Mitigating the effects of Dark Silicon

Ali Mustafa Zaidi

Supervisor: Dr. David Greaves
(Second Advisor: Prof. Alan Mycroft)

University of Cambridge
Computer Laboratory
Current Options for Computer Architecture

1. Performance Scaling
   - Energy Eff.
   - Productivity

2. Performance Scaling
   - Energy Eff.
   - Productivity

3. Performance Scaling
   - Energy Eff.
   - Productivity

4. My Goals:
   - Energy Eff.
   - Productivity

- Uniprocessor
- Multicore
- Custom Computing
- Future ???
The Dark Silicon Problem

Amdahl's Law

\[ \text{Speedup} = \frac{1}{\frac{(1-f)}{S} + \frac{f}{N*S}} \]

Utilization Wall

\[ f = \begin{cases} 1 & \text{18\%} \\ 0.99 & \text{7\%} \\ 0.9 & \text{3\%} \\ 0.75 & \text{3\%} \\ 0.5 & \text{3\%} \end{cases} \]

\[ \Rightarrow 45\text{nm} \to 8\text{nm} (32x \text{ resources}) \]

- CPU: 3.5x, GPU 2.4x (Conservative)
- CPU: 7.9x, GPU 2.7x (ITRS Scaling)

Proposal: Thriving in the Dark Silicon Era

Amdahl's Law +
Utilization Wall =

Dark Silicon

• Accelerate 'sequential' portion of code
  – Parallel portion already achieves high scalability

• Increase Energy Efficiency
  – 'Application Specific Hardware' → Custom Computing

Need to resolve Custom Computing issues:
1) Very complex development cycle
2) Poor performance on sequential, irregular code
#1: Fixing Productivity with 'Dynamic Scheduling'

- Supports conventional memory hierarchy
  - Full Support for C language features
- Compiler & Hardware become simpler
- Does not improve Sequential code performance

Why is sequential performance limited?

Control-Data Flow Graph

- Control flow is the primary constraint on ILP
- Conventional processors use aggressive branch prediction
  - 95+% accuracy
  - Single flow of control
- Custom hardware has very limited speculation
  - No Branch prediction
  - Single flow of control

```plaintext
for (i = 0; i < 100; i++)
    if (A[i] > 0) foo();
    bar();
```
Accelerating Sequential Code

- Limits of Control flow on Sequential code performance:
  - 7-12 IPC limit with Single Flow of control.
  - 5-20x increase in IPC with Multiple Flows of Control

- Must overcome control-flow constraints:
  - Without compromising energy efficiency!

- Proposed Solution:
  - Replace the Compiler IR: CDFG → VSDG
#2: Accelerating Sequential Code with the VSDG

- Data-dependences are explicit
- Control-flow is implicit
- Speculation explicit in IR
- Multiple Flows of Control Achievable!

```c
for (i = 0; i < 100; i++)
    if (A[i] > 0) foo();
bar();
```
Putting it All Together

Problem:

Utilization Wall + Programmability + Amdahl's Law = Dark Silicon

Solution:

Application Specific Hardware + Dynamic Execution Scheduling + VSDG-based Program Representation =

High Performance, High Energy Efficiency Hardware Implementation of legacy software & sequential programming models !!!

Implementation:

Any High Level Language → LLVM → VSDG → Bluespec SystemVerilog → ASIC / FPGA

Coarse-Grained, Reconfigurable Architecture (Asynchronous)
internal_int_transpose()
- From 'epic' Mediabench benchmark
- 2 level nested loop
- Compared with:
  - In-Order MIPS-64
  - CDFG SC from CMU,
  - 4-way OOO Superscalar

\[ \text{Power } P = f(R, A) \]
\[ R = \text{Resource Utilization} \]
\[ A = \text{Activity Ratios} \]

**Assumed:**
\[ A_{\text{SpatialComp}} = A_{\text{MIPS64}} \]

**Reality:**
\[ A_{\text{SpatialComp}} \ll A_{\text{MIPS64}} \]

**Energy**
\[ E = P \times T \]
\[ T = \frac{\text{Cycles}}{\text{Iteration}} \times \text{Num Iterations} \times \left( \frac{1}{F} \right) \]

\[ \text{Cycles/Iteration} \]

<table>
<thead>
<tr>
<th>VSDG x1</th>
<th>VSDG x2</th>
<th>VSDG x4</th>
<th>MIPS 64</th>
<th>CDFG SC</th>
<th>Superscalar</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>50</td>
</tr>
</tbody>
</table>

\[ \text{Power} \quad \text{mW} \]

<table>
<thead>
<tr>
<th>VSDG x1</th>
<th>VSDG x2</th>
<th>VSDG x4</th>
<th>MIPS-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>20</td>
<td>40</td>
<td>60</td>
</tr>
</tbody>
</table>

\[ \text{Energy} \quad \text{uJ} \]

<table>
<thead>
<tr>
<th>VSDG x1</th>
<th>VSDG x2</th>
<th>VSDG x4</th>
<th>MIPS-64</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.29</td>
<td>2.24</td>
<td>2.48</td>
<td>16.71</td>
</tr>
</tbody>
</table>
Assuming fixed frequency:

\[ R_{\text{Superscalar}} \approx R_{\text{InOrder}} \times 100 \]

\[ T_{\text{Superscalar}} \approx \frac{T_{\text{InOrder}}}{10} \]

\[ E_{\text{Superscalar}} \approx P_{\text{InOrder}} \times 100 \times \frac{T_{\text{InOrder}}}{10} \approx E_{\text{InOrder}} \times 10 \]
Project Plan

Analysis
(First Year Target, Includes PhD Thesis)

In Progress, Near Completion

Compiler
- VSFG IR
- LLVM → Bluespec
- Evaluation
- Comparison
- Static MLP

Prior Work:
- PipeRencher
- Tartan
- SCORE
- Tabula

Analysing Runtime Issues
- Dynamic MLP?
- Graph Partitioning, Paging

Analytical Model for Reconfigurable Hardware
- Asynchronous?
- Memory / Interconnect / Execution resource design
- Virtualization

Implementation

Compiler + Runtime Scheduler

Coarse Grained, Asynchronous Reconfigurable Architecture (Simulator)

Full System Evaluation

Implement Android OS on Spatial Computation Hardware

Analysis:

- In Progress, Near Completion
  - Compiler
  - Prior Work:
    - PipeRencher
    - Tartan
    - SCORE
    - Tabula

Implementation:

- Compiler + Runtime Scheduler
- Coarse Grained, Asynchronous Reconfigurable Architecture (Simulator)

Full System Evaluation:

- Implement Android OS on Spatial Computation Hardware

Analysis

- Analyse Runtime Issues
  - Dynamic MLP?
  - Graph Partitioning, Paging

Implementation

- Compiler + Runtime Scheduler
- Coarse Grained, Asynchronous Reconfigurable Architecture (Simulator)

Full System Evaluation

- Implement Android OS on Spatial Computation Hardware

Analysis:

- Analyse Runtime Issues
  - Dynamic MLP?
  - Graph Partitioning, Paging

Implementation:

- Compiler + Runtime Scheduler
- Coarse Grained, Asynchronous Reconfigurable Architecture (Simulator)

Full System Evaluation:

- Implement Android OS on Spatial Computation Hardware