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

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

\[
= C_L V_{DD} \int_0^{v_{PP}} dv_{out} = \left[ C_L V_{DD}^2 \right]_{v_{PP}}^{0}
\]

\[
E_c = \int_0^\infty i_{VDD}(t)v_{out}dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{v_{PP}} dv_{out} = \left[ C_L V_{DD}^2 \right]/2
\]

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

\[
P_{\text{dynamic}} = C_L \cdot V_{DD}^2 \cdot f
\]

• Example
  – 1.2\( \mu \) CMOS chip
  – 100 MHz clock rate
  – Average load capacitance of 30 fF/gate
  – 5V power supply

• Power consumption/gate = 75 \( \mu \)W

• Design with 200,000 gates: 15W !

• Pessimistic evaluation: not all gates switch at the full rate

• Have to consider the activity factor \( \alpha \): Effective switching capacitance = \( \alpha C_L \)

• Reducing \( V_{DD} \) has a quadratic effect on \( P_{\text{dynamic}} \)
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

$$P_{SC} = I_{mean} \cdot V_{DD}$$
**Short Circuit Current with Loads**

![Graph showing short circuit current with loads](image)

**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 Consumption: Simple R-C model of Pass pMOS transistor

Consider a pMOS pass transistor with a capacitive load $C$ at the output. 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.

\[ RC \left( \frac{dV_C}{dt} \right) + V_C = \Phi \]

The supply voltage can be expressed as

\[ V_C = \begin{cases} 
0, & t < 0 \\
\frac{V_{dd}}{T}, & 0 \leq t < T \\
V_{dd}, & t > T 
\end{cases} \]

RC Circuits (contd.)

Solving the voltage equation, we’ve

\[ V_C = \begin{cases} 
0, & t < 0 \\
\Phi - \left( \frac{RC}{T} \right) V_{dd} (1 - e^{\frac{-t}{RC}}), & 0 \leq t < T \\
\Phi - \left( \frac{RC}{T} \right) V_{dd} (1 - e^{\frac{-t}{RC}}) e^{\frac{(t-T)}{RC}}, & t > T 
\end{cases} \]

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

\[
\frac{T}{iV_R} dt = \frac{T}{R} (\Phi - V_c)^2 dt
\]

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

\[
= \left( \frac{RC}{T} \right)^2 CV_{dd}^2 \int_0^T (1 - e^{-\frac{t}{RC}})^2 d\left( \frac{t}{RC} \right)
\]

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

The second term of the energy dissipation

\[
\frac{\infty}{T} iV_R dt = \frac{\infty}{T} (\Phi - V_c)^2 dt
\]

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

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

Final energy expression

\[
E_{linear} = \left( \frac{RC}{T} \right) CV_{dd}^2 \left[ 1 - \frac{RC}{T} + \frac{RC}{T} e^{-\frac{T}{RC}} \right]
\]
Energy Dissipation: 2 Cases

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

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

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

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

$$= \frac{1}{2} CV_{dd}^2$$
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 $2V_{\text{ref}}$
- 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
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**

- Memory Cache: 9%
- Synthesizable Logic: 9%
- Rest of circuit: 21%
- Datapath: 4%
- Clock network: 45%

Ref: ISCAS01, Q.K. Zhu

**Power Distribution of McKinley (Itanium) Processor**

- Voltage: 34%
- Package: 2%
- Global Repeater: 3%
- Contention Logic: 5%
- I/O: 3%
- Clock Distribution: 49%
- Switching Logic: 5%

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:

\[ f = \frac{1}{2\pi\sqrt{LC}} \]

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
**Energy-Recovery Results**

Results include clock network and flip-flops

- 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

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

Parallel implementation
Architecture-Driven Voltage Scaling

Pipelined implementation

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

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

\[ f_1 = 1 \Rightarrow Z = 1 \quad f_2 = 1 \Rightarrow Z = 0 \]

N-bit comparator

\[ f_1 = A(n-1) \cdot \overline{B(n-1)} \quad f_2 = \overline{A(n-1)} \cdot B(n-1) \]
Precomputation-Based Optimization for Low Power

Adder-comparator circuit

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

Precomputation using Shannon’s expansion

\[ Z = x_f Z_{x_f} + \overline{x_f} Z_{\overline{x_f}} \]
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 $\frac{V_{HH}^2}{V_{LO}^2}$ 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

<table>
<thead>
<tr>
<th></th>
<th>Pentium Pro</th>
<th>Alpha 21264</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch</td>
<td>22.2%</td>
<td></td>
</tr>
<tr>
<td>Register Alias Table</td>
<td>6.3%</td>
<td></td>
</tr>
<tr>
<td>Reservation Stations</td>
<td>7.9%</td>
<td></td>
</tr>
<tr>
<td>Reorder Buffer</td>
<td>11.1%</td>
<td></td>
</tr>
<tr>
<td>Integer Exec. Unit</td>
<td>14.3%</td>
<td></td>
</tr>
<tr>
<td>Data Cache Unit</td>
<td>11.1%</td>
<td></td>
</tr>
<tr>
<td>Memory Order Buffer</td>
<td>6.3%</td>
<td></td>
</tr>
<tr>
<td>FP Exec. Unit</td>
<td>7.9%</td>
<td></td>
</tr>
<tr>
<td>Global Clock</td>
<td>7.9%</td>
<td></td>
</tr>
<tr>
<td>Branch Target Buffer</td>
<td>4.7%</td>
<td></td>
</tr>
</tbody>
</table>

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

![Clock Gating Diagram]
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.

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

 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

- Little opportunity
- Already done
- Info not available ahead of time
- Info available at IQ
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

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**

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

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

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

< Coefficient Decomposition >

\[ c = 111010001100 \]

\[ c = 2^9(111) + 2^7(1) + 2^2(11) \]

alphabet set = \{1, 11, 111\}

Alphabets - chosen basic bit sequences

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

\[ c \cdot x = 111010001100 \cdot x \]

\[ c \cdot x = 2^9(0111 \cdot x) + 2^7(0001 \cdot x) + 2^2(0011 \cdot x) \]

if 0111 \cdot x, 0001 \cdot x and 0011 \cdot x are available, \( c \cdot x \) can be significantly simplified as add and shift operation

**Shared Multiplier Architecture**

[Diagram showing the architecture of the shared multiplier]

Input \( x \)

Precomputer bank (8 alphabets)

Coefficient .... 11101000

MUX (8:1)

1000 \( \Rightarrow \) 0001

SHIFTER

111x (<<4)

AND gate

1100x

Select unit

1x (<<3)

AND gate

1000x

Product 11101000 \( \cdot x \)

Adder

1110x (<<4)

Adder
**16×16 Shared Multiplier Implementation**

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

### Table: Performance Comparison

<table>
<thead>
<tr>
<th></th>
<th>Precomputer</th>
<th>Select units &amp; Adders</th>
<th>WTM</th>
<th>CSAM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delay</td>
<td>6.923 ns</td>
<td>11.231 ns</td>
<td>16.638 ns</td>
<td>23.398 ns</td>
</tr>
<tr>
<td>Power</td>
<td>18.86 mW</td>
<td>18.91 mW</td>
<td>22.80 mW</td>
<td>21.78 mW</td>
</tr>
<tr>
<td>Area</td>
<td>162340 µm²</td>
<td>252120 µm²</td>
<td>241000 µm²</td>
<td>175640 µm²</td>
</tr>
</tbody>
</table>

* CMU library (0.35 µm technology)

**FIR filter using Shared Multiplier**

- Computations \( a_k \cdot x \) 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

\[ y(n) = \sum_{i} c_i \cdot x(n-i) \]

**Numerical Results**

<table>
<thead>
<tr>
<th>Filter</th>
<th>FIR filter using Shared Multiplier</th>
<th>FIR filter using Wallace Tree</th>
<th>FIR filter using Carry Save Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clock Cycle</td>
<td>13 ns</td>
<td>18 ns</td>
<td>25 ns</td>
</tr>
<tr>
<td>Power</td>
<td>398.4 mW</td>
<td>412.2 mW</td>
<td>401.1 mW</td>
</tr>
<tr>
<td>Area</td>
<td>(4.41 \times 10^6 \mu m^2)</td>
<td>(3.87 \times 10^6 \mu m^2)</td>
<td>(3.15 \times 10^6 \mu m^2)</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Filter</th>
<th>Radix-16 CSHM (Synth.)</th>
<th>CSHM (Custom)</th>
<th>WTM (Synth.)</th>
<th>CSM (Synth.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption at 10 ns CLK (mW)</td>
<td>226.1</td>
<td>286.6</td>
<td>344.3</td>
<td>357.1</td>
</tr>
<tr>
<td>Min. CLK Cycle (ns)</td>
<td>5</td>
<td>7</td>
<td>8.5</td>
<td>10</td>
</tr>
<tr>
<td>Area ((\times ) mm²)</td>
<td>4.1</td>
<td>5.0</td>
<td>4.4</td>
<td>4.1</td>
</tr>
</tbody>
</table>
DFE using Shared Multiplier

- Coefficient Update

\[ C_{k+1} = C_k + \text{StepSize} \times \text{Error} \times V_k \]

<table>
<thead>
<tr>
<th>Filter</th>
<th>Clock Cycle</th>
<th>Area</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFE using CSHM</td>
<td>96.67 ns</td>
<td>4.06 \times 10^7 \mu m^2</td>
</tr>
<tr>
<td>DFE using WTM</td>
<td>112.73 ns</td>
<td>2.51 \times 10^7 \mu m^2</td>
</tr>
<tr>
<td>DFE using CSAM</td>
<td>117.23 ns</td>
<td>2.51 \times 10^7 \mu m^2</td>
</tr>
</tbody>
</table>

* CMU library (0.35 \mu m technology)

DCT: Shared Multiplier Application

DCT (Discrete Cosine Transform)

- The number of \textit{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 \textit{Precomputer bank} and \textit{Select unit}.
- DCT with the modified coefficients generates acceptable quality of image
Shared Multiplier Application

DCT (Discrete Cosine Transform)

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

Note the symmetry of the DCT coef. matrix

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

\[ Z = T^T \cdot X = T^T X \]

DCT (Background)

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

\[ Z = T^T \cdot X = T^T X \]

**Even DCT**

\[
\begin{align*}
    z_0 &= d + d + d + d \\
    z_2 &= b + f - f - b \\
    z_4 &= d - d - d - d \\
    z_6 &= f - b + f - b
\end{align*}
\]

\[
\begin{align*}
    x_0 &= x_0 \\
    x_1 &= x_1 + x_1 \\
    x_2 &= x_2 + x_2 \\
    x_3 &= x_3 + x_3
\end{align*}
\]

**Odd DCT**

\[
\begin{align*}
    z_1 &= a + c + e + g \\
    z_3 &= c - g - a - e \\
    z_5 &= e + a - g - c \\
    z_7 &= g - e + c - a
\end{align*}
\]

\[
\begin{align*}
    x_0 &= x_0 \\
    x_1 &= x_1 - x_1 \\
    x_2 &= x_2 - x_2 \\
    x_3 &= x_3 - x_3
\end{align*}
\]

Image data

\[ X = \begin{bmatrix}
    x_{00} & x_{01} & x_{02} & x_{03} & x_{04} & x_{05} & x_{06} & x_{07} \\
    x_{10} & x_{11} & x_{12} & x_{13} & x_{14} & x_{15} & x_{16} & x_{17} \\
    x_{20} & x_{21} & x_{22} & x_{23} & x_{24} & x_{25} & x_{26} & x_{27} \\
    x_{30} & x_{31} & x_{32} & x_{33} & x_{34} & x_{35} & x_{36} & x_{37} \\
    x_{40} & x_{41} & x_{42} & x_{43} & x_{44} & x_{45} & x_{46} & x_{47} \\
    x_{50} & x_{51} & x_{52} & x_{53} & x_{54} & x_{55} & x_{56} & x_{57} \\
    x_{60} & x_{61} & x_{62} & x_{63} & x_{64} & x_{65} & x_{66} & x_{67} \\
    x_{70} & x_{71} & x_{72} & x_{73} & x_{74} & x_{75} & x_{76} & x_{77}
\end{bmatrix}
\]

\[ Z = TX^T \]

\[ Z = TX^T \]

Transpost

\[ TX = Z \]

\[ TX = Z \]

\[ T \]
DCT using Shared Multiplier

\[ X = \frac{c(\lambda)c(t)}{4} \sum x_l \cos \left( \frac{(2l+1)\pi}{16} x \right) \cos \left( \frac{(2j+1)\pi}{16} y \right) \]

- \( Z = Tz^d, \ X = TZ^d \)

**Even DCT**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
<th>Binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.50</td>
<td>00110000</td>
</tr>
<tr>
<td>1</td>
<td>0.47</td>
<td>00110100</td>
</tr>
<tr>
<td>2</td>
<td>0.44</td>
<td>00111010</td>
</tr>
<tr>
<td>3</td>
<td>0.41</td>
<td>00111110</td>
</tr>
<tr>
<td>4</td>
<td>0.38</td>
<td>00101000</td>
</tr>
<tr>
<td>5</td>
<td>0.34</td>
<td>00111001</td>
</tr>
<tr>
<td>6</td>
<td>0.28</td>
<td>00100100</td>
</tr>
<tr>
<td>7</td>
<td>0.24</td>
<td>00100010</td>
</tr>
<tr>
<td>8</td>
<td>0.20</td>
<td>00010000</td>
</tr>
</tbody>
</table>

**Odd DCT**

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Value</th>
<th>Binary code</th>
</tr>
</thead>
<tbody>
<tr>
<td>0'</td>
<td>0.50</td>
<td>00110000</td>
</tr>
<tr>
<td>1'</td>
<td>0.47</td>
<td>00110100</td>
</tr>
<tr>
<td>2'</td>
<td>0.44</td>
<td>00111010</td>
</tr>
<tr>
<td>3'</td>
<td>0.41</td>
<td>00111110</td>
</tr>
<tr>
<td>4'</td>
<td>0.38</td>
<td>00101000</td>
</tr>
<tr>
<td>5'</td>
<td>0.34</td>
<td>00111001</td>
</tr>
<tr>
<td>6'</td>
<td>0.28</td>
<td>00100100</td>
</tr>
<tr>
<td>7'</td>
<td>0.24</td>
<td>00100010</td>
</tr>
<tr>
<td>8'</td>
<td>0.20</td>
<td>00010000</td>
</tr>
</tbody>
</table>

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

**8bit DCT Coefficients**

**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 Multiplier: 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:

\[ C_k = C_{k-1} + \delta_{k-1/k} \]

- The delta’s are called the “First Order Differences”

- By expanding the expression for consecutive Y’s and subtracting we get:

\[ Y_{j+1} = Y_j + C_0 X_{j+1} + \sum_{k=1}^{N-1} \delta_{k-1/k} X_{j-k+1} - C_{N-1} X_{j-N+1} \]

- The “First Order Partial Sum” is defined as:

\[ \sum_{k=1}^{N-1} \delta_{k-1/k} X_{j-k+1} = \left\{ S_P \right\}_{j=N+1} \]

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:
  \[
  Y_{j+1} = Y_j + C_0 X_{j+1} + \sum_{k=1}^{N-1} \left( \delta^k X_{j-k+1} \right)
  \]

- 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

![Bar chart showing percentage savings for different bit widths (16, 24, 32, 48, 64 bits) for $m=1$ and $m=2$.]
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 $K = \{0, 1, 2, 3, \ldots, M-1\}$, $\Delta c_i = c_{i+1} - c_i$)

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

Obtaining the Reduced Form

Step 1: Move the delay elements into branches

Overhead Adders
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 \( C_1 = F_1 \cdot F_2 \) & \( C_2 = F_1 \cdot F_3 \), the value of \( F_1 \cdot x(n) \) 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:
  \[ C_i = \prod_{k=1}^{Q(n)} \left\{ f_n(k) \right\}^{p_n(k)} \]
- 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: \( f = \{2,3,5,7,11,\ldots\} \)
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.

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

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 are essentially “better” single gate 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,

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
Applying bias to the back gate can modify “on” and “off” currents of the device.
Digital Back Bias (DBB) in GP-SOI

Circuit block

V_{dd}

V_{pb}

V_{nb}

in

out

Active: Swapped NMOS, PMOS biases

Standby: Regular bias

Digital Back Bias

Active mode: FBB
Standby mode: ZBB

Active : Swapped NMOS, PMOS biases

Standby : Regular bias

Dynamic Digital Back Biasing with GP-SOI

Performance

V_{dd} (V)

Dynamic DBB improves performance and leakage

Standby Leakage

V_{dd} (V)
Digital Circuit Design using Independent Gate Operation in Double-Gate Devices

4-Terminal Operation of DG (Symmetric)

\[ V_{\text{Front}} = V_{\text{DD}} \]

\[ V_{\text{Back}} = V_{\text{DD}} - \Delta V_{\text{Back}} \]

\[ I_2 = I_1 - \Delta I_{\text{ON}} \]

Voltage at Front Gate (V)

Drain Current (A/\mu m)

\[ \Delta I_{\text{ON}} (A/\mu m) \]

\[ \Delta V_{\text{Back}} (V) \]
4-Terminal Operation of DG (Asymmetric)

- Front Gate
- Drain Source
- ToxF
- ToxB
- ToxB/ToxF=1
- \[ \frac{r}{A/\mu m} \]
- Normalized to the current of symmetric device (ToxF=ToxB)

On Current

- Back Gate
- ToxB > ToxF
- ToxB/ToxF=1.4
- ToxB/ToxF=2.0

Back Gate Bias [V]

- Drain Current [A/\mu m]

- Normalized "On" Current

Back Oxide to Front Oxide Ratio

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

6T Schmitt Trigger

Schmitt-circuit is a high-performance circuit used to shape input pulses and reduce noises
Independent control of the back gate reduces the number of transistors in the Schmitt Trigger circuit.
4-T Schmitt Trigger with symmetric devices can have large hysteresis window

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

Domino with 3-T Devices

IG-Domino

Conventional Domino

Normalized Delay

Keeper/MPK transistor width (nm)

Conventional Domino

IG-Domino

Skewed Logic

Evaluation Delay

Pre-charge Delay

Technology=50nm, V_{DD}=1.2V

Lager Skew

NMOS Width [normalized]
Independent Gate Skewed Logic

DG Skewed Logic

IG Skewed Logic

IG Skewing reduces evaluation and precharge delay
Independent Gate Skewed Logic

IG Skewing reduces evaluation and precharge delay

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

Voltage difference between BL and BLB produces current difference between ND1 and ND2
Independent Gate Sense Amplifier

Sense Amplifier Circuit with 3-Terminal Devices

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, ….