RT and Algorithmic-level optimization for Low power

- A very new area of research
- Target of the optimization is dynamic (switching) component of power.

- RT-level power optimization
  * architecture-driven voltage down scaling
  * Reduction of total switching capacitance of the design

- Architecture-driven voltage down scaling

\[ P = \frac{1}{2} E_{SW} \cdot C_l \cdot V_{dd}^2 \cdot f \]

Try to reduce $V_{dd}$.

But, this affects circuit speed.

Circuit speed

Three different delays:

- $t_f$
- $t_r$
- $t_{PHL}$
\[ t_{PLH} = \frac{C_L}{\beta_p (V_{DD} - V_{TP})} \left[ \frac{2V_{TP}}{V_{DD} - V_{TP}} + \ln \left( \frac{3V_{DD} - 4V_{TP}}{V_{DD}} \right) \right] + \left[ 1 + \frac{2V_{TP}}{V_{DD}} \right] \frac{T}{6} \]

where \( \beta_p = \frac{W_p}{L_p} \frac{E_{ox} M}{t_{ox}} \)

\[ t_{PLH} \text{ can be approximated by } \frac{C_L V_{DD}}{k (W/L) (V_{DD} - V_{TP})^2} \]

when \( V_{DD} \to V_{TH} \Rightarrow T \) dramatically
The increase in delay due to $V_{dd}$ reduction must be compensated through architecture modification.

Use parallel/pipeline architecture to maintain the same throughput.

Example:

$$V_{ref} = 5V$$

Worst delay = 25 ns

$T_{ref} = 25$ ns.
\[ P_{\text{ref}} = \frac{1}{2} C_{\text{ref}} V_{\text{ref}}^2 f_{\text{ref}} \]

(Since activity is not \( p \), \( A, B, C \) are changed, \( \Rightarrow \) ignored. Random #s)

where \( C_{\text{ref}} \) is total effective capacitance of the ref.

Use parallel architecture.

The speed can be halved.

\( \Rightarrow \) \( \text{double speed clock.} \)

\( \circ \) \( \text{CLK can be reduced to } T_{\text{ref}} > 50 \text{ ns} \)

With the same throughput.
How much can the $V_{DD}$ be reduced?

$\Rightarrow V_{par} = 2.9V = 0.5PV_{Ref}$

$C_{par} \approx 2.15C_{Ref}$

$\Rightarrow P_{par} = \frac{1}{2} C_{par} V_{par}^2 f_{par} = \frac{1}{2} (2.15 C_{Ref}) (0.5P_{Ref})^2 \frac{f_{Ref}}{2}$

$\approx 0.36 P_{Ref}$

*How much parallelism can be used?*

$\Rightarrow$ can be 4 times slower!?

Can $V_{DD}$ be highly reduced?
Problem: When $V_{dd}$ approaches $V_T$, 
- delay $\uparrow$ rapidly.
- much more duplication of the circuit is required.
- power $\uparrow$ more than power saved!!
- parallelism does not pay off when $V_{dd} \rightarrow V_T$.

Pipeline implementation

- The critical path has been reduced.
- Time = $\max [T_{adder}, T_{comparator}]$
  - worst delay of adder & comparator, respectively.
Assume \( T_{adder} = T_{comparator} = 12.5 \text{ ns} \)

The clock period can be 12.5 ns.

But, we just like to keep the same speed

\[ \text{Clock speed is still } 25 \text{ ns} \]

\[ \text{Clock speed of each one can be slowed by } 2 \text{ times (25 ns)} \]

\[ \text{Voltage supply of each } \text{circuit can be reduced} \]

Again, \( V_{pipe} = 2.9 \text{V} = 0.5 \times \text{Ref} \) by checking the curve

\[ C_{pipe} = 1.15 \times C_{ref} \text{ (increased slightly only)} \]

\[ f_{pipe} = f_{ref} \text{ (still work with the same speed)} \]
\[ P_{\text{pipe}} = \frac{1}{2} C_{\text{pipe}} V_{\text{pipe}}^2 f_{\text{pipe}} \]

\[ = \frac{1}{2} \left( 1.15 \text{Ref} \right) \left( 0.5 \times V_{\text{Ref}} \right)^2 f_{\text{ref}} \]

\[ \approx 0.39 \text{Pref} \]

- Power saved is about the same as in parallel architecture.
- But, area penalty is much smaller.
- Further, pipelining causes reduction of capacitance depth.
  - \( \Rightarrow \) reduce the power dissipation by glitches.
- Pipelining is preferred if possible.

- Merging parallel & pipelining \( \Rightarrow \) reduce even more significant power

\[ P_{\text{pp}} \approx 0.2 \text{Pref} \]
Effective Capacitance Reduction

* Data representation

Try to find a good data representation so the effective capacitance can be reduced.

Average switching activity x total capacitive load

Two's complement representation

\[ 2 - 3 = ? \]
\[ 2 + (-3) = -1. \]
\[ \begin{array}{c}
010 \\
101 \\
111 \\
\end{array} \]
\[ \begin{array}{c}
-1 \\
-2 \\
-3 \\
1 \\
2 \\
3 \\
4 \\
000 \\
001 \\
010 \\
011 \\
100 \\
101 \\
110 \\
111 \\
\end{array} \]

Very useful in DSP and up chip

Make arithmetic operations simple.

Sign-bit extension.
Example:

![Diagram of 3 bits of dynamic range being extended to 8 bit bus]

111 = -1 must be extended to 11111111.

- This happens when dynamic range << actual bit width.
- If the numbers being transferred or manipulated are switching around zero frequently, do not use the entire bit width.

\[ +1 = 00000001_2 \quad \text{and} \quad -1 = 11111111_2 \]

- However, if the #s are represented by sign and magnitude notation, \[ +1 = 00000001_2 \quad \text{and} \quad -1 = 10000001_2 \]

Very small # of switchings.
Conclusion: The degree of correlation between data being processed influences the power dissipation.

Case study:

The dynamic range of numbers is mostly $[-2^{11}, 2^{12}]$.

⇒ Only 12 bits out of the 16 bits are used.

Problems: Try to determine the transition probability, $p$, for each of the 16 bus lines for
1. 2's complement representation
2. Sign and magnitude representation.

Data correlations
1. No correlation
2. Positive correlation: $p$ (2 consecutive #s have same sign) ↑
3 Negative Correlation: \( p(2 \text{ consecutive } \#s \text{ have same sign}) \) ↓

- Data distribution: Gaussian distribution
  - Mostly, in these areas: \( 3\sigma = 2'' \)
  - 93.32\% that data will be distributed

\[ \leq 2'' \text{ numbers} \]

\( p \): transition probability

\[ \begin{array}{c}
0.5 \\
0.6 \\
0.7 \\
0.8 \\
0.9 \\
1.0
\end{array} \]

\( x \): bit number

\[ \begin{array}{ccccccccccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15
\end{array} \]

\[ \rightarrow \text{N: bit } \#
\]

2's complement representation

\[ \begin{array}{c}
\text{sign extension}
\end{array} \]

12 bits of dynamic range.
The biggest win of sign and magnitude system is

1. Dynamic range of the represented numbers is small
2. Highly negatively correlated between the consecutive numbers.
Conclusion: sign & magnitude representation $\Rightarrow$ good for data transfer devices, e.g., buses.

2's complement representation $\Rightarrow$ good for arithmetic ckt.

- Simple algorithm $\Rightarrow$ simple ckt $\Rightarrow$ smaller ckt

$\Rightarrow$ lower power dissipation.

Best solution:

Diagram:

- Hardware devices
- 2's complement
- Buses
- Sign & magnitude
- Communication
- Encoder
- Decoder
- Logic

On-chip or off-chip
Bus Encoding

- Try to use signed magnitude representation for communication.
- Can be further improved by bus encoding.

**Motivation:**
- Power dissipation at I/O pins accounts for 5% of overall switching power.
- Pin has very high capacitance.

- Should try to reduce the switching activities of I/O pins by encoding.

**Encoding methods:**

*Active high encoding:*

- High-level voltage: 1
- Low-level voltage: 0.

*Transition-based encoding:*

- Voltage change: 1
- No voltage change: 0.
Transition-based encoding

Assume: $p(0) > p(1)$

No temporal correlation between consecutive values

Switching activities:

**Active-high encoding:**

Average # of transitions / clock cycle

$p(0)p(1) + p(1)p(0) = 2p(1)(1-p(1))$

if $p(1) = \frac{1}{2}$

$p(0) = \frac{3}{8}$

$\Rightarrow$ Switching activity is $\frac{3}{8}$, average
Transition based encoding:

Transition required only when there is a logic one.

Average switching activity is \( p(1) \).

\( \frac{1}{4} \) in this case

\( 0.33\% \) less than the active-high encoding.

* Transition-based encoding is good when inputs are strongly not equiprobable.

* If \( p(0) > p(1) \) for a line, then the switching activity of the line is smaller (than active high code).

* Try to use limited-weight code to make sure that \( p(0) > p(1) \) for each bit line.

\[ \begin{aligned} &\text{N-bit} \\ &\uparrow \\ &\text{K-limited-weight code: There are } k \text{ 1's in the } n \text{ bits.} \end{aligned} \]

\( \text{can represent } \binom{n}{k} \text{ data} \)
Example: $k = 1$.

\[
\begin{align*}
0 & : 0 \quad 0 \quad 0 \quad 1 \quad \text{Code word 1} \\
0 & : 0 \quad 0 \quad 1 \quad 0 \quad \text{Code word 2} \\
8 & : 1 \quad 0 \quad 0 \quad 0 \\
1 & : 0 \quad 0 \quad 0 \quad 0 \\
\end{align*}
\]

5 code words

\[
\begin{align*}
P(0) &= \frac{n-1}{n} \quad P(1) = \frac{1}{n}
\end{align*}
\]

\[
k = \frac{n}{2}
\]

\[
P(0) = P(1) = \frac{1}{2}
\]

* for $1 < k < \frac{n}{2}$, we are sure that $P(0) > P(1)$

> the transition-based encoding benefits.

Conclusion:

\[
\begin{align*}
\text{transfer to} \\
k - \text{limited} \\
\text{weight code} \\
\end{align*}
\]

○ $P(0) < P(1)$

\[
\begin{align*}
\text{transmit by} \\
\text{transition - based} \\
\text{encoding.}
\end{align*}
\]
Bus inventing encoding

- N bit lines
- Represent $2^n$ data
- $P(d) = P(1) = \frac{1}{2}$ for each line

6° Average # of transitions
\[
\frac{\text{clock cycle}}{\text{clock cycle}} = \frac{n}{2}.
\]

Worst # of transitions
\[
\frac{\text{clock cycle}}{\text{clock cycle}} = n.
\]

* Try to reduce both average (worst) # of transitions per cycle.
* Compare each pair of consecutive patterns.

1. If Hamming distance between current & next patterns
   \[ \leq \frac{n}{2}, \quad I = 0, \text{ send as what it is.} \]

2. If Hamming distance between
   \[ > \frac{n}{2}, \quad I = 1, \text{ transmit the data.} \]

   \[ \text{pattern is converted} \]

\[ \text{This guarantees that the max \# of transition per} \]
\[ \text{clock cycle is } \frac{n}{2}. \]

\[ \text{By statistical analysis, the average \# of transitions} \]
\[ \text{per clock cycle is decreased by } 25\% \text{ of the original.} \]

\[ \text{Hardware overhead for code generation must be invested.} \]
\[ \text{All above discussions are for data bus.} \]

\[ \text{How about address bus?} \]

\[ \text{Addresses generated by a running rep are often} \]
\[ \text{consecutive.} \]

\[ \text{Try to represent address by } G \text{ray code.} \]
\[ \text{(guarantee single-bit change for current \& next address)} \]
- Computer simulation demonstrate about 87% power reduction for several benchmark programs.

- **Signal Synchronization**
  
  - Try to reduce the \# of glitches in RT-level, by balancing signal paths or reducing logic depth.

**Example:**

A → adder 1 → adder 2 → adder 3

\[ \text{Chain crit.} \]

\[ \text{Glitch 1} \quad \text{Glitch 2} \]

\[ \begin{align*}
\text{transition by} & \quad \text{old} (\text{A}+\text{B}) + \text{C} \\
\text{transition by} & \quad \text{new} (\text{A}+\text{B}) + \text{C} \\
\text{transition by} & \quad \text{glitch} (\text{A}+\text{B}) + \text{D}
\end{align*} \]
Low-power design

Paths are balanced, so the # of glitches is minimized.

8 inputs

Decrease the logic depth increases the # of registers required by the design. → add extra power consumption!!
Resource sharing: readily assignment.

Algorithmic-level optimization

- Try to modify the computational structure of the algorithm while preserving I/O behavior.
- Objective: optimize the power dissipation with a specific throughput.

Speed-up transformations

\[ Y_N = A \cdot Y_{N-1} + X_N \]

Infinite impulse response

First-order IIR filter

\[ \text{Ceff} = 1 \]
\[ \text{Vdd} = 5 \text{V} \]
\[ \text{Throughput} = 1 \]
\[ \text{Critical path} = 2 \]
\[ C_0^2 \rightarrow \text{power} = 25 \]
- Can't be further optimized by pipelining, re-timing.

- Try to unfold the loop

\[ Y_{N+1} = Y_{N+2} \cdot A + X_{N+1} \]

\[ Y_N = Y_{N+1} \cdot A + X_N \]

\[ = \left( Y_{N+2} \cdot A + X_{N+1} \right) A + X_N \]

\[ = Y_{N+2} A^2 + X_{N+1} A + X_N \]

So function is the same.
\[ C_{\text{eff}} = 1 \quad (C \uparrow \text{ but } E(52.5) \downarrow) \]

\[ \text{Vdd} = 5\, \text{V} \]

\[ \text{throughput} = 1 \]

\[ \text{Critical path} = \bigcirc \oplus 4, \text{ but effective critical path} = 2. \]

\[ \text{Power} = 25 \]

\[ 37.5\, \text{ns} \]
\[ Y_{n-1} = Y_{n-2} \cdot A + X_{n-1} \]

\[ Y_n = (X_{n-1} \cdot A + X_n) + A^2 \cdot Y_{n-2} \]

\[ = A^2 Y_{n-2} + A X_{n-1} + X_n \]

* Critical path has been reduced from 4 to 3.

→ Supply voltage can be reduced to keep the same throughput

\[ C_{eff} = 1.5 \] (added more multipliers & adders)

Also, more devices are shared simultaneously.

\[ V_{dd} = 3.7V \]

Throughput = 1

Critical path = 3

Power = 20

→ 20% power saving over the original design.

* Further improvement by pipelining
\[ \begin{align*}
Y_0 &= x_0 + A \cdot y_1 \\
Y_1 &= x_1 + A \cdot y_0 = x_1 + A(x_0 + A \cdot y_1) \\
Y_2 &= x_2 + A \cdot y_1 = x_1 + A \cdot x_0 + A^2 \cdot y_1 \\
Y_3 &= x_3 + A \cdot y_2 = x_3 + A(x_2 + A \cdot y_1) \\
&= x_3 + A x_2 + A^2 y_1
\end{align*} \]
\[ X_N \rightarrow C \rightarrow [D-2] \rightarrow \oplus \rightarrow Y_N \]

\[ A \rightarrow \otimes \rightarrow [D-1] \rightarrow \oplus \rightarrow Y_{N-1} \]

\[ X_{N-1} \rightarrow [D-1] \rightarrow \oplus \rightarrow Y_{N-1} \]

\[ X_{N-1} \quad X_N \quad D-1 \quad D-2 \quad 2D \quad Y_{N-1} \quad Y_N \]

\[ x_0 \quad x_1 \quad x \quad x \quad x \quad x \]

(by reset)

\[ x_0 \quad A \cdot x_0 + x_1 \quad Y_{-1} \quad Y_0 \quad Y_1 \]

\[ x_2 \quad x_3 \]

\[ X_2 \quad AX_2 + x_3 \quad Y_1 \quad Y_2 = AY_1 + x_2 \]

\[ Y_3 = X_3 + AX_2 + A^2 Y_1 \]
To get the same throughput, we still use 2t
Clock cycle (50 ns)

\[ Y_n \text{ and } Y_{n+1} \text{ can be generated simultaneously.} \]

\[ \text{New } X^n \]
\[ \text{Old } X^n \Rightarrow D \]
\[ \text{Old } X_{n-1} \Rightarrow D \]

\[ Ceff = 1.5 \]
\[ Vdd = 2.9V \]
\[ \text{delay is changed from } 50 \text{ ns} \rightarrow 25 \text{ ns} \]

Throughput = 1

Critical path = 2

\[ C_{D^2} \]
\[ \text{power} = 12.5 \]

50% power reduction
power reduction for unrolling factor 2.3

power increase from factor = 4