Voltage Overscaling Algorithms for Energy-Efficient Workflow Computations With Timing Errors

Aurélien Cavelan[1], Yves Robert[1,2], Hongyang Sun[1] and Frédéric Vivien[1]

1. ENS Lyon & INRIA, France
2. University of Tennessee Knoxville, USA

aurelien.cavelan@ens-lyon.fr

Green Days – Toulouse
March 17, 2015

Outline

Dynamic Power Consumption

One can use *Dynamic Voltage and Frequency Scaling (DVFS)* to reduce power consumption.

Dynamic Power Consumption

One can use *Dynamic Voltage and Frequency Scaling (DVFS)* to reduce power consumption.

$$Power = \alpha f V^2$$

- α the effective capacitance
- f the frequency
- V the operating voltage

Dynamic Power Consumption

One can use *Dynamic Voltage and Frequency Scaling (DVFS)* to reduce power consumption.

$$Power = \alpha f V^2$$

- α the effective capacitance
- f the frequency
- V the operating voltage

$\Rightarrow$Voltage has a quadratic impact on the dynamic power.

The Easy Way

We target energy consumption only, not time.

For any frequency value, there is a threshold voltage:

The Easy Way

We target energy consumption only, not time.

For any frequency value, there is a threshold voltage:

1. Find the frequency that minimizes energy consumption

The Easy Way

We target energy consumption only, not time.

For any frequency value, there is a threshold voltage:

1. Find the frequency that minimizes energy consumption
2. Decrease the voltage to *threshold voltage*

The Easy Way

We target energy consumption only, not time.

For any frequency value, there is a threshold voltage:

1. Find the frequency that minimizes energy consumption
2. Decrease the voltage to *threshold voltage*

?

Can we do better ?

Threshold Voltage

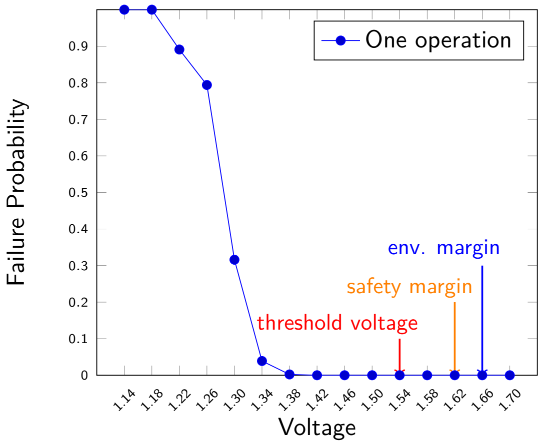

Figure: Set of voltages of a FPGA multiplier block and the associated error probabilities measured on random inputs at 90MHz and 27°C

Timing Errors

Definition

The results of some logic gates could be used before their output signals reach their final values.

- Occur when $V_{\mathrm{DD}} < V_{\mathrm{TH}}$
- *Deterministic* but *unpredictable*
- Induce Silent Data Corruptions (SDC)

Timing Errors

Definition

The results of some logic gates could be used before their output signals reach their final values.

- Occur when $V_{\mathrm{DD}} < V_{\mathrm{TH}}$
- *Deterministic* but *unpredictable*
- Induce Silent Data Corruptions (SDC)

Unlike lightning, timing errors always strike twice

Timing Errors

Definition

The results of some logic gates could be used before their output signals reach their final values.

- Occur when $V_{\text{DD}} < V_{\text{TH}}$
- *Deterministic* but *unpredictable*
- Induce Silent Data Corruptions (SDC)

Unlike lightning, timing errors always strike twice

Silent errors are detected only when the corrupt data is activated

Two Approaches

Near-Threshold Computing ($V_{\mathbf{dd}} \approx V_{\mathbf{th}}$)

- Used in NTC circuits (hardware)
- Almost *safe* ☺
- Great energy savings ☺

Two Approaches

Near-Threshold Computing ($V_{dd} \approx V_{th}$)

- Used in NTC circuits (hardware)
- Almost *safe* ☺
- Great energy savings ☺

Voltage Overscaling ($V_{dd} < V_{th}$)

- Even more energy savings ☺
- Purely software-based approach ☺
- Generate timing errors ☹
- Require a verification mechanism ☹
- Require probabilities of failure for the platform ☹

Question

?

Is it possible to obtain the (correct) result of a computation for a lower energy budget than that of the best DVFS / NTC solution?

Outline

Model Assumptions

Consider a task and a set of voltages $\mathcal{V}$:

Voltages	V_1	V_2	$\cdots$	$V_m = V_{\text{TH}}$
$\mathbb{P}(V_\ell\text{-fail})$	p_1	p_2	$\ldots$	$p_m = 0$
Cost	c_1	c_2	$\ldots$	c_m

Model Assumptions

Consider a task and a set of voltages $\mathcal{V}$:

Voltages	V_1	V_2	$\cdots$	$V_m = V_{\text{TH}}$
$\mathbb{P}(V_\ell\text{-fail})$	p_1	p_2	$\ldots$	$p_m = 0$
Cost	c_1	c_2	$\ldots$	c_m

Remember: timing errors always strike twice.

Model Assumptions

Consider a task and a set of voltages $\mathcal{V}$:

Voltages	V_1	V_2	$\cdots$	$V_m = V_{\text{TH}}$
$\mathbb{P}(V_\ell\text{-fail})$	p_1	p_2	$\ldots$	$p_m = 0$
Cost	c_1	c_2	$\ldots$	c_m

Remember: timing errors always strike twice.

- When an error strikes, a higher voltage *must* be used
- Switching from voltage V_ℓ to V_h incurs a cost $o_{\ell,h}$
- Execution at V_{TH} always succeeds

Model Assumptions

Consider a task and a set of voltages $\mathcal{V}$:

Voltages	V_1	V_2	$\cdots$	$V_m = V_{\text{TH}}$
$\mathbb{P}(V_\ell\text{-fail})$	p_1	p_2	$\ldots$	$p_m = 0$
Cost	c_1	c_2	$\ldots$	c_m

Remember: timing errors always strike twice.

- When an error strikes, a higher voltage *must* be used
- Switching from voltage V_ℓ to V_h incurs a cost $o_{\ell,h}$
- Execution at V_{TH} always succeeds

How to compute the probability of failure ?

Model Assumptions

Consider a task and a set of voltages $\mathcal{V}$:

Voltages	V_1	V_2	$\cdots$	$V_m = V_{\text{TH}}$
$\mathbb{P}(V_\ell\text{-fail})$	p_1	p_2	$\dots$	$p_m = 0$
Cost	c_1	c_2	$\dots$	c_m

Remember: timing errors always strike twice.

- When an error strikes, a higher voltage *must* be used
- Switching from voltage V_ℓ to V_h incurs a cost $o_{\ell,h}$
- Execution at V_{TH} always succeeds

How to compute the probability of failure ?

The optimal sequence of voltages ?

Property of Timing Errors

1. Given an operation and an input I, there exists a *threshold voltage* $V_{\mathrm{TH}}(I)$:
   - $V < V_{\mathrm{TH}}(I)$ will *always* lead to an incorrect result
   - $V \geq V_{\mathrm{TH}}(I)$ will always lead to a successful execution

Property of Timing Errors

1. Given an operation and an input I, there exists a *threshold voltage* $V_{\mathrm{TH}}(I)$:
   - $V < V_{\mathrm{TH}}(I)$ will *always* lead to an incorrect result
   - $V \geq V_{\mathrm{TH}}(I)$ will always lead to a successful execution
2. Given an operation and a voltage $V \in \mathcal{V}$:
   - $\mathcal{I}$ denotes the set of all possible inputs
   - $\mathcal{I}_f(V) \subseteq \mathcal{I}$ is the set of inputs that will fail at voltage V
   - Failure probability is computed as $p_V = |\mathcal{I}_f(V)|/|\mathcal{I}|$
   - For any two voltages $V_1 \geq V_2$, we have $\mathcal{I}_f(V_1) \subseteq \mathcal{I}_f(V_2)$

Property of Timing Errors

1. Given an operation and an input I, there exists a *threshold voltage* $V_{\mathrm{TH}}(I)$:
   - $V < V_{\mathrm{TH}}(I)$ will *always* lead to an incorrect result
   - $V \geq V_{\mathrm{TH}}(I)$ will always lead to a successful execution

2. Given an operation and a voltage $V \in \mathcal{V}$:
   - $\mathcal{I}$ denotes the set of all possible inputs
   - $\mathcal{I}_f(V) \subseteq \mathcal{I}$ is the set of inputs that will fail at voltage V
   - Failure probability is computed as $p_V = |\mathcal{I}_f(V)|/|\mathcal{I}|$
   - For any two voltages $V_1 \geq V_2$, we have $\mathcal{I}_f(V_1) \subseteq \mathcal{I}_f(V_2)$

$$\mathbb{P}(V_\ell\text{-fail} \mid V_0 V_1 \cdots V_{\ell-1}\text{-fail}) = \frac{|\mathcal{I}_f(V_\ell)|/|\mathcal{I}|}{|\mathcal{I}_f(V_{\ell-1})|/|\mathcal{I}|} = \frac{p_\ell}{p_{\ell-1}}$$

Energy Consumption of a Single Task

Consider a sequence L of k voltages $V_1 < V_2 < \cdots < V_k = V_{\mathrm{TH}}$,

Execution starts at voltage V_1:

1. In case of success, return the result !
2. In case of failure, go to next (higher) voltage

Energy Consumption of a Single Task

Consider a sequence L of k voltages $V_1 < V_2 < \cdots < V_k = V_{\mathrm{TH}}$,

Execution starts at voltage V_1:

1. In case of success, return the result !
2. In case of failure, go to next (higher) voltage

$$\begin{aligned} E(L) &= c_1 + p_1 \left(o_{1,2} + c_2 + \frac{p_2}{p_1} \left(o_{2,3} + c_3 + \dots \frac{p_{k-1}}{p_{k-2}} (o_{k-1,k} + c_k) \right) \right) \\ &= c_1 + p_1(o_{1,2} + c_2) + p_2(o_{2,3} + c_3) + \cdots + p_{k-1}(o_{k-1,k} + c_k) \end{aligned}$$

Energy Consumption of a Single Task

Consider a sequence L of k voltages $V_1 < V_2 < \cdots < V_k = V_{\mathrm{TH}}$,

Execution starts at voltage V_1:

1. In case of success, return the result !
2. In case of failure, go to next (higher) voltage

$$\begin{aligned} E(L) &= c_1 + p_1 \left(o_{1,2} + c_2 + \frac{p_2}{p_1} \left(o_{2,3} + c_3 + \dots \frac{p_{k-1}}{p_{k-2}} (o_{k-1,k} + c_k) \right) \right) \\ &= c_1 + p_1(o_{1,2} + c_2) + p_2(o_{2,3} + c_3) + \cdots + p_{k-1}(o_{k-1,k} + c_k) \end{aligned}$$

We generalize:

$$E(L) = c_1 + \sum_{\ell=2}^{k} p_{\ell-1} \left(o_{\ell-1,\ell} + c_\ell \right) \quad (1)$$

Optimal Sequence of Voltages

Theorem

To minimize the expected energy consumption for a single task, the optimal sequence of voltages to execute the task with a preset voltage $V_p \in \mathcal{V}$ of the system can be obtained by dynamic programming with complexity $O(k^2)$.

Optimal Sequence of Voltages

Theorem

To minimize the expected energy consumption for a single task, the optimal sequence of voltages to execute the task with a preset voltage $V_p \in \mathcal{V}$ of the system can be obtained by dynamic programming with complexity $O(k^2)$.

$$E(L_s^*) = c_s + \min_{s<\ell\leq k} \left\{ E(L_\ell^*) - c_\ell + p_s(o_{s,\ell} + c_\ell) \right\} \qquad (2)$$

and the optimal sequence starting with V_s is $L_s^* = \langle V_s, L_{\ell'}^* \rangle$ where

$$\ell' = \arg\min_{s<\ell\leq k} \left\{ E(L_\ell^*) + p_s o_{s,\ell} + (p_s - 1)c_\ell \right\}.$$

The dynamic program is initialized with $E(L_k^*) = c_k$ and $L_k^* = \langle V_k \rangle$

Chain of Tasks

- Without switching cost: optimal sequence for one task can be used to execute each task.

Chain of Tasks

- Without switching cost: optimal sequence for one task can be used to execute each task.
- With switching cost:
  - After execution of a task, platform is left at voltage V_e
  - Optimal sequence starts at voltage V_s
  - Additional switching cost $o_{e,s}$ must be paid
  - Algorithm for one task is no longer optimal

Chain of Tasks

- Without switching cost: optimal sequence for one task can be used to execute each task.
- With switching cost:
  - After execution of a task, platform is left at voltage V_e
  - Optimal sequence starts at voltage V_s
  - Additional switching cost $o_{e,s}$ must be paid
  - Algorithm for one task is no longer optimal

Theorem

To minimize the expected energy consumption for a linear chain of tasks, the optimal sequence of voltages to execute each task, given the terminating voltage of its preceding task (or given the preset voltage V_p of the system for the first task), can be obtained by dynamic programming with complexity $O(nk^2)$.

Outline

Blocked Matrix-Matrix Multiplication

Consider the blocked matrix multiplication $C = A \times B$.

Application Workflow

for $i = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
  for $j = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
    for $k = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
      $C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$

- m denotes the matrix size
- b denotes the block size

Blocked Matrix-Matrix Multiplication

Consider the blocked matrix multiplication $C = A \times B$.

Application Workflow

for $i = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
  for $j = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
    for $k = 1$ to $\lceil \frac{m}{b} \rceil$ **do**
      $C_{i,j} \leftarrow C_{i,j} + A_{i,k} \times B_{k,j}$

- m denotes the matrix size
- b denotes the block size

ABFT can be used to add per-block verification.

Algorithm Based Fault Tolerence *(ABFT)*

Let $e^T = [1, 1, \cdots, 1]$, we define

$$A^c := \begin{pmatrix} A \\ e^T A \end{pmatrix},\ B^r := \begin{pmatrix} B & Be \end{pmatrix},\ C^f := \begin{pmatrix} C & Ce \\ e^T C & e^T Ce \end{pmatrix}.$$

Where A^c is the *column checksum matrix*, B^r is the *row checksum matrix* and C^f is the *full checksum matrix*.

Algorithm Based Fault Tolerence *(ABFT)*

Let $e^T = [1, 1, \cdots, 1]$, we define

$$A^c := \begin{pmatrix} A \\ e^T A \end{pmatrix},\ B^r := \begin{pmatrix} B & Be \end{pmatrix},\ C^f := \begin{pmatrix} C & Ce \\ e^T C & e^T Ce \end{pmatrix}.$$

Where A^c is the *column checksum matrix*, B^r is the *row checksum matrix* and C^f is the *full checksum matrix*.

$$\begin{aligned} A^c \times B^r &= \begin{pmatrix} A \\ e^T A \end{pmatrix} \times \begin{pmatrix} B & Be \end{pmatrix} \\ &= \begin{pmatrix} AB & ABe \\ e^T AB & e^T ABe \end{pmatrix} = \begin{pmatrix} C & Ce \\ e^T C & e^T Ce \end{pmatrix} = C^f \end{aligned}$$

Matrix Parameters

Consider the matrix multiplication as a chain of $n = \lceil \frac{m}{b} \rceil^3$ tasks.

Time to Execute one Task

- $t = \tau \cdot w/\eta$
  - $\tau = 1/f$ time to do one cycle
  - $\eta = 0.8$ peak performance
- $w = b(b+1)^2 + \sigma$
  - $\sigma = 8^3$ initialization overhead
  - $(b+1)$ ABFT overhead

Matrix Parameters

Consider the matrix multiplication as a chain of $n = \lceil \frac{m}{b} \rceil^3$ tasks.

Time to Execute one Task

- $t = \tau \cdot w / \eta$
  - $\tau = 1/f$ time to do one cycle
  - $\eta = 0.8$ peak performance
- $w = b(b+1)^2 + \sigma$
  - $\sigma = 8^3$ initialization overhead
  - $(b+1)$ ABFT overhead

Failure Probabilities

Consider a set of voltages $\mathcal{V}$. For any voltage $V_\ell \in \mathcal{V}$

- $p_\ell = 1 - (1 - p_\ell^{(1)}/\gamma)^w$
  - $\gamma = \frac{\text{silent errors}}{\text{timing errors}}$
  - $p_\ell^{(1)}$ probablity of timing error for one random operation

Platform Settings

From [1] for a FPGA at $f = 90$MHz and 27°C:

- Set of voltages
- Timing errors probabilities

Dynamic Power Consumption

- $P(V, f) = \alpha f V^2$
- We assume (wlog) $\alpha f = 1$

Platform Settings

From [1] for a FPGA at $f = 90$MHz and 27°C:

- Set of voltages
- Timing errors probabilities

Dynamic Power Consumption

- $P(V, f) = \alpha f V^2$
- We assume (wlog) $\alpha f = 1$

Platform Settings

From [1] for a FPGA at $f = 90$MHz and 27°C:

- Set of voltages
- Timing errors probabilities

Dynamic Power Consumption

- $P(V, f) = \alpha f V^2$
- We assume (wlog) $\alpha f = 1$

Voltage Switching Cost

- $o_{\ell,h} = \begin{cases} 0, & \text{if } \ell = h \\ \beta \cdot \frac{|V_\ell - V_h|}{V_k - V_1} & \text{otherwise} \end{cases}$
- $\beta = o_{1,k}$

Algorithms

- *N-Voltage*: Baseline algorithm that applies NTC and always uses threshold voltage.
- *DP_1-detect* & *DP_1-correct*: Optimal dynamic programming algorithms for a single task.
- *DP_n-detect* & *DP_n-correct*: Optimal dynamic programming algorithms for a for a chain of tasks.

detect algorithms use ABFT for error detection.
correct algorithms use ABFT for detection and correction.

Probabilities of failure

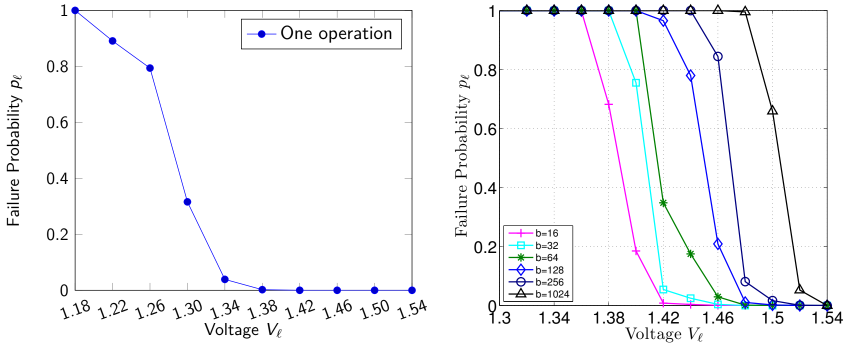

Figure: Failure probabilities for one operation and for one task under different block sizes and voltages.

Simulations (without switching cost)

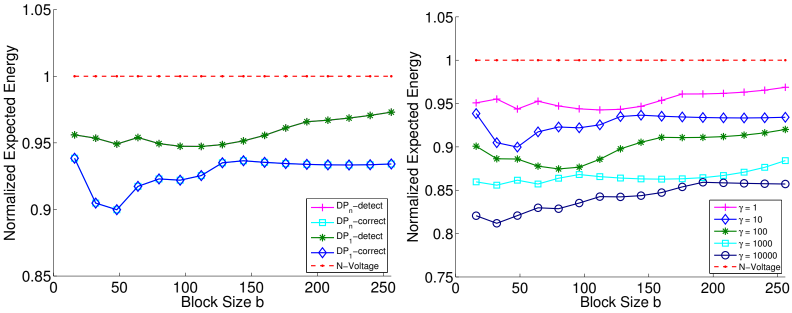

Figure: Impact of b and γ on the expected energy consumption for zero voltage switching cost. Only the results for the DP$_n$-correct algorithm are shown.

Simulations (with switching cost)

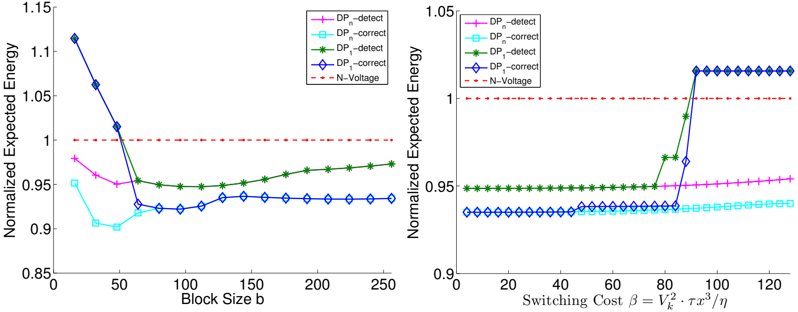

Figure: Impact of b and β on the expected energy consumption. The voltage switching cost is equivalent to the energy consumed to multiply two 32×32 matrices at threshold voltage without overhead.

Outline

Conclusion

We use dynamic voltage overscaling to reduce power consumption.

Summary

- Software based approach
- Model for timing errors
- Optimal polynomial-time solution for a chain of tasks
- Simulations on matrix multiplication using ABFT

Conclusion

We use dynamic voltage overscaling to reduce power consumption.

Summary

- Software based approach
- Model for timing errors
- Optimal polynomial-time solution for a chain of tasks
- Simulations on matrix multiplication using ABFT

Original problem and encouraging results; needs further research.

Conclusion

We use dynamic voltage overscaling to reduce power consumption.

Summary

- Software based approach
- Model for timing errors
- Optimal polynomial-time solution for a chain of tasks
- Simulations on matrix multiplication using ABFT

Original problem and encouraging results; needs further research.

Future Work

- Algorithms for other task graphs
- Additional simulations, emulations and experiments

Questions

Thanks! ☺

Questions?

Bibliography

D. Ernst, S. Das, S. Lee, D. Blaauw, T. Austin, T. Mudge, N. S. Kim, and K. Flautner.
Razor: circuit-level correction of timing errors for low-power operation.
IEEE Micro, 24(6):10–20, 2004.