The Effects of Energy Management on Reliability in Real-Time Embedded Systems

Authors: Dakai Zhu, Rami Melhem, and Daniel Mossé

Presentors: Sarah Chen, Jiayi Liu, Jielun Tan, Griffin Whybra, Boyang Zhang
Fault Tolerance and Power Efficiency

Using slack time

Transient faults depend on frequency and voltage
Soft Errors - Transient Faults

Alpha particles from radioactive decay of a package

Cosmic rays from the atmosphere

Not a hard fault
Soft Error Masking

Logic

Timing

Electrical

Microarch

Software

Clock

Masked Fault Latched Fault

$\text{t}_{\text{setup}} + \text{t}_{\text{hold}}$

Attenuated Pulse

Latch
Measuring Faults

FIT: Failures in $10^9$ hours.

$114\text{FIT} = 114$ Failures in $10^9$ hours. Or once every thousand years

@ 10k parts failure once a month

Mean time between failure = $1 / \text{FIT}$
Modeling

L: Worst case execution time assuming application executing on $f_{\text{max}}$.

D: Deadline for the application to finish execution.

$f_{\text{max}}$ is normalized frequency and $f_{\text{max}} = 1$.

Assumption: The execution time of an application is linearly related to the frequency.
Power Modeling

\[ P = P_s + h(P_{\text{ind}} + P_d) = P_s + h(P_{\text{ind}} + C_{\text{ef}} V^2 f) \]

\( P_s \) is the sleep power

\( P_{\text{ind}} \) is the frequency independent active power.

\( h \) determines whether the system is in sleep mode.

\( C_{\text{ef}} \) is the switch capacitance, \( V \) is the supply voltage and \( f \) is the frequency.

The sleep power \( P_s \) is always consumed and is not manageable because the researchers assumed the system is always on.
Frequency Scaling

\[ E = (P_{\text{ind}} + C_{\text{ef}} V_{\text{max}}^2 f) \frac{L}{f} = P_{\text{ind}} \frac{L}{f} + C_{\text{ef}} V_{\text{max}}^2 L \]

If \( P_{\text{ind}} > 0 \), \( \frac{dE}{df} < 0 \). Thus, more energy will be consumed by reducing frequency.
Voltage Scaling

Like frequency, normalized voltage is being used, \( V_{\text{max}} = 1 \).

\[ V = f \cdot V_{\text{max}} = f \]. This implies that:

\[
E = (P_{\text{ind}} + C_{ef} V^2 f) \frac{L}{f} = P_{\text{ind}} \frac{L}{f} + C_{ef} f^2 L
\]

By reducing frequency, less frequency dependent energy is consumed (2nd term) while more frequency independent energy will be consumed (1st term).
Voltage Scaling

$F_{ee}$ the frequency to minimize energy consumption which is given by:

$$f_{\text{min}} = \sqrt[3]{\frac{P_{\text{ind}}}{2C_{ef}}}$$

$f_{\text{low}}$ the minimum frequency for the application to finish the application before the deadline.

$f_{\text{min}}$ the minimum energy consumption frequency for the application to finish the before the deadline.
Voltage Scaling

\[ f_{\text{min}} = \max\{f_{\text{ee}}, f_{\text{low}}\} \]
Fault Model

Transient faults can be caused by high-energy particles striking the device

Resulting in:

- Logic errors
- *Critical charge* for memory cell exceeded, causing memory bit flip
Fault Model

Faults are assumed to follow a Poisson distribution.

For simplicity, only supply voltage and frequency affect the average fault rate.

General Model

\[ \lambda(f, V) = \lambda_0 \cdot g(f, V) \]

where \( \lambda_0 \) is the average fault rate wrt \( V_{max} \) and \( f_{max} \).
Fault Model

Frequency scaling - Linear model (when $b = 1$)

Fixed voltage

Fault rate *decreases linearly* when frequency is reduced

Results from larger safety margins in clock cycles when frequency reduced

\[
\lambda(f, V) = \lambda(f) = \lambda_0 \cdot f^b
\]
Fault Model

Voltage scaling - Exponential model

Fault rate *increases exponentially* when voltage is reduced for lower frequencies

Results from a smaller critical charge

\[
\lambda(f, V) = \lambda(f) = \lambda_0 10^{\frac{d(1-f)}{1-f_{min}}}
\]
Recovery Model

Detect transient fault → re-execute application

Lower operation frequency → longer execution time
- Advantage: fault rate reduced
- Disadvantage: fewer recoveries
- Energy - help or hurt efficiency?

Missing from paper: When/how to detect faults
How much is re-executed

Average fault rate: \( \lambda(f,V) = \lambda_0 \cdot f^b \)

D = deadline
L = worse-case execution time
Analysis Results

- Analysis performed on Intel Pentium M Processor
  - $P_{\text{ind}} = \{0, 0.2, 0.4\}$
- 1,000 to 100,000 FITs per Mb of memory, 10e-6 faults/second on each chip (assuming 100 Mb chip) and 10e-6 to 10e-2 faults/second in a system
  - Highly unrealistic, FIT in a system is not the accumulation of faults in individual components due to masking
  - Almost no terrestrial system is getting a fault every 100 second
- If fault scales more than linearly with frequency, then frequency scaling results in higher performability
- When it’s possible to efficiently remove $P_{\text{ind}}$ by entering sleep state, scaling down only frequency only causes increase in energy consumption
  - If the system is put to sleep, why would it be executing?
- Voltage scaling reduces energy consumption but also reliability as well, and performability if faults scale non-linearly
Analysis Results

● The analytical results did not factor in the time for error detection
  ○ Moreover, how to detect error in the first place
    ■ If done via Razor-like techniques, then it doesn’t take much time
    ■ However if it’s done after program execution, then it can a lot of time to verify the results
  ○ If the program throws an exception because of soft error, then it would not take L time to complete, the execution time of the program is also a random variable should a fault occur

● Also assumed very coarse-grain recovery by re-executing the program
  ○ In the typical DMR or TMR systems, recovery can be done at the pipeline level
Questions?
Recovery Model Equations

\[
# \text{ re-executions} = k_f = \left\lfloor \frac{D_f}{L} \right\rfloor - 1 \quad (8)
\]

\[
\text{probability of at least 1 fault} = \rho_f = 1 - e^{-\lambda(f,V) \frac{L}{f}} \quad (9)
\]

Thus, when there are \( k_f \) recoveries, the performability is:

\[
R_f = 1 - \rho_f^{k_f+1} = 1 - \left( 1 - e^{-\lambda(f,V) \frac{L}{f}} \right)^{\left\lfloor \frac{D_f}{L} \right\rfloor} \quad (10)
\]

\[
\frac{k_f+1}{\rho_f} = \text{probability of faulting every execution}
\]

Expected energy consumption =

\[
EE_f = E_f \sum_{i=0}^{k_f} \rho_f^i = E_f \frac{1 - \rho_f^{k_f+1}}{1 - \rho_f} \quad (11)
\]
Split up topics

Griffin: Intro and background
Jiayi: Power modeling
Bo: Fault mode (Section II, part C)
Sarah: Recovery model
Tan: Results and critique