Throughput Analysis and Voltage-Frequency Island Partitioning for Streaming Applications under Process Variation

Davit Mirzoyan, Benny Akesson, Sander Stuijk, Kees Goossens
Background

- Real-time streaming applications implemented on an MPSoC

An application: a (cyclic) task graph

- Mapping and scheduling to satisfy real-time requirements
- Examples
  - Software defined radio, video decoding (encoding)
Variation in Manufacturing Process

- Variation in transistor parameters
  - Die-to-die and Within-die [1]
  - Variation in the maximum supported frequency of cores in an MPSoC

![Probability Density Distribution](image)

Figure 2. Full-wafer ELM CD measurements.

Conventional Worst-Case Design

- Design margins or guard-bands (GB)
  - To guarantee the target frequency ($f_{tg}$) of cores in manufactured chips

- Deterministic core frequencies
  - Application mapping, such that a given timing requirement (e.g. \textit{throughput}, latency) is satisfied
Better than Worst-Case Design

- Reduced design margins
  - Smaller circuit area and thus more gross dies on a wafer
  - Target frequency of cores is not guaranteed any more

- We aim at maximizing the number of good dies
  - Dies that satisfy the throughput requirement of an application
  - Accomplished by exploiting frequency variation in cores in application mapping [2] and VFI partitioning steps.

Contributions (Work Overview)

- Framework to compute the probability distribution of application throughput in a system with VFIs under variation
- Heuristic VFI partitioning algorithm for maximized timing yield
  - Percentage of chips satisfying the throughput requirement
- Good dies = timing yield x gross dies
Outline

*Modeling*

VFI partitioning

Experimental results
**Hardware Platform**

- Heterogeneous multi-processor platform
  - Multiple processing elements (PE) connected to each other by a network on chip
- Globally asynchronous and locally synchronous architecture
  - With voltage-frequency islands
- A clock-generation unit (CGU) is assigned to each VFI
  - Each CGU provides discrete frequency levels

![Diagram of hardware platform with processing elements (PE) and clock-generation units (CGU) connected by a network on chip (NoC).]
Variation in PE Frequency

- Frequency of a PE is described by normal distributions
- Two variation types
  - Die-to-die (global)
  - Within-die (local)
- Clock-frequency levels for VFIs are selected based on combined distributions
Clock-Frequency Characterization

- A set of clock-frequency levels for each VFI
- All combinations of clock-frequency levels for the chip
- Characterizing a combination with a probability
  - Probability that VFIs are operated at specified clock-frequency levels

\[
\begin{array}{c|c|c|c}
\text{NoC} & \text{VFI} & \text{Probability} \\
\hline
\text{pe} & \{vfi_1, vfi_2, vfi_3\} & \{x_1, y_1, z_1\} & p_1 \\
\text{pe} & \{x_1, y_1, z_2\} & p_2 \\
\text{pe} & \{x_1, y_2, z_1\} & p_3 \\
\text{pe} & \{x_1, y_2, z_2\} & p_4 \\
\text{pe} & \{z_1, z_2\} & \cdot \\
\end{array}
\]
Unbound graph

- Application modelled as a *synchronous dataflow graph* (SDFG)
  - Binding unaware
  - Execution times of actors are given in clock cycles of PEs
- This graph is *decoupled* from hardware variation
Bound graph

- SDFG model of an application mapped to PEs
  - For a given binding
    - Resource sharing: Static Order Schedules on PEs
  - Execution time of application actors (in seconds)
- *Not decoupled* from hardware variation

\[ et(a, r) = \frac{ec(a, r)}{f_{pe}} \]

![Diagonal DAG with nodes and edges labeled with numbers]
• Characterizing each combination with a *throughput* value

\[ et(a, r) = \frac{ec(a, r)}{f_{pe}} \]

![Diagram with NoC and CDF](image)

<table>
<thead>
<tr>
<th>NoC</th>
<th>(cgu_1) (x_1\ x_2)</th>
<th>(cgu_2) (y_1\ y_2)</th>
<th>(cgu_3) (z_1\ z_2)</th>
<th>(vfi_1\ vfi_2\ vfi_3)</th>
<th>Probability</th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(x_1\ y_1\ z_1)</td>
<td>(x_1\ y_1\ z_2)</td>
<td>(x_1\ y_2\ z_2)</td>
<td>(x_1\ y_1\ z_1)</td>
<td>(p_1\ p_2\ p_3\ p_4)</td>
<td>(t_1\ t_2\ t_3\ t_4)</td>
</tr>
</tbody>
</table>
Modeling

**VFI partitioning**

Experimental results
Area Cost & Timing Yield Trade-Offs

- Reducing the number of VFI partitions
  - Less clock-generation units and thus smaller circuit area
  - More dies on a wafer
  - May reduce the timing yield

- Finding a beneficial trade-off between timing yield and the number of VFI partitions, such that more good dies are obtained

- *Partition PEs into VFIs, such that the highest timing yield is obtained*
Island criticality-based partitioning

- **Island criticality** metric guides the partitioning process
  - Quantifies the sensitivity of application’s throughput to the frequency of a VFI
- Merge a pair of VFIs with low criticality values in iterations
  - Islands may have equal (close) criticality values

1. Sort VFIs in increasing criticality
2. Evaluate the timing yield for all adjacent pairs
3. Merge the islands providing the lowest timing yield
4. Update VFI list and criticalities
Modeling
VFI partitioning
Experimental results
Experimental setup

- MPSoc platform with 8 homogeneous PEs
- For all PEs $f_n = 500$ MHz
- Variation in each PE
  - Die-to-die: $3\sigma / f_n = 12\%$
  - Within-die: $3\sigma / f_n = 10\%$
    - Measurements at 45 nm technology [3]
- 8 clock-frequency levels for each VFI
- A synthetic SDFG consisting of 15 actors
  - Enough parallelism for 8 PEs

Timing Yield vs. VFI Granularity

- Not all reductions in Nr. VFIs reduce the timing yield
  - VFI-5 results in a negligible reduction in timing yield

- VFI-2
  - 7% reduction
- FS:
  - 27% reduction

- VFI-5
  - More good dies?
Work Overview (recap)

- Variation and area characterization for each GB reduction

```
<table>
<thead>
<tr>
<th>GB_red %</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y_t % (VFI-8) SB</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>98</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>79</td>
<td>69</td>
<td>51</td>
<td>38</td>
</tr>
<tr>
<td>Y_t % (VFI-5) SB</td>
<td>99</td>
<td>99</td>
<td>99</td>
<td>98</td>
<td>97</td>
<td>94</td>
<td>88</td>
<td>78</td>
<td>67</td>
<td>49</td>
<td>36</td>
</tr>
</tbody>
</table>
```
Number of Good Dies

- 10 mm² chip, 7 mm² logic, 3 mm² SRAM, 0.03 mm² per CGU
- Design with (FB) and without fixed blocks (WFB)
- VFI-8, VFI-5 architectures

- 30% GB reduction (VFI-5)
  - 3.7% more good dies (FB)
- 40% GB reduction (VFI-5)
  - 7.7% more good dies (WFB)

- 4K wafers = 30M good dies
- 3.7% increase
  - 142 less wafers
- Wafer cost: $3K
  - Cost saving = $426K
Conclusions

• Framework to compute the probability distribution of application throughput in a system with VFI s under variation

• Heuristic VFI partitioning algorithm for maximized timing yield

• The framework is used to estimate the number of good dies
  • Dies that satisfy the throughput requirement

• It is possible to increase the number of good dies by using the proposed framework
Thank you for your attention
d.mirzoyan@tudelft.nl