Design of Scaled CMOS Circuits in the Nano-meter Regime: Dynamic Energy Dissipation

Kaushik Roy

Professor of Electrical & Computer Engineering

Purdue University

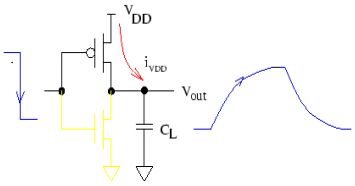
Switching/Dynamic Power

Switching Power

- Signal properties
  - Signal probability, P_i, - probability of a signal being logic ONE
  - Signal activity, a_i, - probability of signal switching(0->1, or 1->0)
- Energy dissipated per transition

$$E_{VDD} = \int_0^\infty i_{VDD}(t) V_{DD} dt = V_{DD} \int_0^\infty C_L \frac{dv_{out}}{dt} dt$$

$$= C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2$$



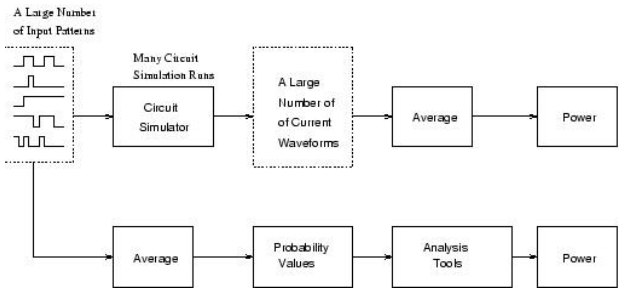
$$E_C = \int_0^\infty i_{VDD}(t) v_{out} dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{V_{DD}} v_{out} dv_{out} = C_L V_{DD}^2 / 2$$

Energy dissipated for 1->0 or 0->1 transition: $C_L V_{DD}^2 / 2$

$P_{dynamic} = C_L . V_{DD}^2 . f$

- Example
  - 1.2μ CMOS chip
  - 100 MHz clock rate
  - Average load capacitance of 30 fF/gate
  - 5V power supply
- Power consumption/gate = 75 μW
- Design with 200,000 gates: 15W !
- Pessimistic evaluation: not all gates switch at the full rate
- Have to consider the activity factor α: Effective switching capacitance = αC_L
- Reducing V_{DD} has a quadratic effect on $P_{dynamic}$

Average Number of Transitions



Switching at internal nodes depends on input signals.

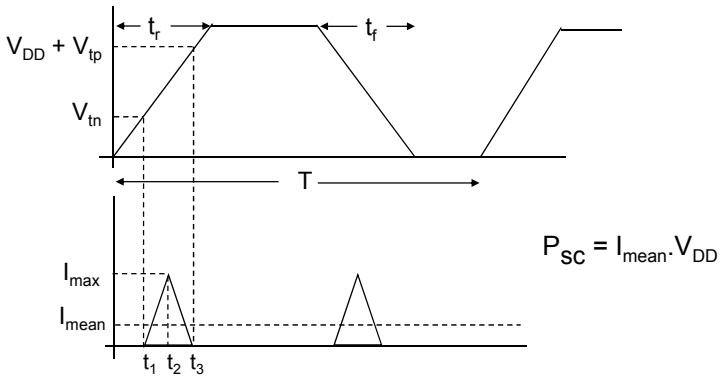
Model input signals as stochastic process. Each signal having some properties:

- Signal probability

- Signal activity

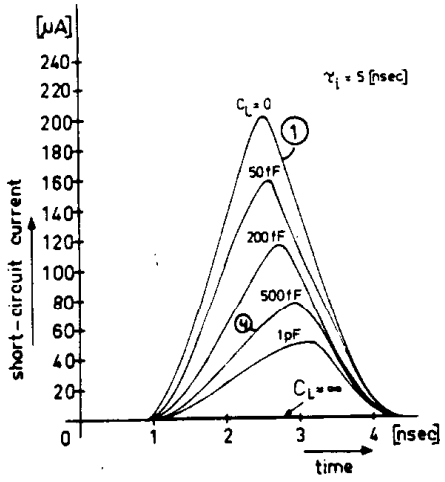
Direct Path Current

- inputs have finite rise and fall times
- Direct current path from V_{DD} to GND while PMOS and NMOS are ON simultaneously for a short period

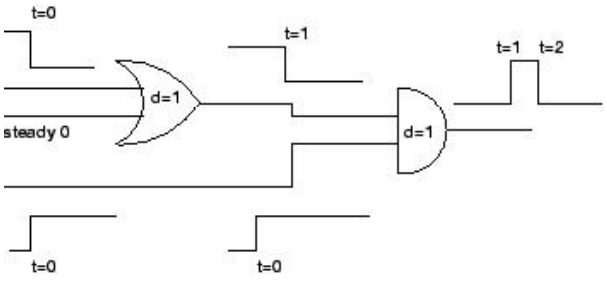


$P_{sc} = I_{mean}.V_{DD}$

Short Circuit Current with Loads



Spurious Transition at a Node



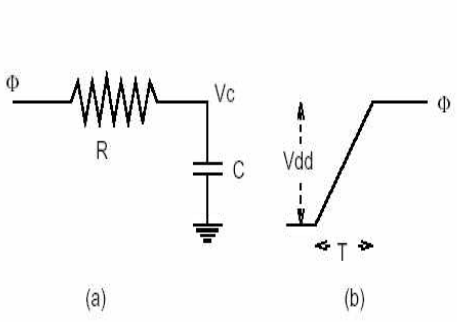
Hazardous transition occurs at the output of AND gate due to different delays through two different paths converging at the inputs to the AND gate.

- Assume each gate has unit delay
- Width of the glitch depends on the delays through the logic gates and interconnects.

Energy Dissipation in RC Circuits & Brief Intro to Energy Recovery

Revisit Dynamic Energy Comsumption: Simple R-C model of Pass pMOS transistor

Consider a pMOS pass transistor with a capacitive load C at the out put. The voltage at the power terminal swings from 0 to V_{dd} to charge the node capacitance through the transistor channel. The channel is modeled by a normal resistance R in fig (a).



Let us compute energy dissipated while charging capacitance C from 0 to V_{dd} in time T with a linear supply voltage as shown in fig (b).

RC Circuit: Energy Dissipation

The voltage relations are shown here.

$$RC(\frac{dV_C}{dt}) + V_C = \Phi$$

The supply voltage can be expressed as

$$V_C = \begin{cases} 0, & t < 0 \\ (\frac{V_{dd}}{T}), & 0 \le t < T \\ V_{dd}, & t > T \end{cases}$$

RC Circuits (contd.)

Solving the voltage equation, we've

$$V_C = \begin{cases} 0, & t < 0 \\ \Phi - (\frac{RC}{T})V_{dd}(1 - e^{-\frac{t}{RC}}), & 0 \le t < T \\ \Phi - (\frac{RC}{T})V_{dd}(1 - e^{-\frac{t}{RC}})e^{-\frac{(t-T)}{RC}}, & t > T \end{cases}$$

The energy dissipation in the charging process can be calculated as

$$E_{linear} = \int_0^T iV_R dt + \int_T^\infty iV_R dt$$

RC Circuits (contd.)

Term wise expanding, the first term can be expressed as

$$\int_0^T iV_R dt = \int_0^T \frac{(\Phi - V_C)^2}{R} dt$$

$$= \int_0^T [\frac{V_{dd}}{T} RC(1 - e^{(-\frac{t}{RC})}] / R dt$$

$$= (\frac{RC}{T})^2 CV_{dd}^2 \int_0^{\frac{T}{RC}} (1 - e^{-\frac{t}{RC}})^2 d(\frac{t}{RC})$$

$$= (\frac{RC}{T}) CV_{dd}^2 \left[1 - \frac{3}{2}(\frac{RC}{T}) + 2(\frac{RC}{T}) e^{(-\frac{T}{RC})} - \frac{1}{2}(\frac{RC}{T}) e^{(-\frac{2T}{RC})} \right]$$

The second term of the energy dissipation

$$\int_T^\infty iV_R dt = \int_T^\infty \frac{(\Phi - V_C)^2}{R} dt$$

$$= \frac{RC}{T} CV_{dd}^2 (1 - e^{-\frac{T}{RC}})^2 \int_T^\infty e^{-2\frac{(t-T)}{RC}} dt$$

$$= (\frac{RC}{T})^2 CV_{dd}^2 [\frac{1}{2}(1 - e^{-\frac{T}{RC}})^2]$$

Final energy expression

$$E_{linear} = (\frac{RC}{T}) CV_{dd}^2 [1 - \frac{RC}{T} + \frac{RC}{T} e^{-\frac{T}{RC}}]$$

Energy Dissipation: 2 Cases

Let us consider the two extreme cases, when **T >> RC**

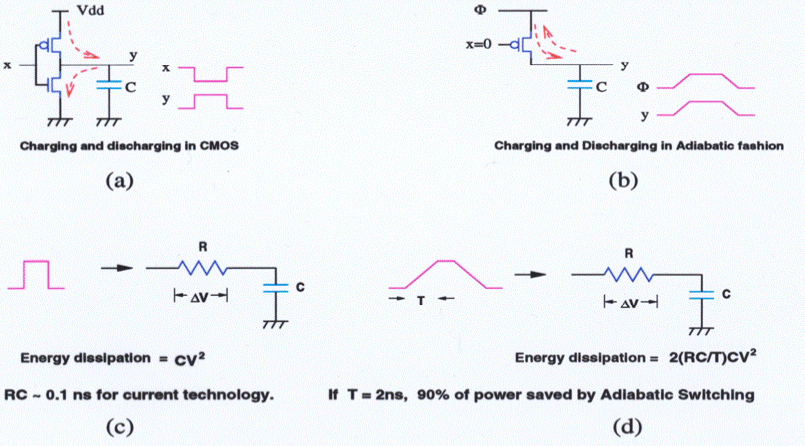
$$E_{linear} = (\frac{RC}{T})CV_{dd}^2$$

And when **T << RC**, as in normal CMOS

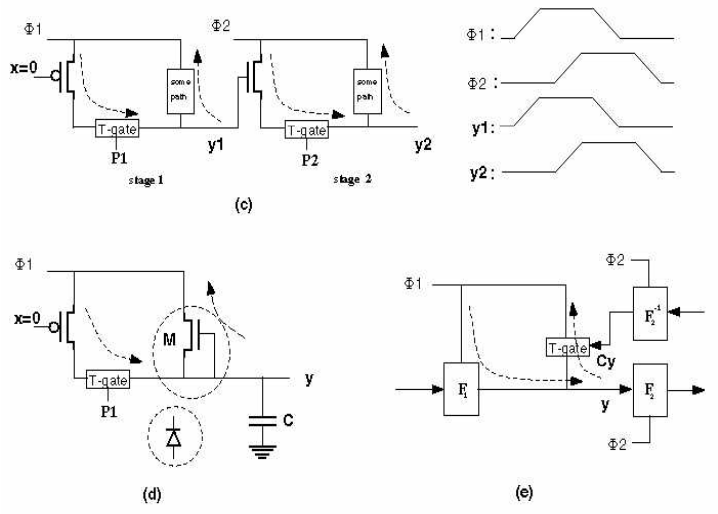
$$E_{linear} = (\frac{RC}{T})CV_{dd}^2[1-\frac{RC}{T}+\frac{RC}{T}(1-\frac{T}{RC}+\frac{1}{2}(\frac{T}{RC})^2)]$$

$$= \frac{1}{2}CV_{dd}^2$$

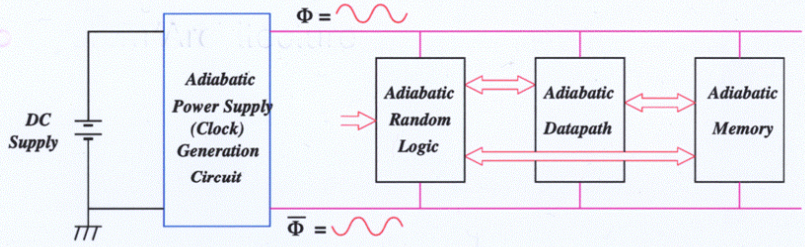
Adiabatic Switching Concept (continued)



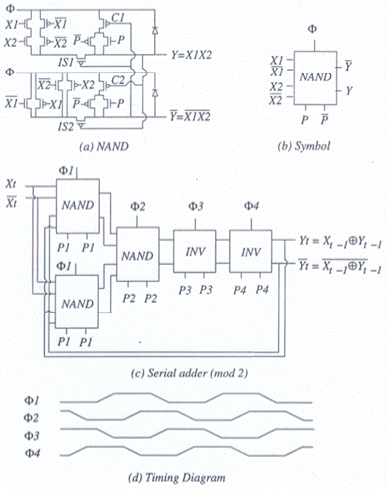
Basic recovery process



Adiabatic Digital System



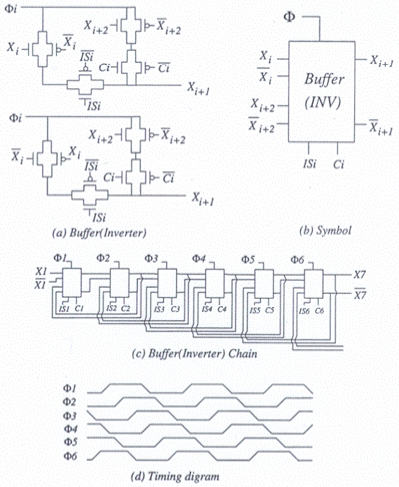
A Bit-Serial Adder using Partially Reversible Logic



- 4 phases of clock required
- Using local signal to control the recovery path. Hence, inverse function is not required
- Differential signaling is used

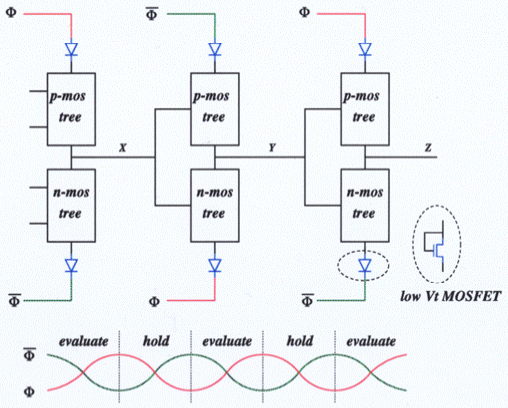
A Buffer (Inverter) Chain using Reversible Logic

- 6 phases of clock required
- Inverse logic naturally available
- Charge recovery path can be controlled by inverse function (next stage gate in buffer chain)
- In general, reversibility is not available



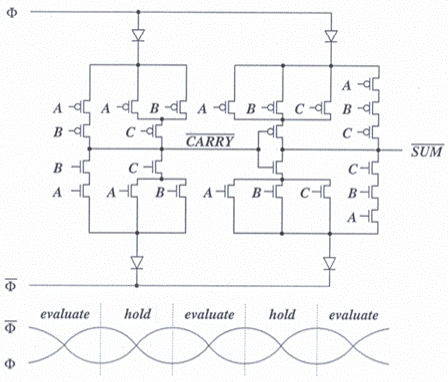
Quasi-Static Energy Recovery Logic (QSERL)

- Two phase clocks
- Comparable complexity with static CMOS
- Low threshold voltage MOSFET as the diode
- Lower switching activity than dynamic adiabatic logic



A Full Adder Using QSERL

- A quasi-CMOS adiabatic adder
- Works in both static CMOS and adiabatic mode
- A 2x2 adiabatic multiplier using this adder implemented

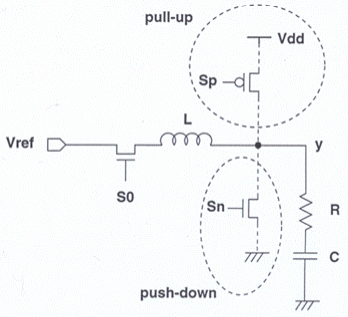


Summary of Simulation Results of Adiabatic Logic Blocks

- A buffer chain using reversible logic
  - At 1 MHz, 94% of energy recovered
  - At 111 MHz, 68% of energy recovered
- A bit-serial adder using partially reversible logic
  - At 1 MHz, 90% of energy recovered
  - At 111 MHz, 61% of energy recovered
- A 2x2 multiplier using QSERL logic
  - At 20 MHz, 60% of energy saved
  - At 100 MHz, 35% of energy saved

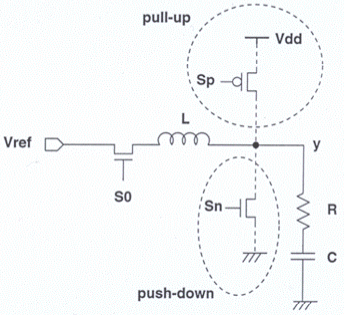
A Generic Resonant Scheme for Energy Recovery

- Ideally, the circuit oscillates between ***0*** and $2V_{ref}$
- Pull-up and pull-down paths to replenish the energy to keep oscillation going
- Extra circuits to generate control signal ***Sp*** and ***Sn***
- External control signals ***Sp*** and ***Sn*** are ***180*** degree out of phase



A Generic Resonant Scheme (continued)

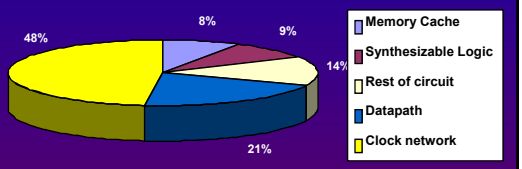
- Serial connected control transistor ***S0*** limits the energy recovery efficiency
- Extra circuitry to generate ***Sp*** and ***Sn***
- Energy in charging the gate capacitances of ***Sp*** and ***Sn*** are dissipated
- It requires an additional reference voltage source
- Single phase clock is generated. More than one resonant circuit is required



Motivation For Reducing Clock Power

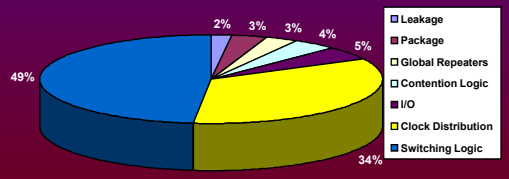
- **Power Consequences:**
  - **Increased cooling cost**
  - **Shortens battery lifetime in portable applications**
- **Clock power is significant**
  - **For microprocessors clock distribution power can range from 30% to 50%**
- **Clock power reduction is a promising approach to low power**
  - **Energy Recovery from the clock network**

Power Distribution of Pentium II



Ref: ISCAS01, Q.K. Zhu

Power Distribution of McKinley (Itanium) Processor

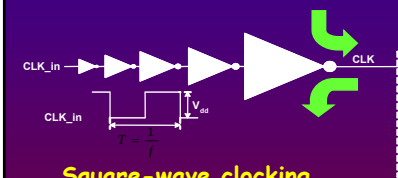


Ref: ISSCC01, McKinley

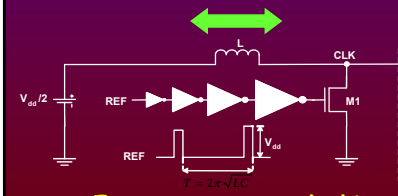
Energy-Recovery Clock Generation

- Resonant clock generator
- Sinusoidal clock
- Clock network: distributed RC load
- To adjust frequency, L is changed according to:

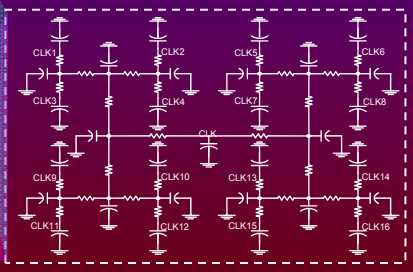
$$f = \frac{1}{2\pi\sqrt{LC}}$$



Square-wave clocking



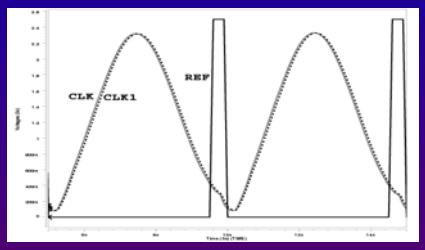
Energy-recovery clocking



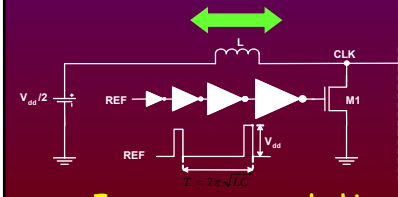
Clock distribution network

Clock Tree Simulation

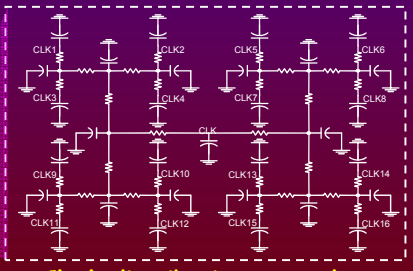
- Integrated 1024 flip-flops across an area of 4mmX4mm
- Compared proposed flip-flops to 3 square wave flip-flops
  - Hybrid-latch Flip-Flop
  - Conditional Capture Flip-Flop
  - Transmission Gate Flip-Flop
- Distributed RC model was extracted from layout
- In square wave case, clock tree was driven by a single buffer



Approximate Sinusoidal Generated



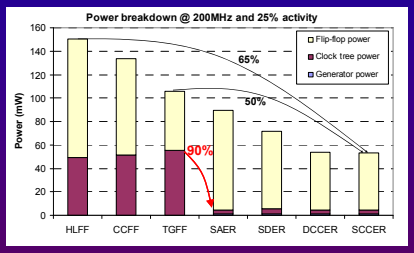
Energy-recovery clocking

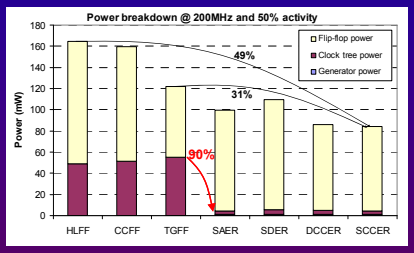


Clock distribution network

Energy-Recovery Results

Results include clock network and flip-flops





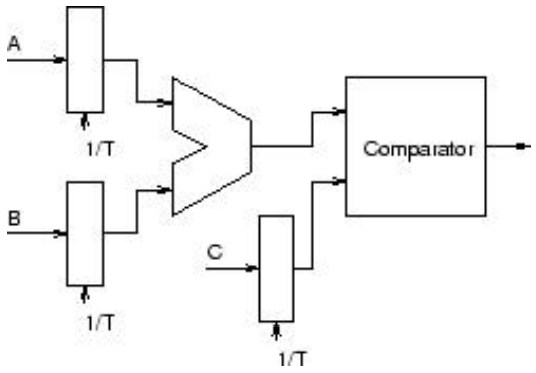
HLFF: Hybrid Latch Flip-Flop CCFF: Conditional Capture Flip-Flop TGFF: Transmission Gate Flip-Flop

- Negligible power overhead for clock generation
- Over 90% power savings over the clock tree!
- Total power savings including flip-flops over square-wave clocking:
  - Up to 83% for 0% data switching activity
  - Up to 65% for 25% data switching activity
  - Up to 49% for 50% data switching activity

65% power reduction at typical data switching rate

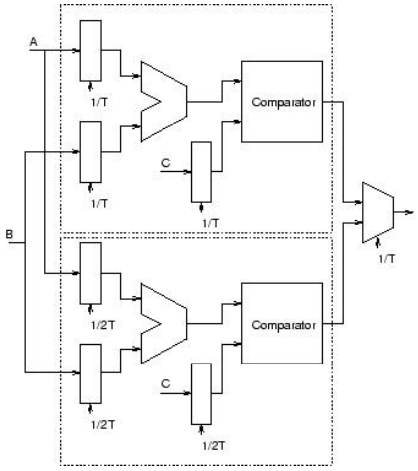
Dynamic Energy Minimization

Architecture-Driven Voltage Scaling



Data Path Operator

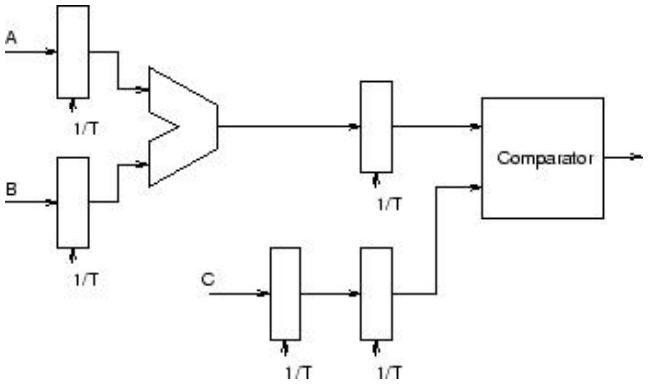
Architecture-Driven Voltage Scaling



$$P_{par} = (2.15C)(0.58V)^2(0.5f) \approx 0.36P$$

Parallel implementation

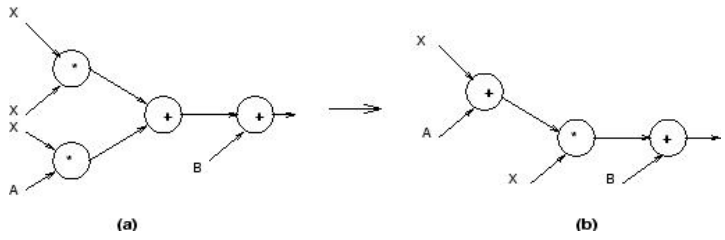
Architecture-Driven Voltage Scaling



Pipelined implementation

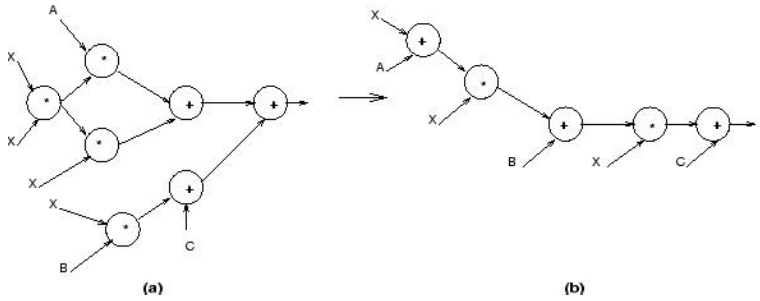
$$P_{pipe} = (1.15C)(0.58V)^2(f) \approx 0.39P$$

Power Optimization Using Operation Reduction



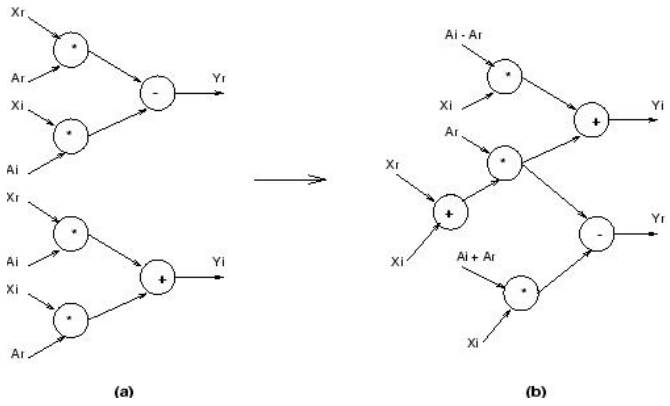
Reducing operations maintaining throughput

Power Optimization Using Operation Reduction



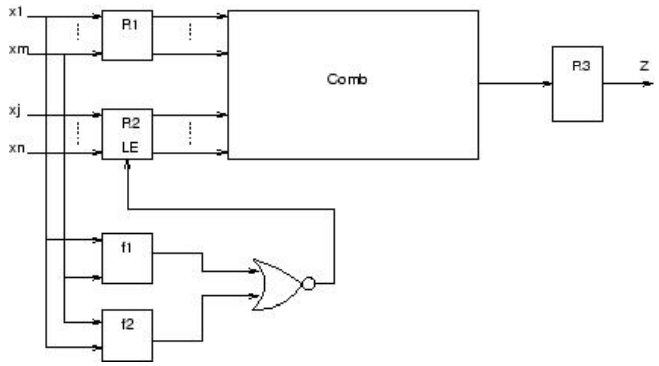
Reducing operations with less throughput

Power Optimization Using Operation Substitution



Substituting addition for multiplication

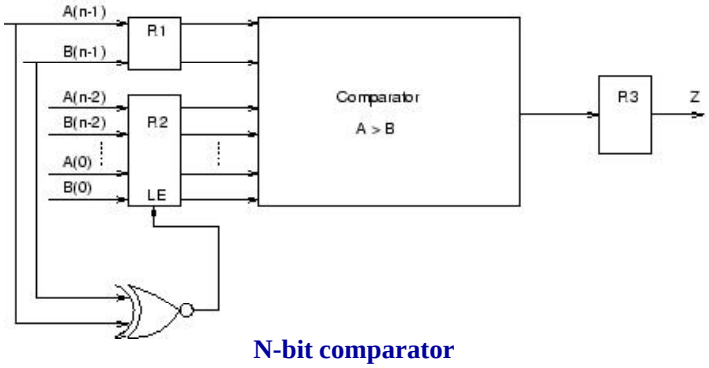
Precomputation-Based Optimization for Low Power



Precomputation architecture

$f_1 = 1 \Rightarrow Z = 1 \qquad f_2 = 1 \Rightarrow Z = 0$

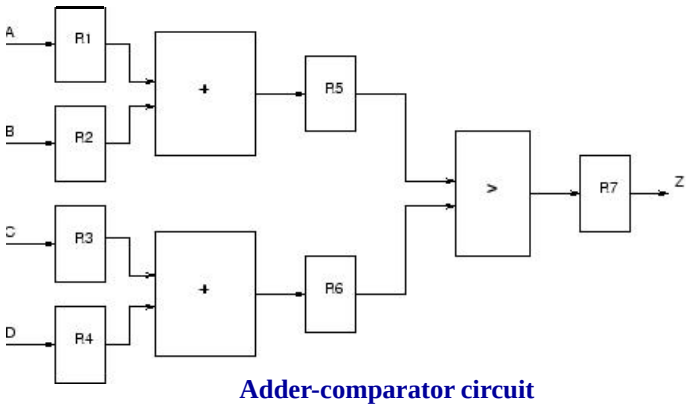
Precomputation-Based Optimization for Low Power



N-bit comparator

$f_1 = A(n-1) \cdot \overline{B(n-1)} \qquad f_2 = \overline{A(n-1)} \cdot B(n-1)$

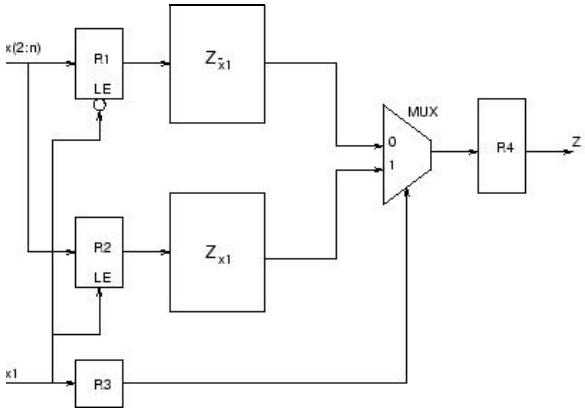
Precomputation-Based Optimization for Low Power



Adder-comparator circuit

$$f_1 = A(n-1)\cdot B(n-1)\cdot \overline{C(n-1)}\cdot \overline{D(n-1)}$$
$$f_2 = \overline{A(n-1)}\cdot \overline{B(n-1)}\cdot C(n-1)\cdot D(n-1)$$

Precomputation-Based Optimization for Low Power

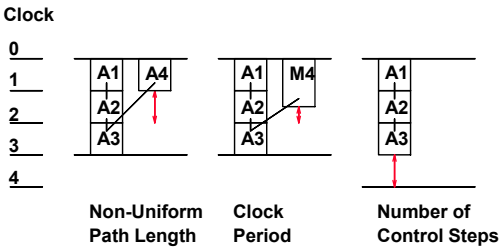


Precomputation using Shannon's expansion

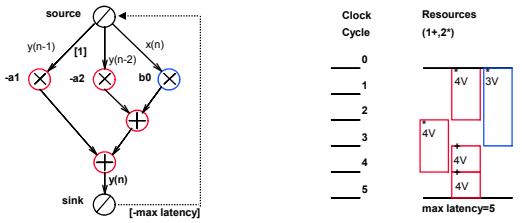
$$Z = x_j Z_{x_j} + \overline{x}_j Z_{\overline{x_j}}$$

Multi-Voltage Scheduling

Bottlenecks for single V_{dd}, V_T, Clock

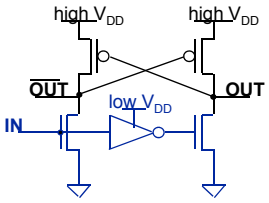


Schedule to exploit slack



Multi-Voltage IC Design Issues

Level Conversions



DC-DC Efficiency

- need efficiency of at least $\frac{V_{HI}^2}{V_{LO}^2}$ to break even

Layout:

- separate power and ground routing
- substrate contacts between voltage regions

Multi-Voltage Results

- Summary of results:
  - up to 50% energy savings 1 vs. 2 voltages
  - less than 15% additional savings 2 vs. 3
  - area penalties vary from 0 up to 170%

Clock Gating

Why Clock Gating?

- Power breakdowns for processors

Pentium Pro

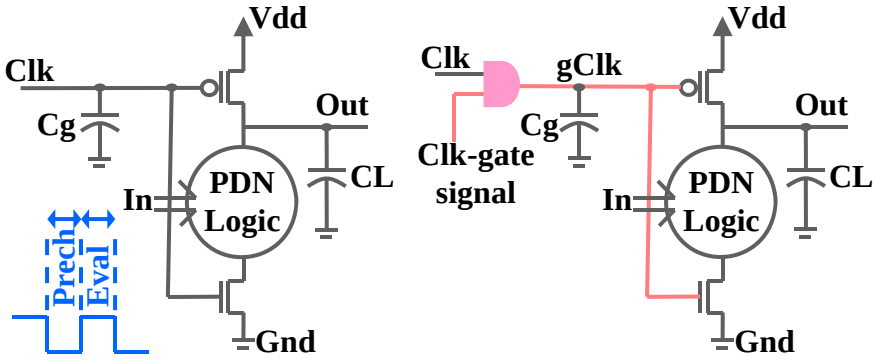
Instruction Fetch	22.2%
Register Alias Table	6.3%
Reservation Stations	7.9%
Reorder Buffer	11.1%
Integer Exec. Unit	14.3%
Data Cache Unit	11.1%
Memory Order Buffer	6.3%
FP Exec. Unit	7.9%
Global Clock	7.9%
Branch Target Buffer	4.7%

Alpha 21264

Caches	16.1%
Out-of-Order Issue Logic	19.3%
Memory Management Unit	8.6%
FP Exec. Unit	10.8%
Integer Exec. Unit	10.8%
Total Clock Power	34.4%

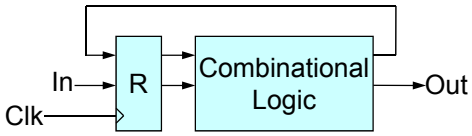
Principle of Clock Gating

- Clock gating a dynamic logic gate



Gated-Clock FSM

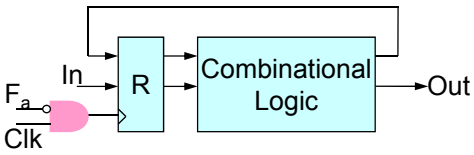
If the FSM enters a state with a self-loop, the signal F_a is asserted and the clock is turned off.



Single-clock, flip-flop based FSM

Limitation:

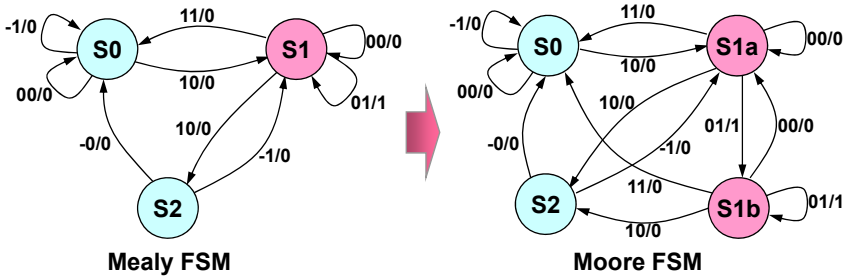
Only applicable to FSMs where the outputs do not depend directly on the primary inputs (*i.e.*,Moore FSMs).



Gated-clock version

FSM Transformation

- Locally transform a Mealy FSM into a Moore FSM



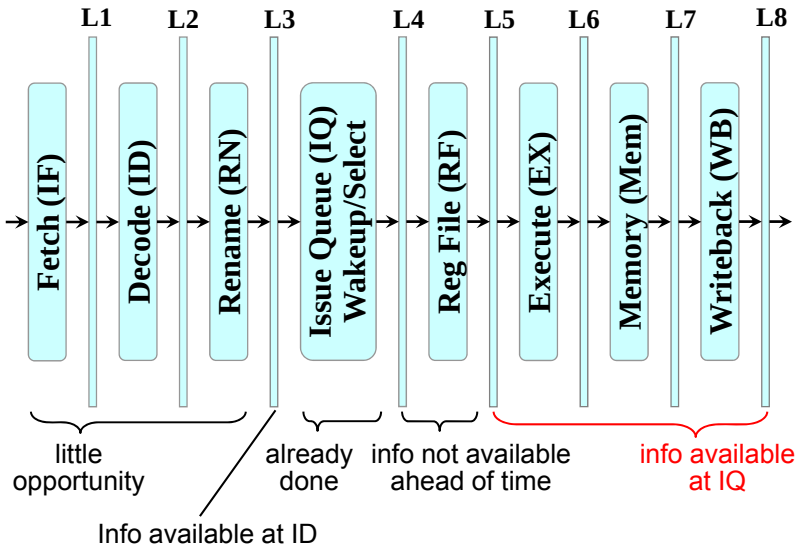
S0: Same output for all of self-loops.

S1: The output depends on the inputs for the diff. self-loops.

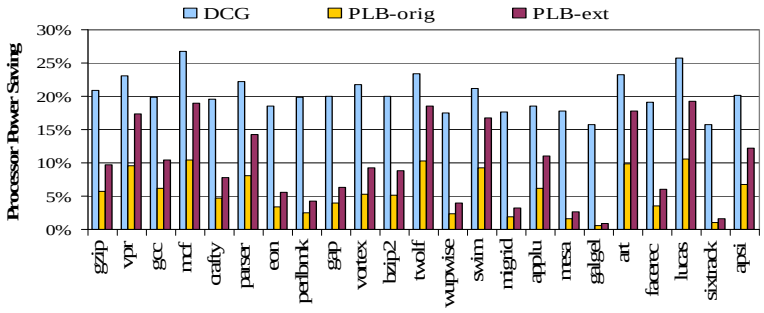
Deterministic Clock Gating (DCG) for High Performance Processors

- Target high-performance processors
- Resource use known in advance deterministically
- No prediction overhead
- More power savings
- Virtually no performance loss

DCG Applied to Back-end Stages, Latches



Effectiveness of DCG

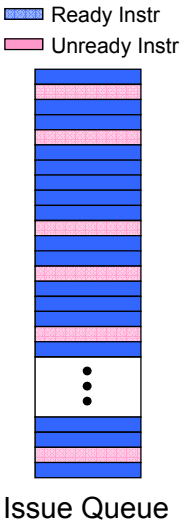


Average power savings: DCG 20%; PLB-orig 5.6%; PLB-ext 10%
Performance loss: DCG ~0%; PLB-orig & PLB-ext 2.8%

VSV:
Variable Supply-Voltage Scaling

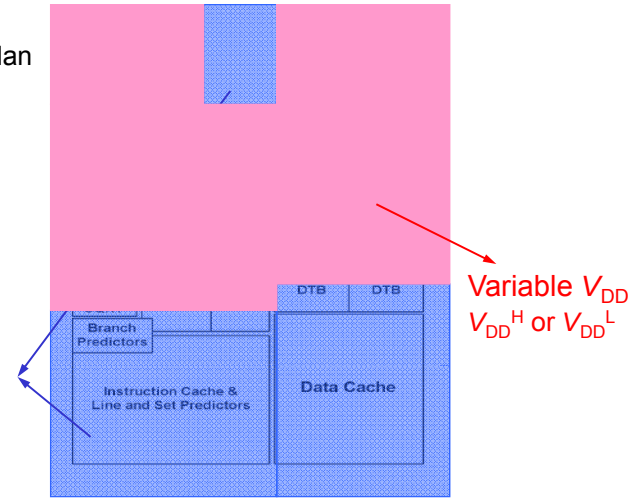
VSV: L2-Miss-Driven Variable Supply-Voltage

- CPU usually end up stalling on L2 misses
- L2 miss as trigger to transit from high to low V_{DD}



Implementation of VSV

Alpha 21264 floorplan

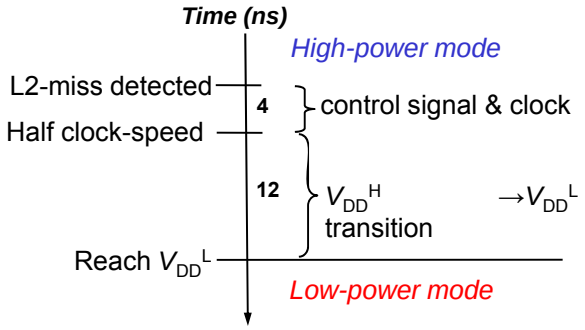


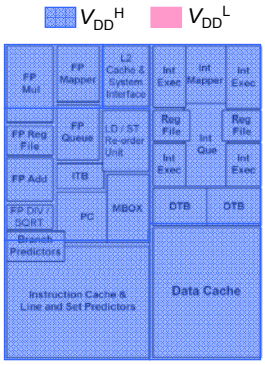
Variable V_{DD}
V_{DD}^{H} or V_{DD}^{L}

Fixed V_{DD}^{H} transition energy overhead

Two steady operation modes: *high-performance* & *low-power*

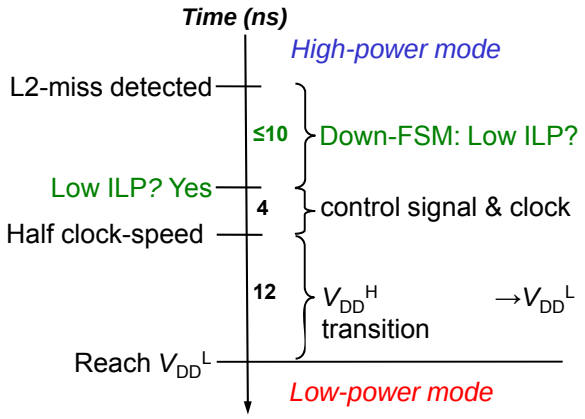
High- to Low-Power Mode Transition

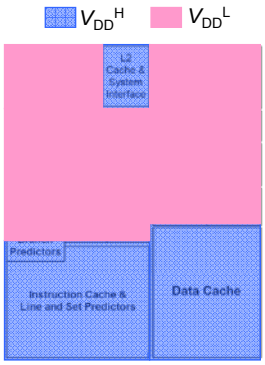




- Not so simple: What if high ILP overlaps misses?

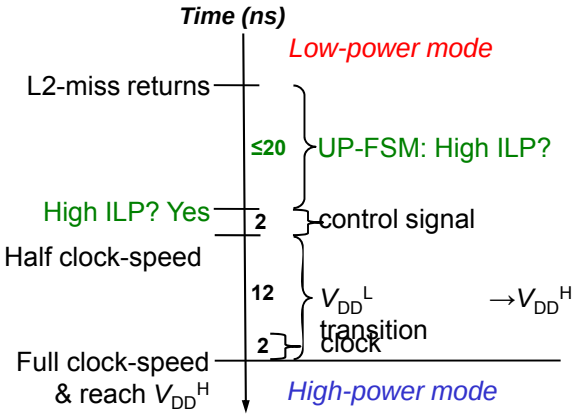
High- to Low-Power Mode Transition

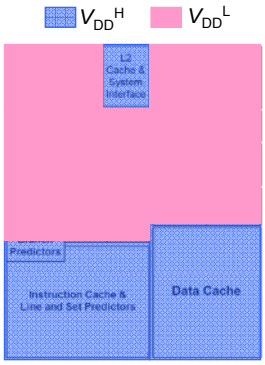




- Go to low-power mode only if low ILP
- Down-FSM avoids unnecessary performance loss

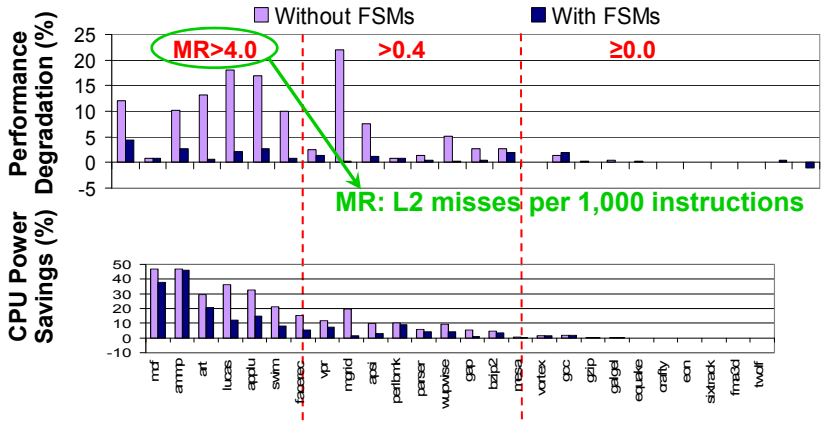
Low- to High-Power Mode Transition





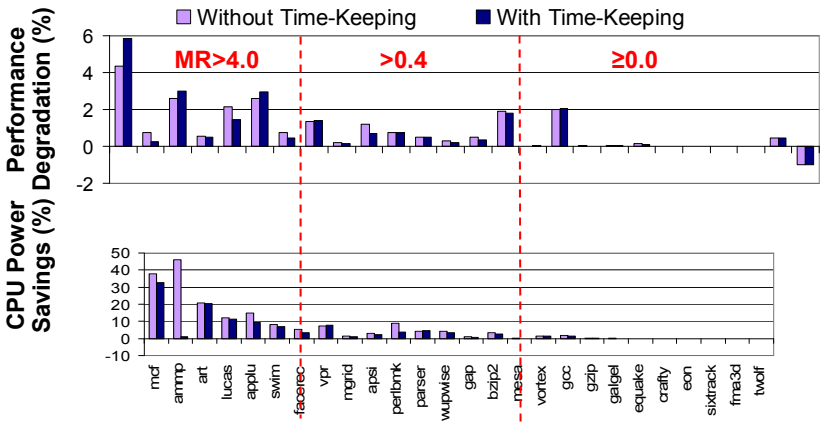
- Back to high-power mode when LAST L2 miss returns
- Up-FSM increase power savings

Effectiveness



- FSMs effectively avoid performance degradation
- Average CPU power savings : performance loss
  - 7% : 1% for all SPEC2K programs
  - 21% : 2% for programs with MR>4.0

Impact of Time-Keeping Prefetching



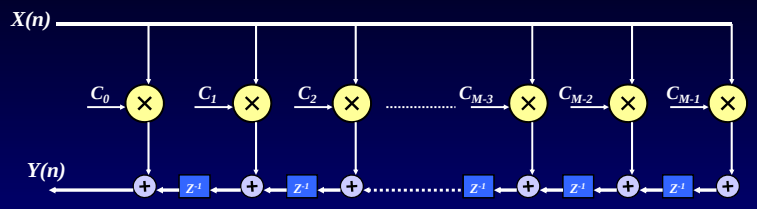
- Average CPU power savings : performance loss
  4% : 1% for all SPEC2K programs
  12% : 2% for programs with MR>4.0

Low-Power VLSI Signal Processing (Low-complexity DSP)

Shared Multiplier

- Reduction of redundant computation by increasing computation re-use
- Complexity reduction in FIR implementation
- High performance
- Low power
- Works efficiently if embedded in large DSP systems

Vector scaling operation



< Transposed direct form FIR filter >

$$[\, c_0, c_1, c_2 \ldots\ldots\ldots c_{M-2}, c_{M-1} \,] \quad \times \quad X(n)$$

- *FIR filtering operation can be expressed as a product of coefficient vector C and scalar X(n)*
  - Vector Scaling Operation , $Y = C \cdot x$

Shared Multiplier Algorithm

- Specifically targets the reduction of *redundant* computation in the vector scaling operation.

< Coefficient Decomposition >

$c = 111010001100$ $\quad c = 2^9(111) + 2^7(1) + 2^2(11)$

alphabet set = {1, 11, 111}

Alphabets - chosen basic bit sequences

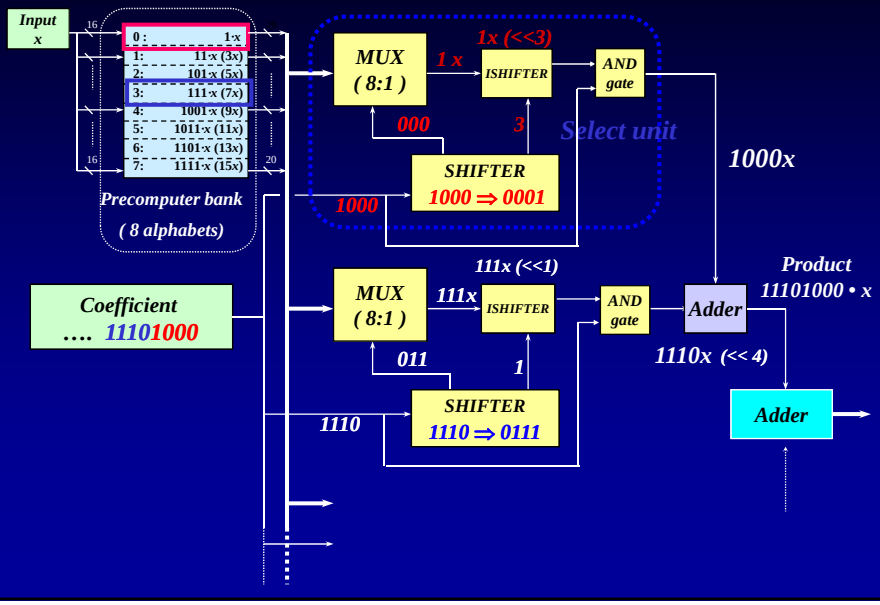
Alphabet set - a set of alphabets that covers all the coefficients in vector C

$c \bullet x = 111010001100 \bullet x$

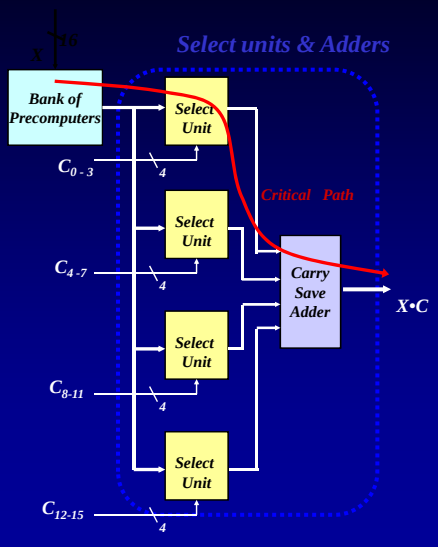
$c \bullet x = 2^9(0111 \bullet x) + 2^7(0001 \bullet x) + 2^2(0011 \bullet x)$

if $0111 \bullet x$, $0001 \bullet x$ and $0011 \bullet x$ are available, $c \bullet x$ can be significantly simplified as add and shift operation

Shared Multiplier Architecture



16×16 Shared Multiplier Implementation

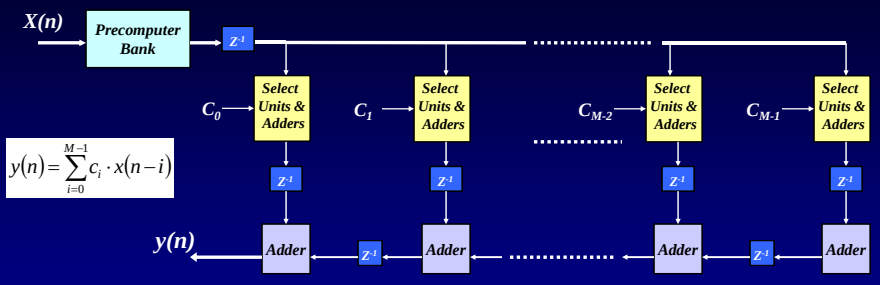


- 16 × 16 Wallace tree multiplier (WTM) and carry save array multiplier (CSAM) are also implemented for comparison.

	Precomputer	Select units & Adders	WTM	CSAM
Delay	6.923 ns	11.231 ns	16.638 ns	23.398 ns
Power	18.06 mW	18.91 mW	22.80 mW	21.78 mW
Area	162340 μm²	252120 μm²	241000 μm²	175640 μm²

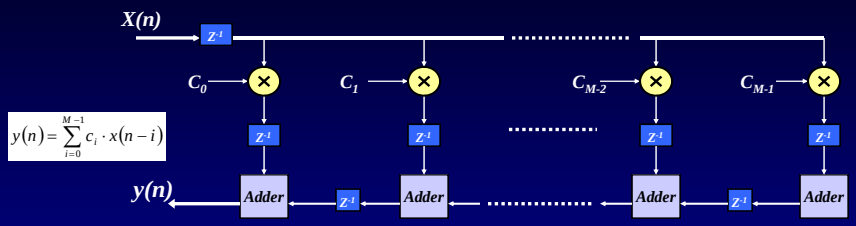
- CMU library (0.35 μm technology)

FIR filter using Shared Multiplier



- Computations $a_k \bullet x$ are performed just once for all alphabets and these values are shared by all the select units
- Only *select unit and adders* and lie on the critical path

FIR filter using WT & CSAM



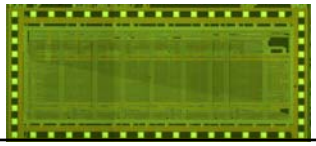
Filter	FIR filter using Shared Multiplier	FIR filter using Wallace Tree	FIR filter using Carry Save Array
Clock Cycle	13 ns	18 ns	25 ns
Power	398.4 mW	412.2 mW	401.1 mW
Area	4.41 × 10 ^{6} µm^2	3.87 × 10 ^{6} µm^2	3.15 × 10 ^{6} µm^2

• CMU library (0.35 µm technology) • Power measured with clock frequency : 25ns

Numerical Results

	Radix-16 CSHM (Synth.)	CSHM (Custom)	WTM (Synth.)	CSM (Synth.)
Power Consumption at 10 ns CLK (mW)	226.1	286.6	344.3	357.1
Min. CLK Cycle (ns)	5	7	8.5	10
Area (× mm^2)	4.1	5.0	4.4	4.1

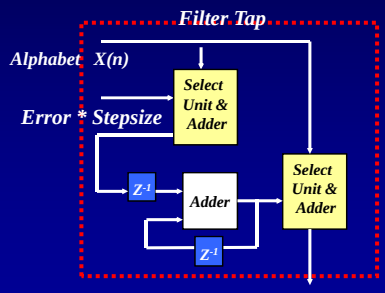
16 % 19 % 34 % 37 % 18 % 30 % 41 % 50 %



DFE using Shared Multiplier

- *Coefficient Update*

$$C_{k+1} = C_k + StepSize \times Error \times V_k$$



Filter	Clock Cycle	Area
DFE using CSHM	96.67 ns	$4.06*10^7$ µm²
DFE using WTM	112.73 ns	$2.51*10^7$ µm²
DFE using CSAM	117.23 ns	$2.51*10^7$ µm²

- *CMU library (0.35 µm technology)*

DCT: Shared Multiplier Application

DCT (Discrete Cosine Transform)

- The number of *alphabets* can be reduced by modifying the coefficients in DCT matrix
- Only *1x* & *3x* are required for the Precomputer bank
- Performance and Power improvement in *Precomputer bank* and *Select unit*.
- DCT with the modified coefficients generates acceptable quality of image

Shared Multiplier Application

DCT (Discrete Cosine Transform)

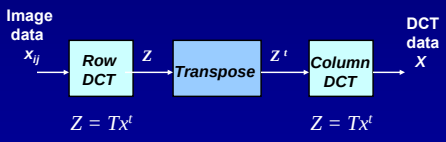
$$X_{kl} = \frac{c(k)c(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} x_{ij}\cos\left(\frac{(2i+1)k\pi}{16}\right)\cos\left(\frac{(2j+1)l\pi}{16}\right)$$

$$k,l = 0,1,\ldots,7 \text{ and } c(k) = \begin{cases}\frac{1}{\sqrt{2}}, k = 0\\ 1, otherwise\end{cases}$$

$$T = \begin{bmatrix} d & d & d & d & d & d & d & d \\ a & c & e & g & -g & -e & -c & -a \\ b & f & -f & -b & -b & -f & f & b \\ c & -g & -a & -e & e & a & g & -c \\ d & -d & -d & d & d & -d & -d & d \\ e & -a & g & c & -c & -g & a & -e \\ f & -b & b & -f & -f & b & -b & f \\ g & -e & c & -a & a & -c & e & -g \end{bmatrix}$$

Note the symmetry of the DCT coef. matrix

- $Z = Tx^t$, $Z = Tx^t$



DCT (Background)

- Using the Symmetry of the DCT coefficient matrix, the matrix multiplication is simplified.

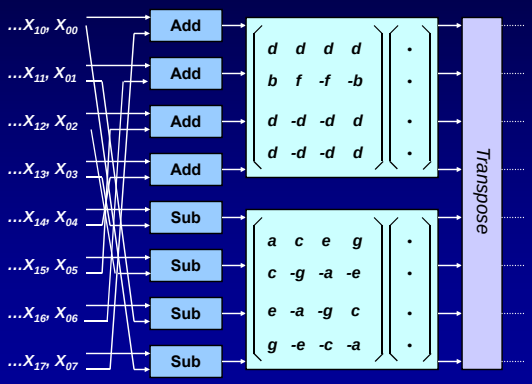
- $Z = Tx^t$, $X = TZ^t$

Even DCT

$$\begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \begin{bmatrix} d & d & d & d \\ b & f & -f & -b \\ d & -d & -d & d \\ f & -b & b & f \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

Odd DCT

$$\begin{bmatrix} z_1 \\ z_3 \\ z_3 \\ z_7 \end{bmatrix} = \begin{bmatrix} a & c & e & g \\ c & -g & -a & -e \\ e & -a & g & c \\ g & -e & c & -a \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}$$



DCT using Shared Multiplier

$$X_{kl} = \frac{c(k)c(l)}{4}\sum_{i=0}^{7}\sum_{j=0}^{7} x_{ij}\cos\left(\frac{(2i+1)k\pi}{16}\right)\cos\left(\frac{(2j+1)l\pi}{16}\right)$$

- $Z = Tx^t$, $X = TZ^t$

Even DCT

$$\begin{bmatrix} z_0 \\ z_2 \\ z_4 \\ z_6 \end{bmatrix} = \begin{bmatrix} d & d & d & d \\ b & f & -f & -b \\ d & -d & -d & d \\ f & -b & b & f \end{bmatrix} \begin{bmatrix} x_0 + x_7 \\ x_1 + x_6 \\ x_2 + x_5 \\ x_3 + x_4 \end{bmatrix}$$

Odd DCT

$$\begin{bmatrix} z_1 \\ z_3 \\ z_3 \\ z_7 \end{bmatrix} = \begin{bmatrix} a & c & e & g \\ c & -g & -a & -e \\ e & -a & g & c \\ g & -e & c & -a \end{bmatrix} \begin{bmatrix} x_0 - x_7 \\ x_1 - x_6 \\ x_2 - x_5 \\ x_3 - x_4 \end{bmatrix}$$

8bit DCT Coefficients

Original 8-bit DCT coefficient			
Coefficient	Value	Binary code	Pre-computer bank Needed
a	0.49	0011 1111	3x, 15x
b	0.46	0011 1011	3x, 11x
c	0.42	0011 0101	3x, 5x
d	0.35	0010 1101	1x, 13x
e	0.28	0010 0100	1x
f	0.19	0001 1000	1x
g	0.10	0000 1100	3x

Modified 8-bit DCT coefficient			
Coefficient	Value	Binary code	Pre-computer bank Needed
a'	0.50	0100 0000	1x
b'	0.47	0011 1100	3x
c'	0.41	0011 0100	1x, 3x
d'	0.34	0010 1100	1x, 3x
e'	0.28	0010 0100	1x
f'	0.19	0001 1000	1x
g'	0.12	0000 1100	3x

- **Only *1x* & *3x* are required in the Modified 8-bit DCT Coefficient**

DCT using Shared Multiplier

< DCT with original 8 bit coefficient >

< DCT with modified 8 bit coefficient >

- **DCT with the modified coefficients generates acceptable quality of image**

Shared Mutiplier: Summary

- Reduces computational complexity
- Possible to trade-off power/performance by judiciously selecting coefficients and alphabets

Differential Coefficients Method (DCM)

< FIR Filtering operation >

- An "***n***" tap FIR Filter performs the following computation :

$$Y_j = \sum_{k=0}^{n-1} C_k X_{j-k}$$

- *C's* are the filter coefficients
- *X* and *Y* are the input and output sequences.
- The filter output Y typically obtained by :
  - Computing each product term by multiplication
  - Summing up the product terms
- Called *Direct Form* (**DF**) computation of the FIR output

FIR Filters using Differential Coefficients

- Excepting the first coefficient the C's can be expressed as :

$$C_k = C_{k-1} + \delta^1_{k-1/k}$$

- The delta's are called the "**First Order Differences**"
- By expanding the expression for consecutive Y's and subtracting we get :

$$Y_{J+!} = Y_j + C_0 X_{j+1} + \sum_{k=1}^{N-1} \delta^1_{k-1/k} X_{j-k+1} - C_{N-1} X_{j-N+1}$$

- The "**First Order Partial Sum**" is defined as :

$$\sum_{k=1}^{N-1} \delta^1_{k-1/k} X_{j-k+1} = \{S^1_P\}_{t=j+1}$$

Advantages of the FD algorithm

- To obtain Y we now do the following :
  - Retrieve the previously computed Y
  - Compute the "partial sum" and the two other product terms by multiplication
  - Add these to obtain the current Y
  - Store the current Y

- Called the "*First Order Differences Algorithm*" (**FD**) for computing the FIR Filter output

- Computational savings of FD algorithm:
  - only if differences (delta's) are smaller than C's
  - product term computation simplified as the differences have reduced word-width

Algorithm using generalized differences

- Generalized *m*-th order differences defined as :

$$\delta^{m}_{k-m/k} = \delta^{m-1}_{k-m+1/k} - \delta^{m-1}_{k-m/k-1}$$

- We can thus generalize the recurrence for Y as :

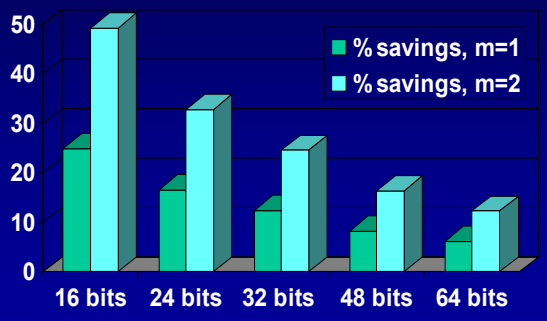
$$Y_{j+1} = Y_j + C_0 X_{j+1} + \sum_{k=1}^{m-1} \left[\left\{ S_P^k \right\}_{t=j} + \delta^{k}_{0/k} X_{j-k+1} \right.$$

$$\left. - \delta^{k}_{N-k-1/N-1} X_{j-N+1} \right] - C_{N-1} X_{j-N+1} + \sum_{k=m}^{N-1} \delta^{m}_{k-m/k} X_{j-k+1}$$

- Multiplications involve only m-th order differences
- Greater computational savings possible if the m-th order differences are even smaller than the coefficients

Example: Low Pass Filter

- Example : FIR Filter with 100 taps
- Shift and Add model for multiplication
- Energy dissipated data from a low-power library



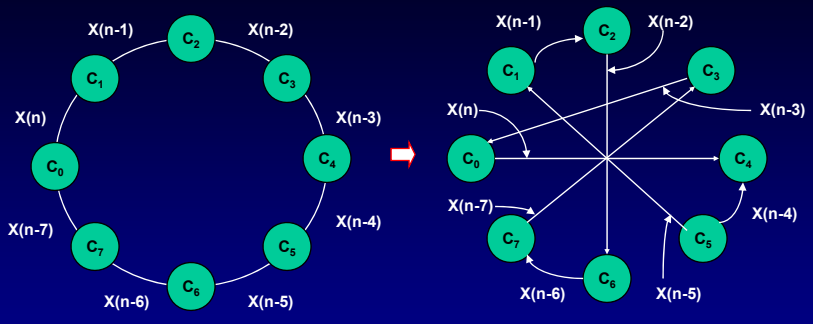
Canonical Signed Digits (CSD)

- Canonical Signed Digits **(CSD)** is another commonly used technique for simplifying multiplications in FIR filters
- Reduces switching-power by reducing number of ones in multiplier
- Typically a 33% reduction in the number of ones (and hence additions) is obtained using CSD

- **Compared to CSD, DCM has 40% better power dissipation**

- **Modifications to DCM is possible for near multiplier-less filters**

Coefficient Reordering



- **We represent this problem using a graph in which vertices represent the coefficients and edges represent the resources required when the differential coefficient corresponding to the edge is used in the computation**
- **The optimal solution is the well-known problem of finding the *Hamiltonian cycle* of smallest weight of this graph (*minimum spanning tree*), thus achieving even lower complexity**

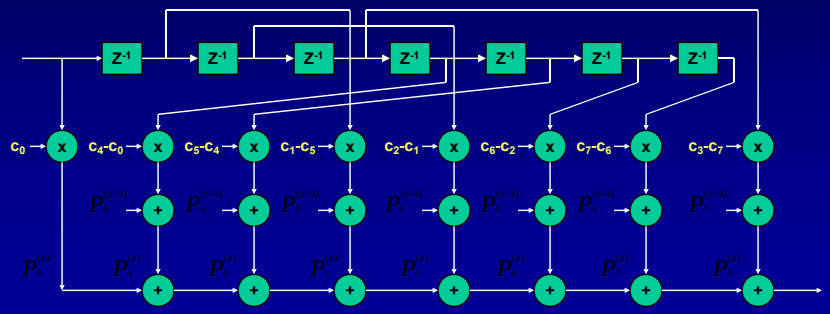
Implementation of DCMI

Compute the Hamiltonian cycle in G

$K = \{k_0, \ldots, k_{M-1}\}$=set of ordering of indices of the coefficients

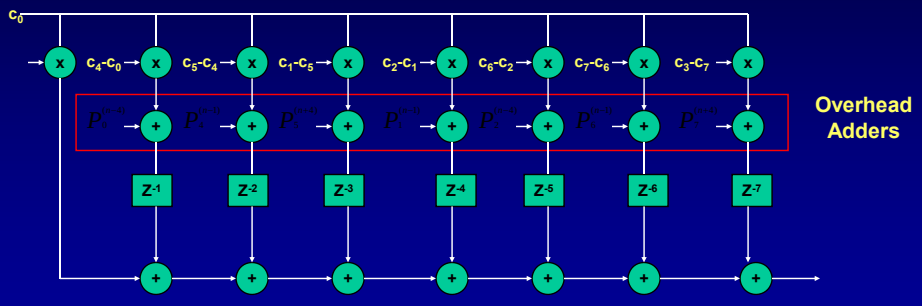
(e.g. In DCM $K=\{0,1,2,3,\ldots,M-1\}$.$\Delta c_j = c_{1+j} - c_j$)

Then $\Delta c_i = c_{k_{i+1}} - c_{k_i}$ and $P_{k_i}^{(n)} = (c_{k_i} - c_{k_{i-1}}) x(n-k) + P_{k_{i-1}}^{(n-k_i+k_{i-1})}$



Obtaining the Reduced Form

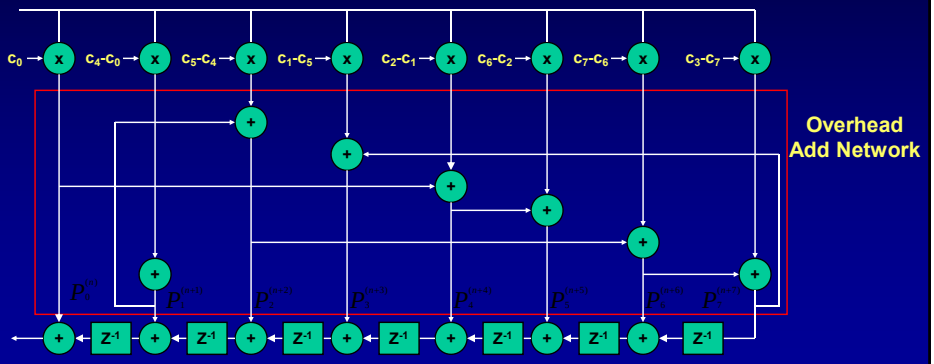
Step 1 : Move the delay elements into branches



Obtaining the Reduced Form

Step 2 : Move the delay elements out of the branches and connect the appropriate partial product.

- Overhead = M-1 Add operations.



Low Complexity FIR Filters with FPC

- Factorization of Perturbed Coefficients (FPC) is a method to design digital filters which require less computation.
- Factorization allows common factors between coefficients in the filter to be used to share computation.
- Perturbation maximizes the benefits of factorization by guaranteeing "good" common factors.
- FPC constrains the frequency response within acceptable limits.

Factorization

- The output of an FIR filter is:

$$y(n) = \sum_{i=0}^{M-1} C_i \, x(n-i)$$

- If two coefficients have a common factor there is a calculation that can be shared.

(i.e., If C1 = F1 * F2 & C2= F1 * F3, the value of F1*x(n) can be reused.)

- The problem is the lack of common factors across multiple coefficients.

Perturbation

- Every whole number is unique product of prime factors. Thus the coefficients can be expressed as:

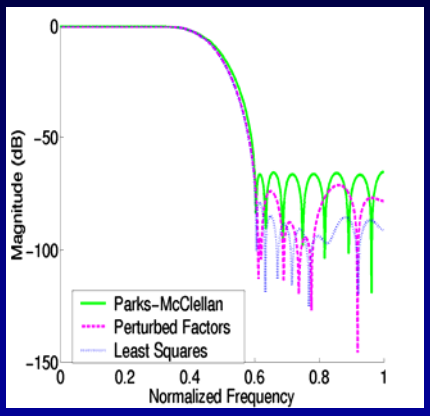
$$C_i = \prod_{k=1}^{Q(n)} \{ f_n(k) \}^{pn(k)}$$

- To maximize the number of common factors, generate the coefficients from a small set of prime factors.
- To minimize the impact on the filter output, these should be the first several prime factors: f = {2,3,5,7,11…}

FPC Algorithm

- Start with Parks-McClellan and Least Squares filters. These provide our bounds.
- Use these filter coefficients to find an intermediate filter.
- Perturb the coefficients of this new filter until they are products of only the first few prime numbers (2,3,5,7,11…) AND still within bounds.
- Build factorization tree that give all of the coefficients.

Sample Filter Response



FPC Results

- FPC can be applied to filters of various types and sizes to give a 24-43% savings in the amount of computation.

Filter	Type	Pass-band	Stop-band	PM taps	LS taps	Savings
EX	LPF	0.0-0.25	0.55-1.0	8	10	24%
A	HPF	0.7-1.0	0.0-0.6	57	75	43%
B	LPF	0.0-0.4	0.5-1.0	54	78	28%
C	BPF	0.5-0.6	0-0.35, 0.75-1	54	76	24%
D	HPF	0.475-1.0	0.0-0.4	91	115	36%
E	LPF	0.0-0.2	0.25-1.0	71	121	39%
F	BPF	0.4-0.5	0-0.3, 0.6-1	85	111	30%
G	HPF	0.75-1.0	0.0-0.7	123	161	39%
H	LPF	0.0-0.575	0.625-1.0	131	167	25%

LPF: Low-Pass Filter

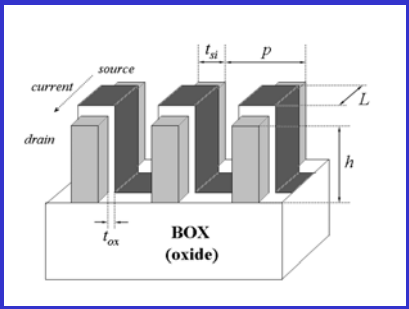
BPF: Band-Pass Filter

HPF: High-Pass Filter

TAPS: Size of filter (# of coefficients)

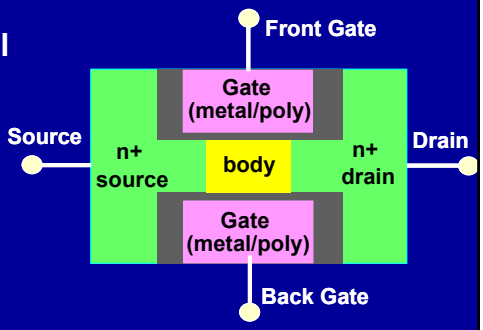
Other Silicon Solutions

- FINFET's and Double Gate MOSFET's
- Better short channel effect
- Intrinsic channel
- Better scalability



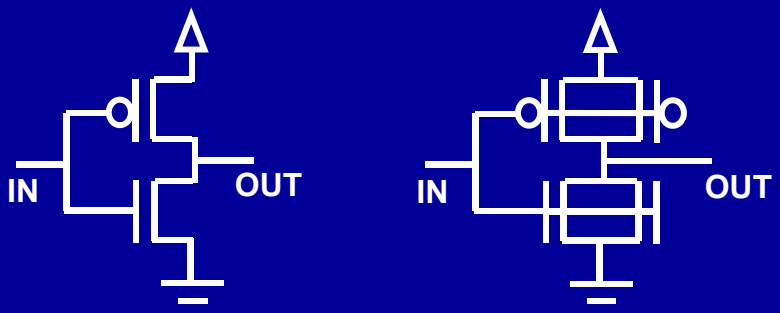
Advantages of Double Gate Devices

- **Short channel effect control**
  - **Better scalability**
  - **Lower subthreshold current**
- **Higher On Current**
- **Near-Ideal Subthreshold slope**
- **Elimination of Vt variation due to Random dopant fluctuation**



DG devices are very promising for circuit design in sub-50nm technology

How do we design circuits in Double Gate technologies ?

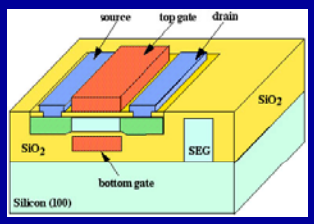


Use the "good" DG devices in place of single gate bulk-CMOS/PD-SOI devices

Are there any new challenges?

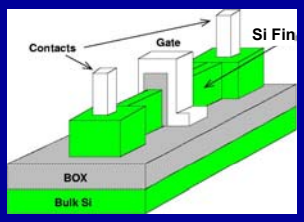
How can we take advantages of DG technologies?

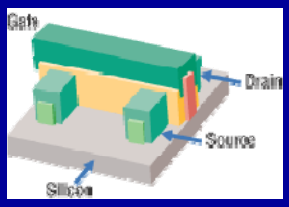
Nano-Scaled Double Gate Devices



DGMOS

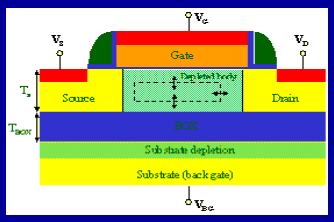
Planar double-gate structure





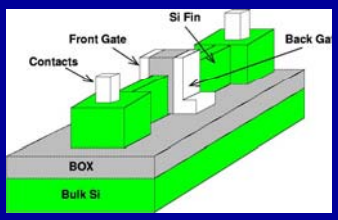
FinFET or Tri-Gate

Quasi-planar DG structure



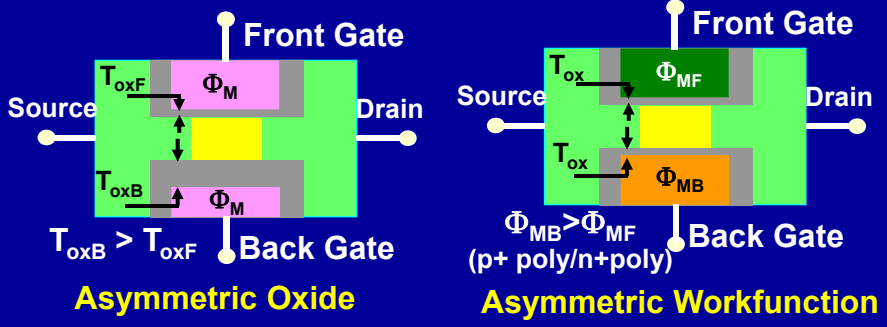
Ground Plane SOI MOS

Shared back gate DG devices



Independent gate FinFET

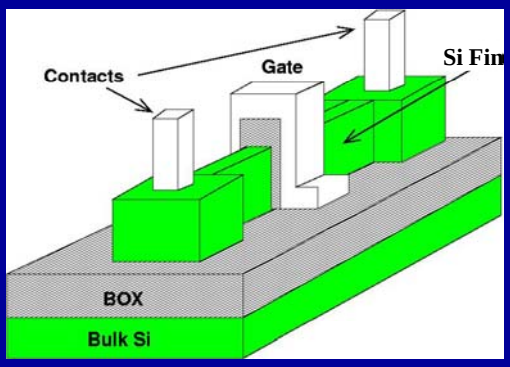
Asymmetric DG Devices

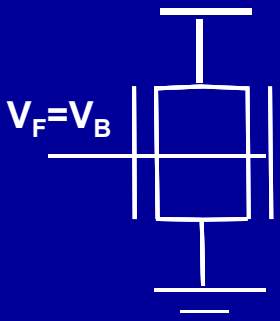


- Asymmetric DG devices has different front and back gate properties
  - Front gate is stronger and has more control over the channel than the back gate

What opportunities do we have for circuit design in DG technologies?

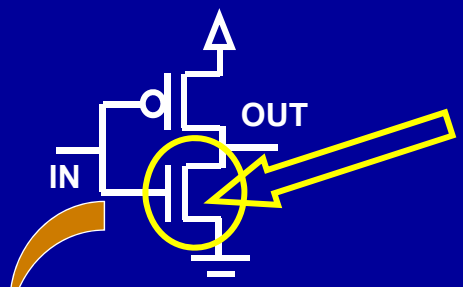
3-Terminal DG Devices

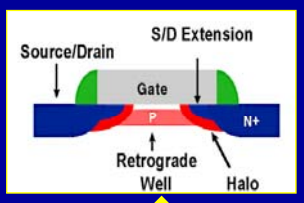


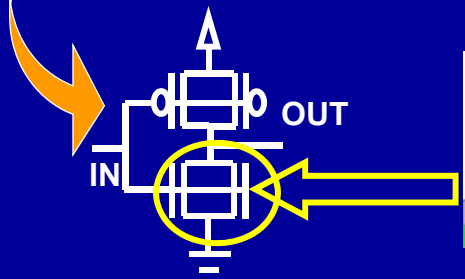


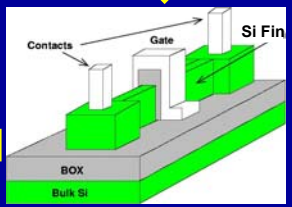
3-Terminal DG devices are essentially "better" single gate devices

3-Terminal DG Devices

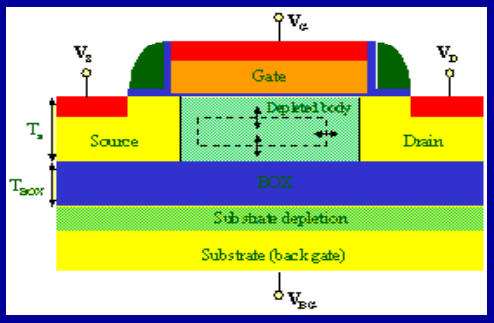


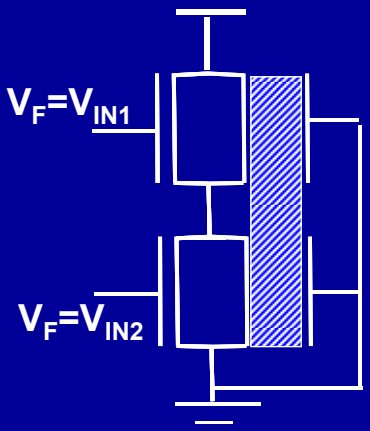






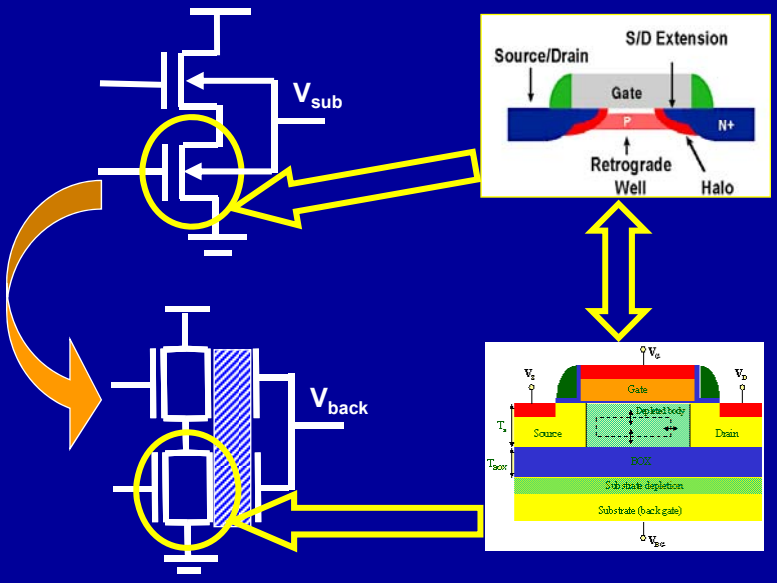
DG Devices with Shared Back Gate (GP-SOI)



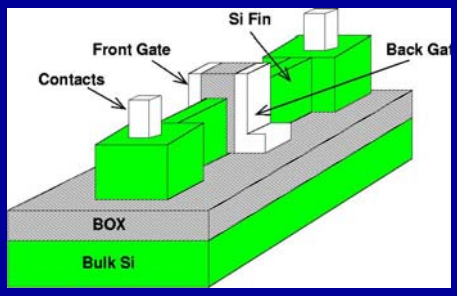


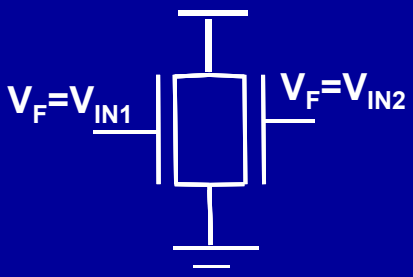
GP-SOI devices are similar to bulk-CMOS devices with substrate biasing option

Back-Biased Circuits in GP-SOI



DG Devices with Independent Front and Back Gates

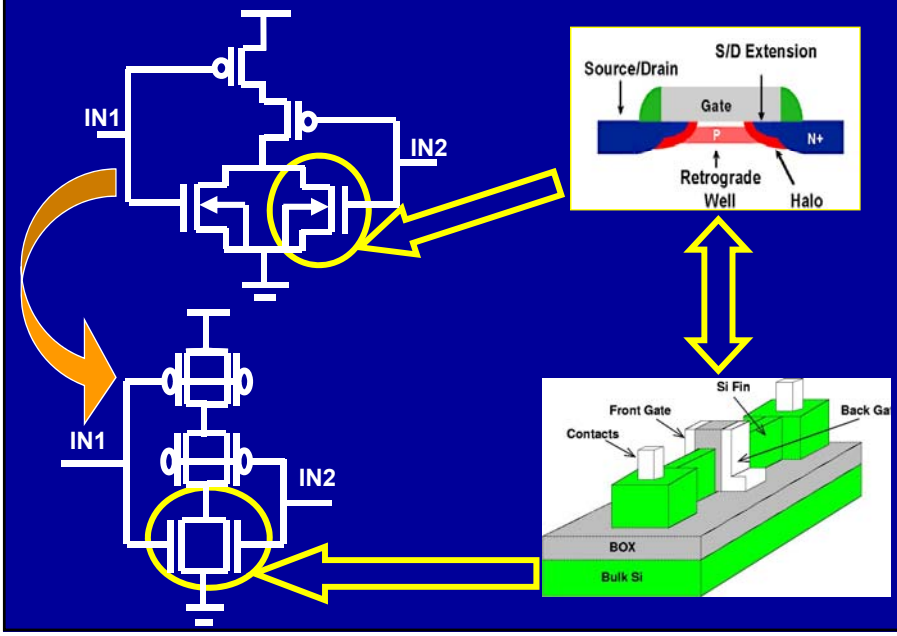




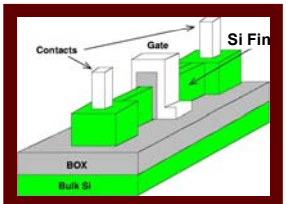
Independent gate devices can have separate input at front and back gates

Unique in DG technologies ➔ Design of new circuit styles

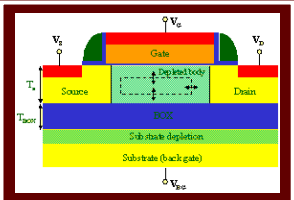
4-Terminal DG Devices



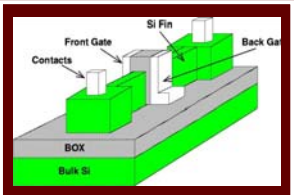
Circuits Design in Double Gate Technologies



3-T DG devices
Directly Translate single gate designs
Width quantization, T_{si} variations,



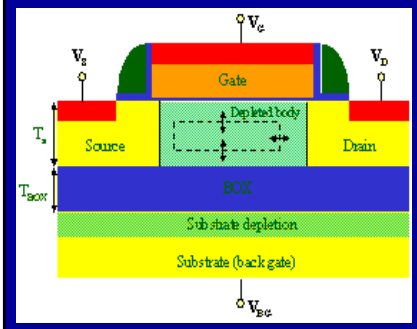
DG devices with shared back gate GP-SOI
Back biased circuits
Dynamic Vt circuits

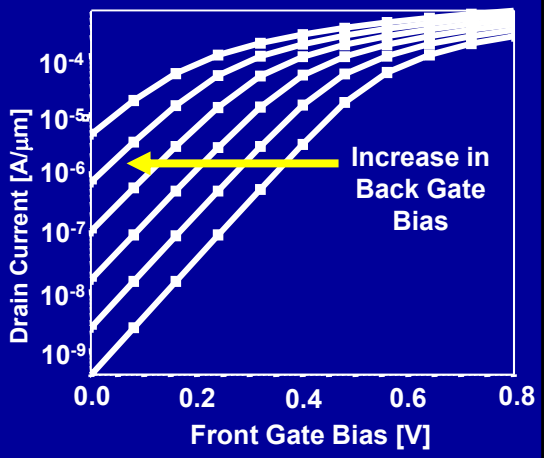


4-T DG Devices
Unique for DG
Isolated front and back gate can have diff. inputs
New circuit styles

Dynamic Vt Circuits in GP-SOI Technology

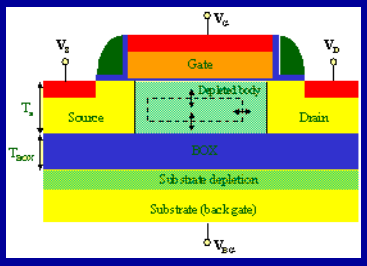
Back Biasing for Dynamic-V_t

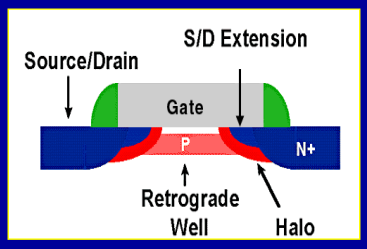


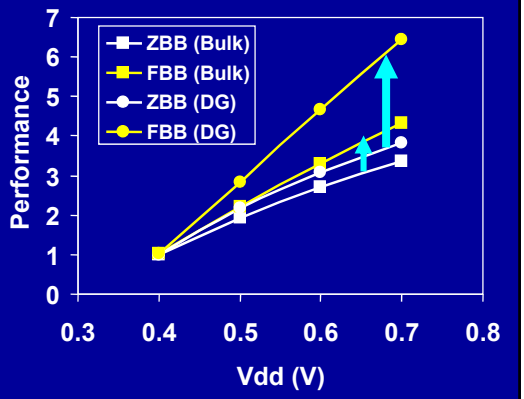


Applying bias to the back gate can modify "on" and "off" currents of the device

Back Biasing for Dynamic-V_t

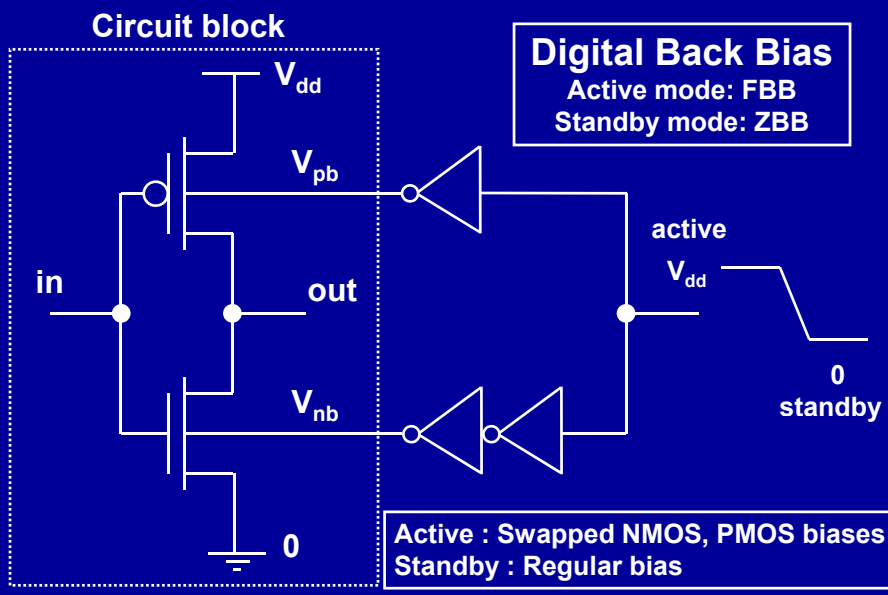




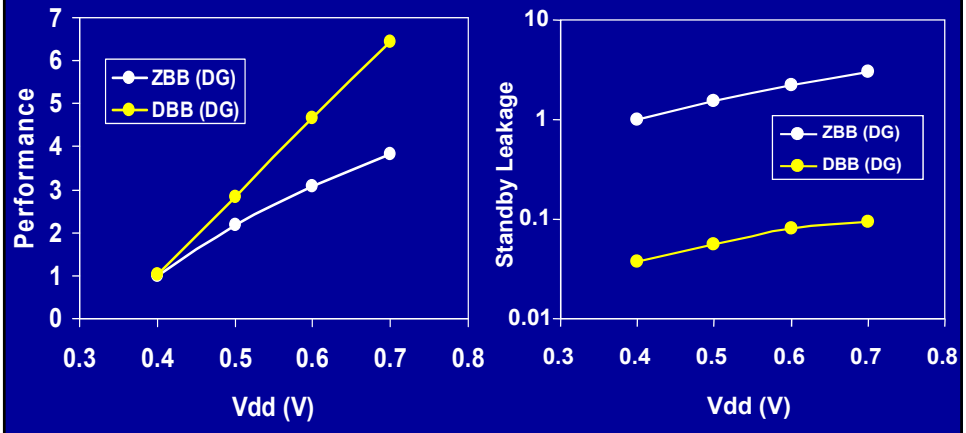


Back-biasing in GP-SOI is more effective than body-biasing in bulk

Digital Back Bias (DBB) in GP-SOI



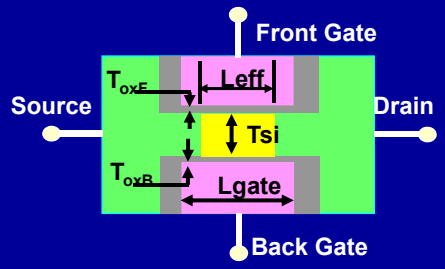
Dynamic Digital Back Biasing with GP-SOI

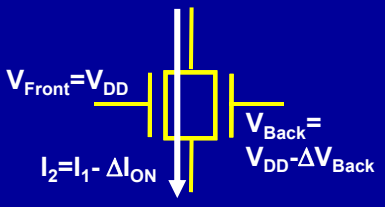


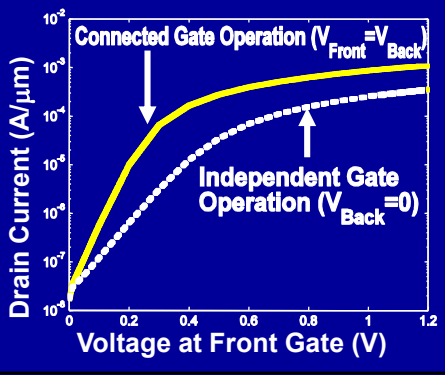
Dynamic DBB improves performance and leakage

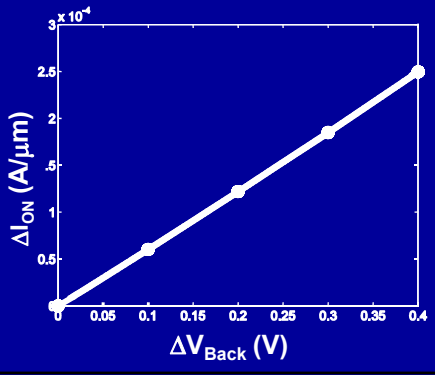
Digital Circuit Design using Independent Gate Operation in Double-Gate Devices

4-Terminal Operation of DG (Symmetric)

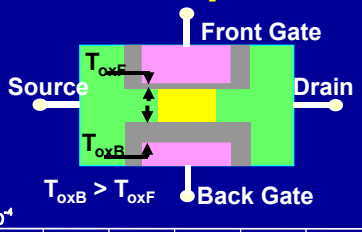


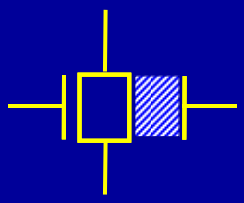


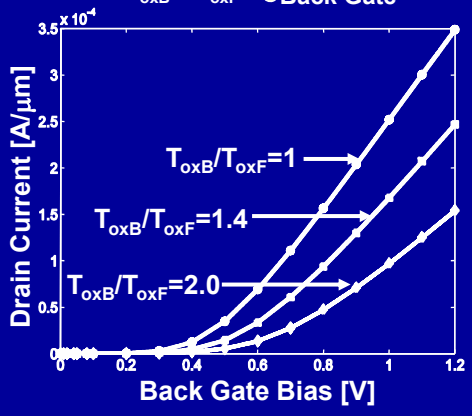


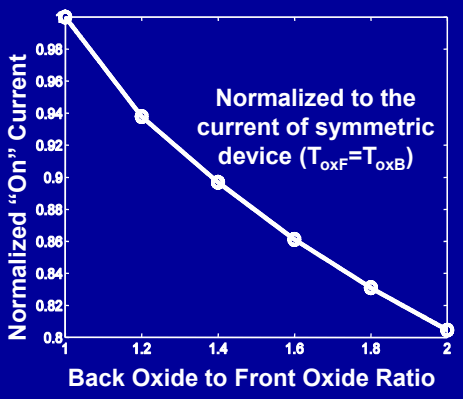


4-Terminal Operation of DG (Asymmetric)

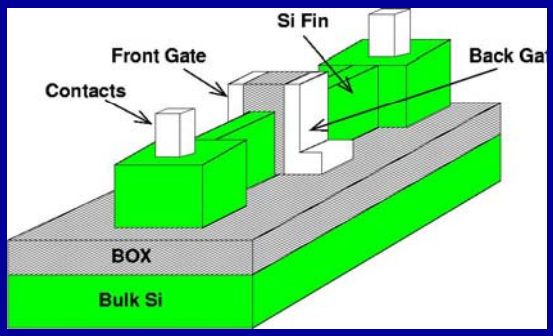








Digital Circuits using Independent Gate Devices



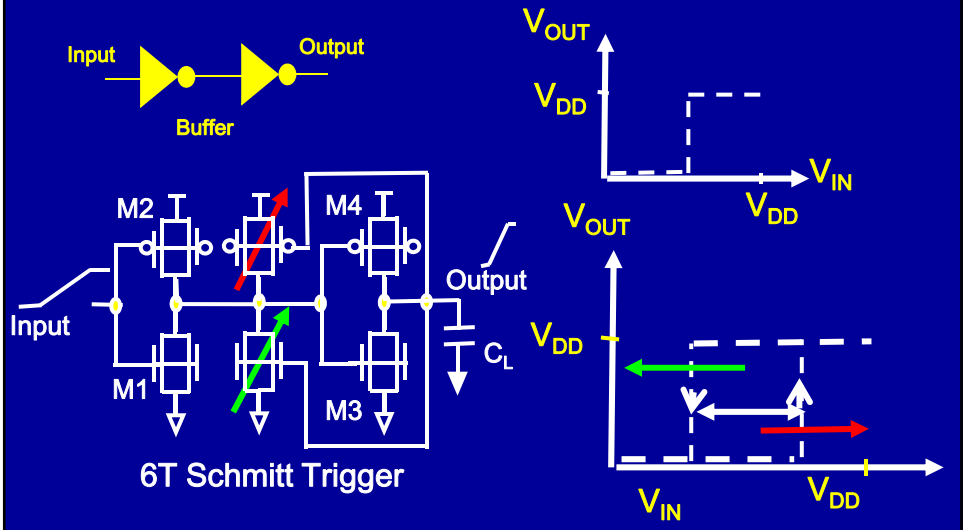
Schmitt Trigger
New circuit style
Lower power and smaller area

Pre-Charge Evaluate Logic
Domino & Skewed CMOS
Better power-delay

Sense-Amplifier
Higher performance
Better robustness
Smaller area

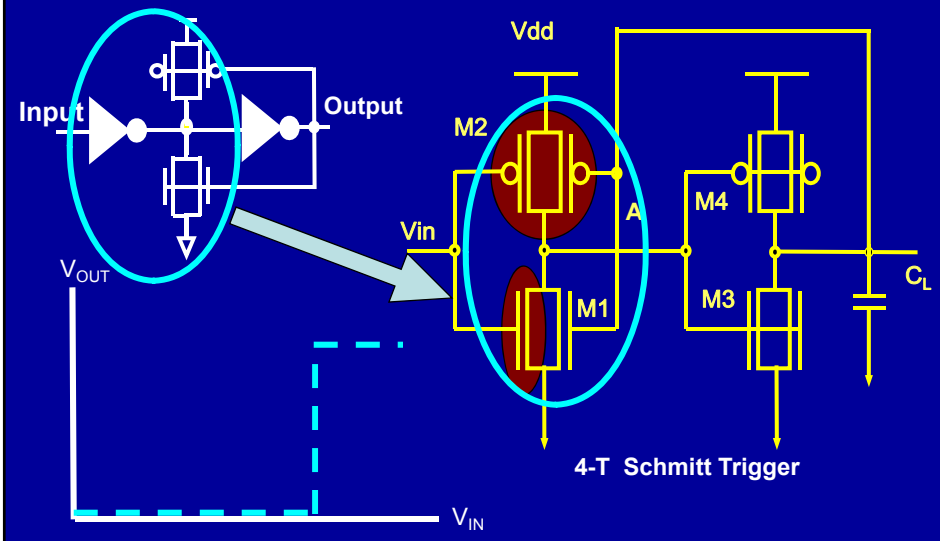
4-Transistor Schmitt Trigger Circuit using Independent Gate Devices

Schmitt Trigger Circuit



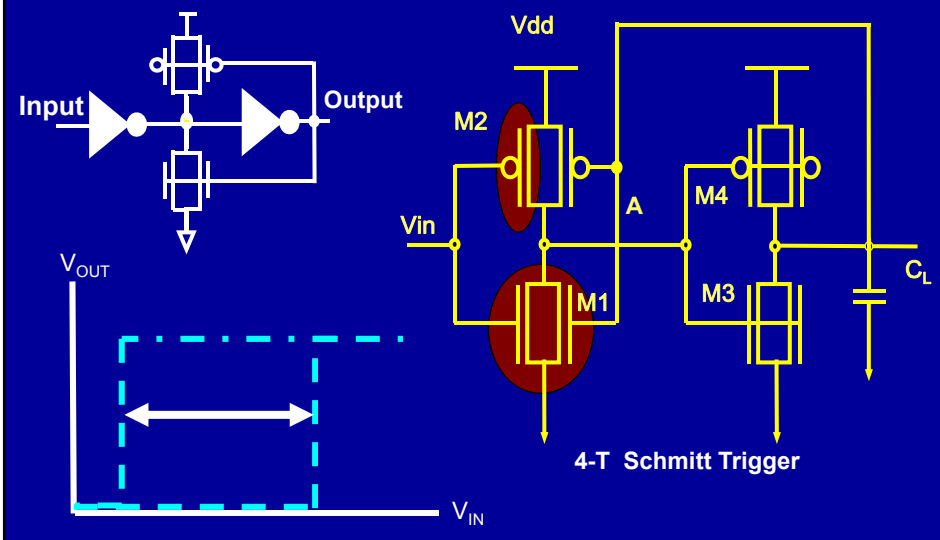
Schmitt-circuit is a high-performance circuit used to shape input pulses and reduce noises

4-T Schmitt Trigger using Independent Gate



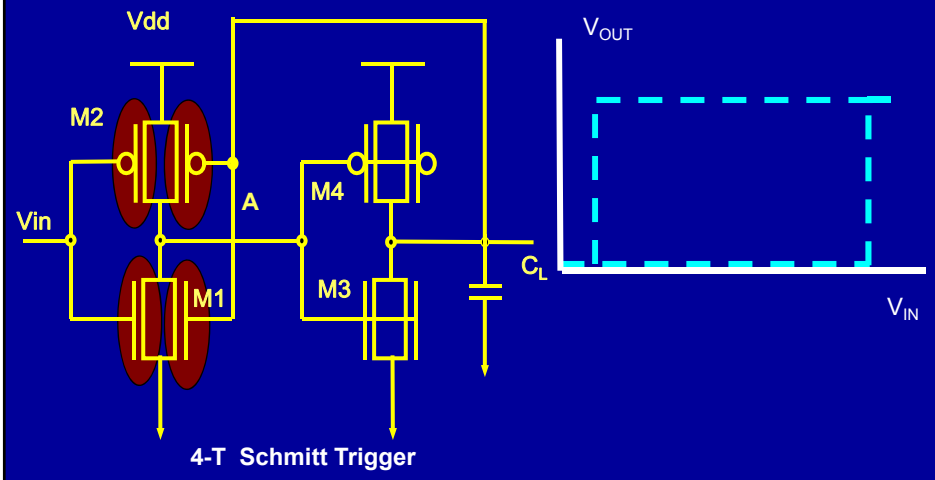
Independent control of the back gate reduces the number of transistors in the Schmitt Trigger circuit

4-T Schmitt Trigger using Independent Gate



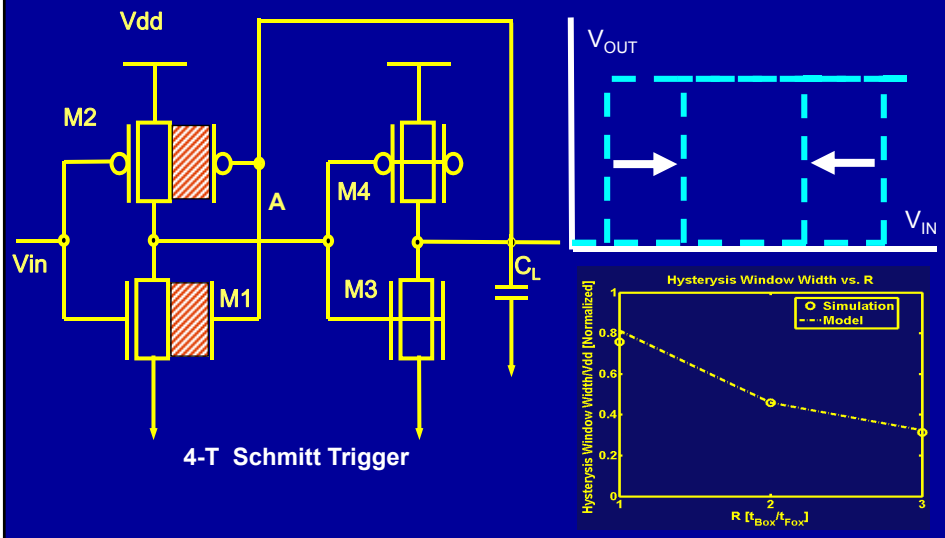
Independent control of the back gate reduces the number of transistors in the Schmitt Trigger circuit

4-T Schmitt Trigger using Independent Gate



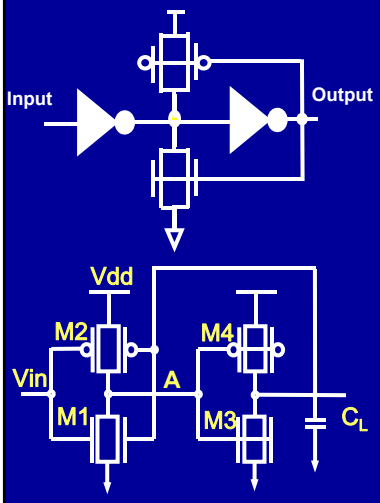
4-T Schmitt Trigger with symmetric devices can have large hysteresis window

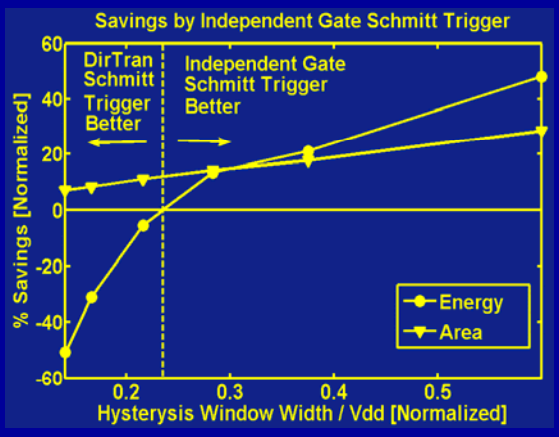
4-T Schmitt Trigger using Independent Gate



Hysteresis window in 4-T Schmitt-Trigger can be designed using Asymmetric DG devices

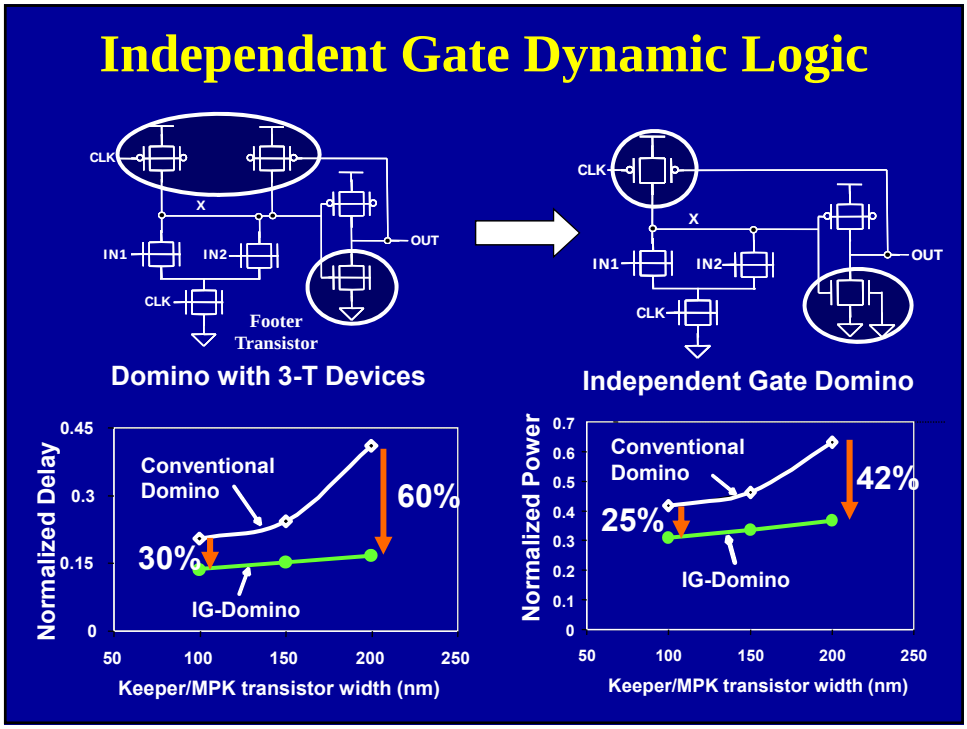
4-T Schmitt Trigger using Independent Gate

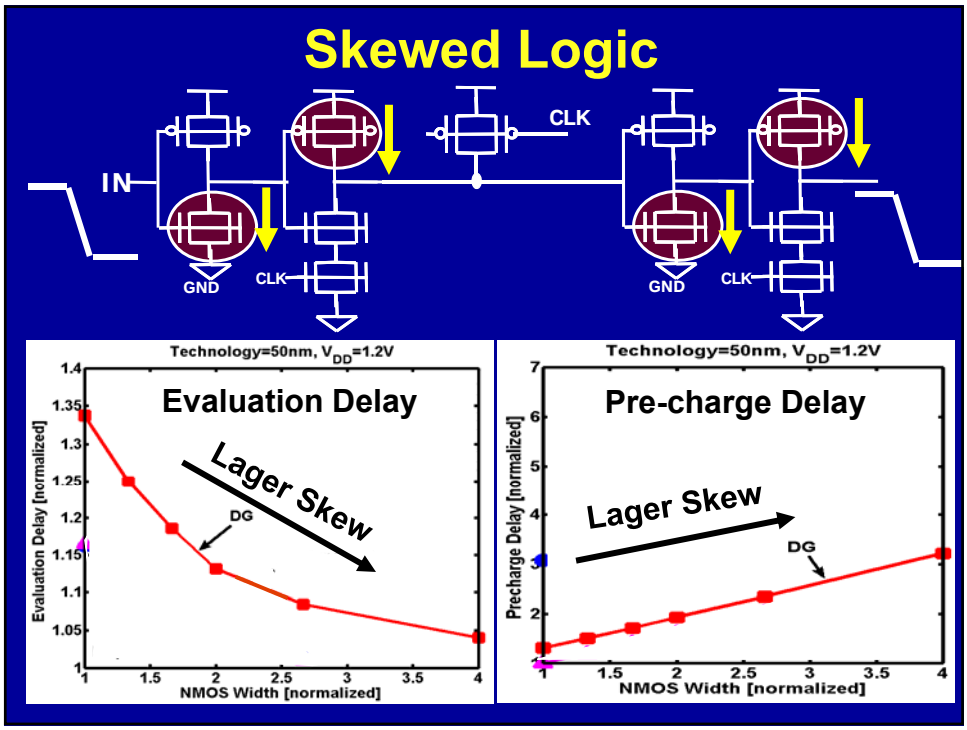




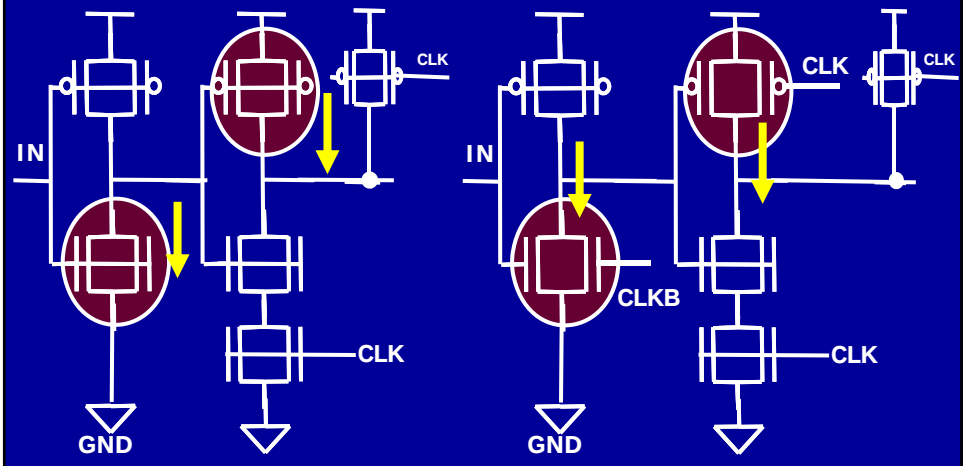
4-T Schmitt-Trigger designed using Asymmetric devices can achieve lower power at high noise immunity corner

Pre-Charge Evaluate Logic Circuits using Independent Gate Devices


Independent Gate Dynamic Logic
CLK
X
IN1
IN2
OUT
CLK
Footer Transistor
Domino with 3-T Devices
Independent Gate Domino
Normalized Delay
Conventional Domino
60%
30%
IG-Domino
Keeper/MPK transistor width (nm)
Normalized Power
Conventional Domino
42%
25%
IG-Domino
Keeper/MPK transistor width (nm)


Skewed Logic
IN
CLK
GND
CLK
Technology=50nm, V_{DD}=1.2V
Evaluation Delay
Lager Skew
DG
Evaluation Delay [normalized]
NMOS Width [normalized]
Technology=50nm, V_{DD}=1.2V
Pre-charge Delay
Lager Skew
DG
Precharge Delay [normalized]
NMOS Width [normalized]

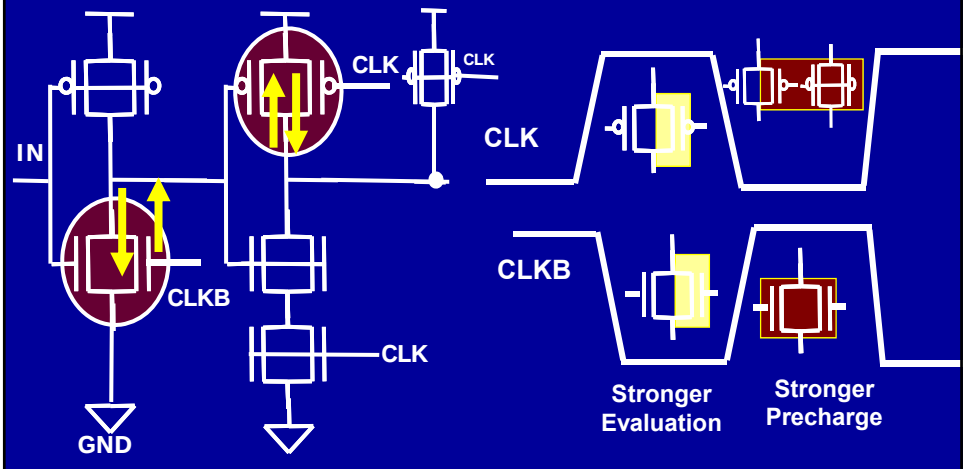
Independent Gate Skewed Logic



DG Skewed Logic

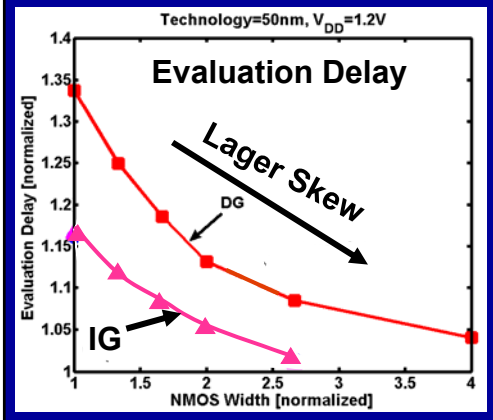
IG Skewed Logic

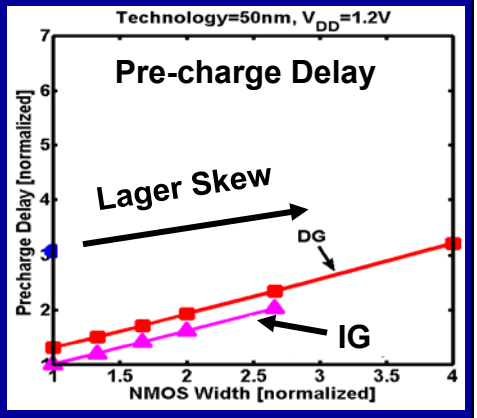
Independent Gate Skewed Logic



IG Skewing reduces evaluation and precharge delay

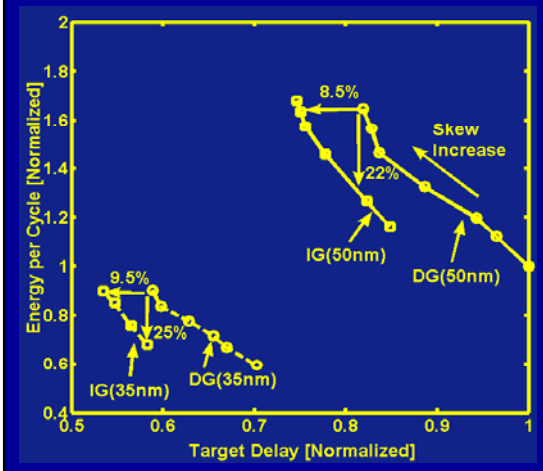
Independent Gate Skewed Logic

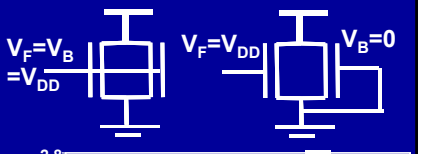


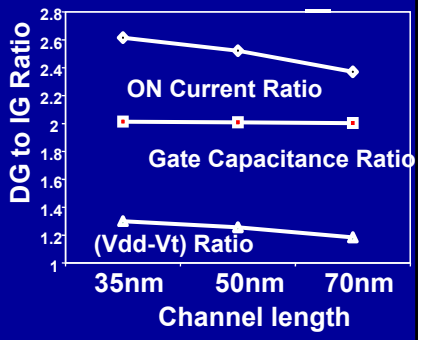


IG Skewing reduces evaluation and precharge delay

Independent Gate Skewed Logic



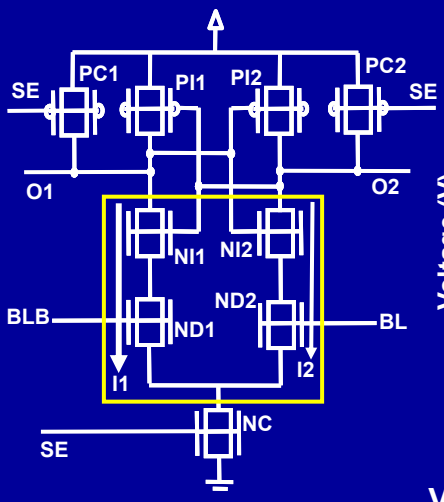




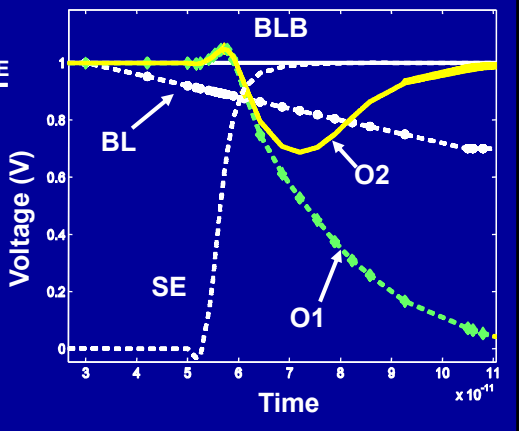
Independent gate operation results in higher performance and lower power in skewed logic

High-Performance Sense-Amplifier using Independent Gate Devices

Sense Amplifier using 3-T Devices

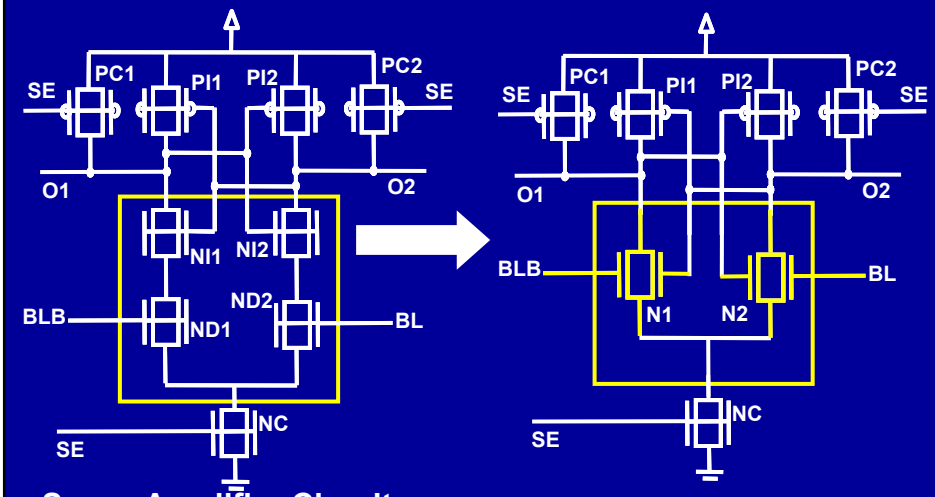


Sense Amplifier Circuit designed using 3-T devices



Voltage difference between BL and BLB produces current difference between ND1 and ND2

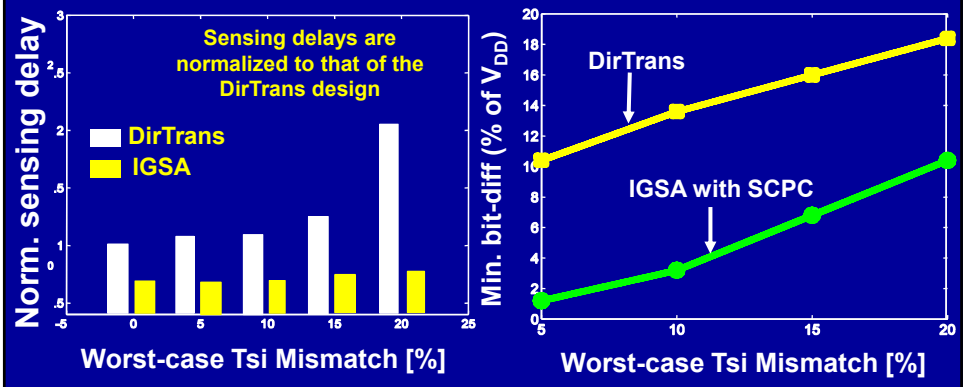
Independent Gate Sense Amplifier



Sense Amplifier Circuit with 3-Terminal Devices

Independent Gate Sense Amplifier Circuit

Process Variation Tolerance of IGSA



- IGSA shows better performance and robustness
  - Lower sensing delay
  - Better tolerance to Tsi mismatch

Conclusions

- Power considerations (both dynamic and leakage) are very important for scaled technologies
- Process parameter variation is also expected to be a major concern. There is a need for leakage statistical design techniques to improve power dissipation and yield
- An integrated approach to design – device/circuit/arch. – is essential for an optimized design
- New failure modes have to be considered for nano-scale designs
  - Process parameter variations
  - High Leakage
  - Soft failures
- New technologies may come to the rescue!
  - DG-MOSFET, FINFET's, CNFET's, Molecular RTD's, ….