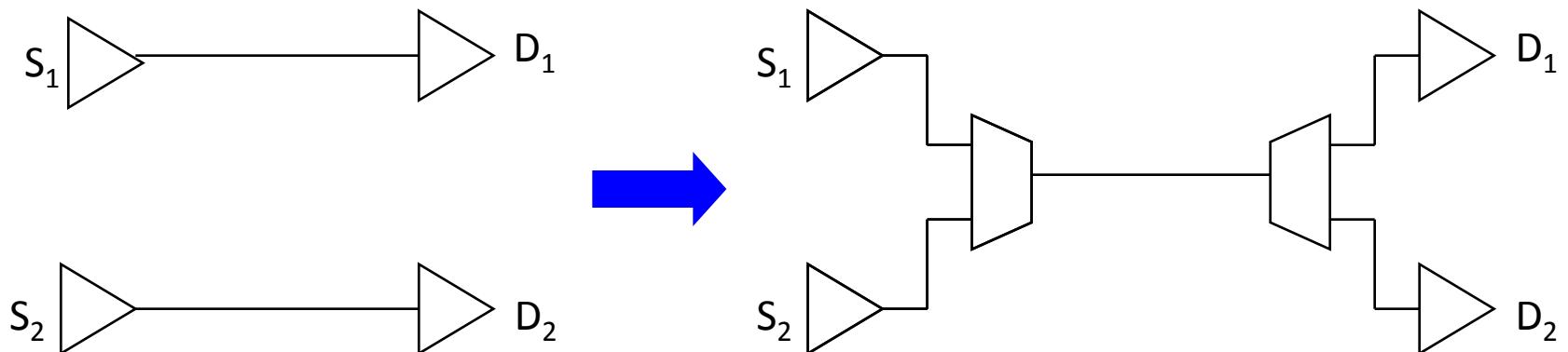
- Number of bus transitions per cycle  
 $= 2 (1 + 1/2 + 1/4 + \dots) = 4$

# Resource Sharing and Activity II

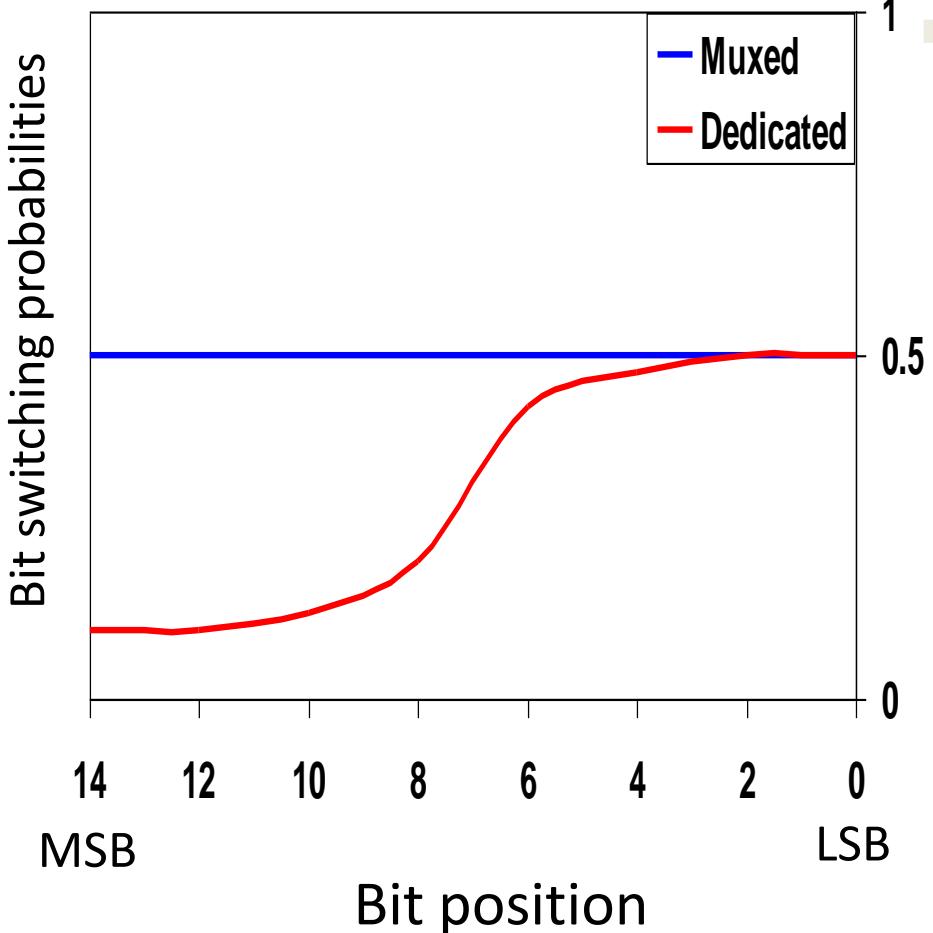


# Bus Multiplexing

- Sharing of long data buses with time multiplexing
- Example:
  - $S_1$  uses even cycles
  - $S_2$  odd



# Correlated Data Streams



- For a shared (multiplexed) bus advantages of data correlation are lost (bus carries samples from two uncorrelated data streams)
  - Bus sharing should not be used for **positively** correlated data streams
  - Bus sharing may prove advantageous in a **negatively** correlated data stream (where successive samples switch sign bits) - more random switching

# Disadvantages of Bus Multiplexing

- If data bus is shared, advantages of data correlation are lost (bus carries samples from two uncorrelated data streams)
- Bus sharing should not be used for positively correlated data streams
- Bus sharing may prove advantageous in a negatively correlated data stream (where successive samples switch sign bits) - more random switching

# Adaptive DVS cont'd

- Implementation

