Variability-tolerant High-reliability Multicore Platforms

DEIS – Micrel Lab – University of Bologna

Francesco Paterna  PhD Student  francesco.paterna@unibo.it

Prof. Luca Benini  Supervisor  luca.benini@unibo.it
Multi processor System-on-Chip

• Applications
  – Multimedia streaming

• Requirements
  – Quality of Service (*real-time constraints*)
  – Energy efficiency (> 10GOPS/Watt)
Variability:
- Independently affects dynamic power, leakage power, and longest path delay of cores
- Die to die effects: differences among platforms
- Within die effects: differences among cores

Reliability:
- Aging and wear-out phenomena as NTBI, HCI, Thermal Cycle, Electromigration, TDDB slow down CPU performance with CPU usage. The amount strongly depends on operating conditions (Vdd, T and Vt) history.
- $f_{\text{MAX}}, V_t$ deteriorate differently for different cores due to aging.
Research activity

- adaptive software techniques for performance improvements, power consumption reduction, and lifetime requirements in MPSoC on sub-45nm CMOS technology

- Using of MPSoC Virtual Platform (VP)
- High Level Modeling of power consumption, aging, thermal, and variability effects. Add-ons implementation for VPs
- Policy development to compensate undesired effects and to improve performances
- Development and using of multi-core multi-thread real-time applications
- Experiments
Virtual Platform

• Industrial multicore platform xSTream provided by STMicroelectronics:
  one ST231 host core, and a regular array of xPE processors
  each one with his own distributed but uniform memory address space

Collaboration with
ST Microelectronics Cornaredo (Milano).

Internship 3 months
Monitor core device

- 11 unsigned int 32-bit read register
- 2 unsigned int 32-bit w/r register
- 1 pin to refresh registers
- 1 pin to update to new aging state
- 1 pin to update to new thermal state

Features:
- Core activity monitoring,
- Power modeling,
- Aging modeling,
- Variability modeling
- Thermal modeling
High level power modeling

- Proportional constant for CPU Kd, Z, Kds, Kss, kdi, Ksi, kdpg, Kspg

\[
\begin{align*}
\text{P}_{\text{dyn-Active}} &= K_d \cdot V_{dd}^2 \cdot f \\
\text{P}_{\text{sta-Active}} &= Z \cdot V_{dd} \cdot T^2 \cdot e^{-q \cdot V_t} \\
\text{P}_{\text{dyn-Stall}} &= K_{ds} \cdot \text{P}_{\text{dyn-Active}} \\
\text{P}_{\text{dyn-Idle}} &= K_{di} \cdot \text{P}_{\text{dyn-Active}} \\
\text{P}_{\text{dyn-PowerGating}} &= K_{dpg} \cdot \text{P}_{\text{dyn-Active}} \\
\text{P}_{\text{sta-Stall}} &= K_{ss} \cdot \text{P}_{\text{sta-Active}} \\
\text{P}_{\text{sta-Idle}} &= K_{si} \cdot \text{P}_{\text{sta-Active}} \\
\text{P}_{\text{sta-PowerGating}} &= K_{spg} \cdot \text{P}_{\text{sta-Active}}
\end{align*}
\]

Checking of max clock frequency supported:

\[
T_g = \frac{1}{f} = \frac{L_f \cdot V_{dd}}{(V_{dd} - V_t)^\alpha}
\]

We are now using parameter values for 32nm provided by ST Microelectronics
Aging Model - NBTI

- Aging and critical path delay:
  - Facelift: Hiding and Slowing Down Aging in Multicores. A. Tiwari, J. Torrellas

\[ \Delta V_{t\text{stress}} = A_{\text{NBTI}} \cdot t_{oX} \cdot \sqrt{(C_{oX} \cdot (V_{dd} - V_t))} \cdot e^{\frac{V_{dd} - V_t - E_a}{t_{oX} \cdot E_0 \cdot K \cdot T \cdot t_{stress}^{0.25}}} \]

\[ \Delta V_{t\text{recovery}} = \Delta V_{t\text{stress}} \cdot \left(1 - \sqrt{\frac{\eta \cdot t_{\text{recovery}}}{t_{\text{recovery}} + t_{\text{stress}}}}\right) \]
Aging Model - NBTI

- Aging and critical path delay:
  - Facelift: Hiding and Slowing Down Aging in Multicores. A. Tiwari, J. Torrellas

\[ \Delta V_{tstress} = A_{NBTI} \cdot t_{ox} \cdot \sqrt{(C_{ox} \cdot (V_{dd} - V_t))} \cdot e^{0.25 \cdot \text{stress}} \]

\[ \Delta V_{recovery} = \Delta V_{tstress} \cdot \left( 1 - \frac{\eta \cdot t_{recovery}}{t_{recovery} + t_{stress}} \right) \]
Thermal Model

- For any Macro block $i$ at instant $t$
  - Forward dependency:
    - Static dependency: $B_{f-i} \cdot P_{\text{aver}_i}(t)/\text{Area}_i$
    - Dynamic dependency: $A_{f-i} \cdot T_i(t-1)$
  - Mutual dependency on any other Macro block $j$:
    - Static dependency: $B_{m-j} \cdot P_{\text{aver}_j}(t)/\text{Area}_j$
    - Dynamic dependency: $A_{m-j} \cdot T_j(t-1)$

$$T_i(t) = A_{f-i} \cdot T_i(t-1) + B_{f-i} \frac{P_{\text{aver}_i}(t)}{\text{Area}_i} + \sum_{j \neq i}^{N} \left[ A_{m-j} \cdot T_j(t-1) + B_{m-j} \frac{P_{\text{aver}_j}(t)}{\text{Area}_j} \right]$$
Variability causes power and performance variations among nominally similar cores in MPSoC platforms.

IMEC provided to us a set of 3D points related to critical path delay ($T_g$), dynamic power ($P_{dyn}$), and leakage power ($P_{lkg}$) generated using VAM (variability aware modeling).
Variability Injection

- VAM extracts a set of values for longest path delay, dynamic power, and leakage power.
- ISS plug-in set clock frequency in according to longest path delay, and stores power values to be evaluated the energy consumption during a run.
System lifetime requirement imposes idleness for each core:

Equalization of $t_{max}$ of each core to target lifetime ($t_{imax} = t_{max}, \forall i$)

Idleness Distribution Policy

Variability counters → Aging model → $t_{i max}$

Idleness Percentage

<table>
<thead>
<tr>
<th>Core #</th>
<th>Idleness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.01</td>
</tr>
<tr>
<td>N</td>
<td>0.7</td>
</tr>
</tbody>
</table>
Software Implementation

**Data allocation**

<table>
<thead>
<tr>
<th>Core #</th>
<th>Idleness</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.5</td>
</tr>
<tr>
<td>2</td>
<td>0.3</td>
</tr>
<tr>
<td>N</td>
<td>0.7</td>
</tr>
</tbody>
</table>

**Task dispatching**
Impact on execution time

lower is better

Our policy:

I. extension of lifetime:

- 1 / 4, 1 / 6 -> 33%
- 3 / 4, 5 / 6 -> 42%

II. exploit at its best a variability affected platform

Variability-tolerant Workload Allocation
for energy minimization
under real-time constraints

**ILP**

\[
E_{tot} = \sum_{i=1}^{N_{\text{cores}}} \left( \frac{P_{Ai} - P_{Li}}{f_{cki}} \right) \sum_{j=1}^{M_{\text{tasks}}} (x_{i,j} \cdot C_{j}) + T \sum_{i=1}^{N_{\text{cores}}} (x_{N+1,i} \cdot P_{Li})
\]

- \(C_{j}\) is the amount of cycles for task \(j\)
- \(x_{i,j} = \begin{cases} 
1 & \text{if task } j \text{ is mapped onto core } i \\
0 & \text{if task } j \text{ is NOT mapped onto core } i 
\end{cases}\)
- \(T\) is the time constraint

**Legend:**

- \(P_{Ai}\): dynamic + leakage power in activity state for cores \(i\)
- \(P_{Li}\): leakage power in idle state of cores \(i\)

**LP**

\[
E_{tot} = \sum_{i=1}^{N_{\text{cores}}} \left[ \frac{P_{Ai}}{f_{cki}} \cdot C_{Ai} + \frac{P_{Li}}{f_{cki}} \cdot C_{Li} \right]
\]

- \(C_{Ai}\) is the number of activity cycles for core \(i\)
- \(C_{Li}\) is the number of idle cycles for core \(i\)

\(C_i\) is the number of cycles that core \(i\) must execute. (capacity)
The sum of all \(C_i\) must be equal to \(I\), the total cycles of application.

\[C1 + C2 = I = I1 + I2 + I3\]
LP+BP Task Allocation Policy

1° Step Linear Programming Formulation
- **Inputs**: cycles budget of the whole application, time constraint, platform configuration (cores frequencies and powers)
- **Outputs**: activity cycles budget for each processor \( P_i \)

2° Step Bin Packing Formulation
- **Inputs**: LP output, required cycles of each independent task \( T_j \)
- **Output**: Tasks mapping on cores
Miss Rate Comparison

Results using 4-cores platform
Task sets with std dev / average = 0.5

Energy Comparison

Circles: Some deadline are not met

8 tasks

Deadline Miss Rate

<table>
<thead>
<tr>
<th></th>
<th>tconstr1</th>
<th>tconstr2</th>
<th>tconstr3</th>
<th>tconstr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>0.13</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>LP+BP</td>
<td>0.75</td>
<td>0.25</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>RankFreq</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.38</td>
</tr>
<tr>
<td>RankPower</td>
<td>1.00</td>
<td>1.00</td>
<td>0.75</td>
<td>0.38</td>
</tr>
</tbody>
</table>

32 tasks

<table>
<thead>
<tr>
<th></th>
<th>tconstr1</th>
<th>tconstr2</th>
<th>tconstr3</th>
<th>tconstr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>LP+BP</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>RankFreq</td>
<td>0.88</td>
<td>0.38</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>RankPower</td>
<td>1.00</td>
<td>0.38</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
Energy Saving Comparison

Results using 8-cores platform

Platform Utilization Percentage

<table>
<thead>
<tr>
<th></th>
<th>tconstr1</th>
<th>tconstr2</th>
<th>tconstr3</th>
<th>tconstr4</th>
</tr>
</thead>
<tbody>
<tr>
<td>ILP</td>
<td>42%</td>
<td>47%</td>
<td>50%</td>
<td>63%</td>
</tr>
<tr>
<td>LP+BP</td>
<td>36%</td>
<td>42%</td>
<td>45%</td>
<td>58%</td>
</tr>
<tr>
<td>RankFreq</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td>RankPower</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
</tbody>
</table>

The 42% value means that the 58% of cores are never used

LP+BP energy loss related to the ILP:

\[
\text{LOSS} = \frac{E_{LP+BP} - E_{ILP}}{E_{ILP}}
\]

4 cores, 8 tasks, standard deviation / average = 0.5: \text{LOSS} = 4.43%
8 cores, 8 tasks, standard deviation / average = 0.5: \text{LOSS} = 8.90%

In relaxed conditions, our policies find allocations that save more energy than rank policies.
Our policies compensate variations by reducing time violations due to variability effects.

The execution time of ILP and LP+BP are very close to the deadline independently from the platform.

Rank policies lead to much more variable execution times.

Our policies provide always lower energy while matching time constraints.

We used:
- 23 different variability affected platform by VAM;
- A group of 8-task set: $N_1 = 8$, std_dev/av = 0.5;

Deadline is 0.04 seconds
Benefits of the Policy


We demonstrated:

- The effectiveness of the proposed policies on an industrial platform for a large set of different workloads and tightness levels of deadline constraints,

- Our policies are more robust in terms of real-time predictability while providing competitive energy saving with respect to state-of-the-art variability aware policies
MPEG2 decoder

- An implementation of an MPEG2 decoder has been provided to UniBO
- The original version:
  - was used on the ST231-MT processor, which had 2 or 4 threads
  - supported 2 or 4 threads, one of which having control + decoding of a quadrant (or half) of the frame
  - Each thread decodes a N-th of the frame slices and a half or a quadrant of the frame image

N = 2

N = 4
This version has been modified so that:

- There is 1 control thread + N decode threads
- The control thread will be mapped on the GPE and the decode threads will be mapped on the hardware threads of the xPEs
- Each decoding thread decodes an N-th of the frame slices and one vertical portion of the frame image
- N has been generalized to be whatever number greater than 0 (the upper limit is the number of slices and the number of horizontal macroblocks in each frame)
Model of execution

GPE

- Initialization, read file header
- Read frame header
- Save previous results (processing power available for implementing variability policies)

B1 → B2 → B3

xPE HT

- Decode slices
- Decode macroblocks (vertical portion of frame)

B1 → B2 → B3
A closed form expression to solve online LP (Prof Alberto Caprara)

- Rank cores with their energy per cycle
- 2 solutions, take that at minimum energy
fVAR Platform Results

- For 4 tasks they are too much large to meet the deadline for all frames, LP+BP and Rank Frequency have the minimum time violation.

- LP+BP consumes less energy in all cases
Also in this case the VA policies are located in the lower side of the plot.
LP+BP execution time

LP+BP can be applied online

<table>
<thead>
<tr>
<th>execution time [musec]</th>
<th>3slaves</th>
<th>6slaves</th>
<th>9slaves</th>
<th>12slaves</th>
<th>16slaves</th>
</tr>
</thead>
<tbody>
<tr>
<td>LP</td>
<td>202</td>
<td>402</td>
<td>617</td>
<td>851</td>
<td>1189</td>
</tr>
<tr>
<td>BP (8 tasks)</td>
<td>6</td>
<td>21</td>
<td>64</td>
<td>109</td>
<td>168</td>
</tr>
<tr>
<td>BP (16 tasks)</td>
<td>13</td>
<td>37</td>
<td>64</td>
<td>112</td>
<td>159</td>
</tr>
<tr>
<td>BP (32 tasks)</td>
<td>36</td>
<td>86</td>
<td>153</td>
<td>240</td>
<td>421</td>
</tr>
<tr>
<td>BP (128 tasks)</td>
<td>271</td>
<td>429</td>
<td>677</td>
<td>1092</td>
<td>1729</td>
</tr>
</tbody>
</table>

- execution on the ST231 processor of the xSTream platform running at 600 MHz
- For MPEG2 decoder we have maximum 12000 musec to find the allocation

Conclusions

• MPSoC VP: using and add-ons development
• High Level Modeling (Power, Variability, Reliability, Thermal)
• Multimedia streaming application development

GOALS:
• Variability-aware Task Allocation Policies
  – Lifetime requirements
  – Energy minimization under real-time constraints
Future Works

Research
- Thermal effects controlling
- Variability-aware techniques for Dependent Task Graphs

Technical Work
- Virtual Platform Development (P2012 ST Microelectronics Grenoble – internship in progress)
- Multimedia streaming application development
Thank you!

• any questions?