The spinal cord is another brain

Spinal stretch reflexes support efficient hand control

Jeffrey Weiler, Paul L. Gribble and J. Andrew Grigg

Motor behaviour is most efficiently controlled by correcting not only disturbances that influence task success. It is currently thought that such control is computed within a transcortical feedback pathway. Here we show that, for postural hand control, even the fastest spinal feedback pathway can produce efficient corrective responses, forcing a re-evaluation of how the nervous system derives the control laws that support motor behavior.

Real-world actions require active control of many joints in the presence of internal and external disturbances. The simplest way for the nervous system to counteract disturbances is to ensure that all of a robotic exoskeleton and placed their hand at a while countering small flexion loads at the elbow a robot then mechanically flexed their elbow stretchi

It is currently thought that such control is computed within [the brain]. Here we show that ... the [spinal cord produces] efficient corrective responses, forcing a re-evaluation.
Example: *low-latency trading*

Communication & computation both done in the network switch
Where does FPGA efficiency come from?

Key Idea: design so that all pipelines operate at nearly 100% efficiency

Resembles GPU, but
- 10,000 independent scratchpads
- Flexible custom pipelines (SPs)
- Flexible custom interconnects
- No restriction on inter-“block” communication!
Key Idea: Design so that all pipelines operate at nearly 100% efficiency.

Key Idea: Direct, application-level, inter-node connectivity for ultra-low communication latency.

Resembles GPU, but:
- 10,000 independent scratchpads
- Flexible custom pipelines (SPs)
- Flexible custom interconnects
- No restriction on inter-'block' communication!

High efficiency extends to the network:

~10,000

80 x 50Gbs + 24 x 100Gbs
Spatial Computing example – MD force pipeline

\[ F^J_{ij} = \sum_{j \neq i}^{\epsilon_{ab}} \left( \frac{\sigma_{ab}}{|r_{ji}|^{14}} - 24 \left( \frac{\sigma_{ab}}{|r_{ji}|^8} \right) \right) \vec{r}_{ji} \]

\[ F^C_i = \frac{q_i}{4\pi} \sum_{j \neq i}^{\epsilon_{ab}} \left( \frac{1}{|r_{ji}|^3} \right) \vec{r}_{ji} \]

Distance computation

Force computation

Full force pipeline
Spatial Computing – hierarchical replication
Compile statement →

\[ a + b \times c - d / (b \times c) \]

3-address code

<table>
<thead>
<tr>
<th></th>
<th>op</th>
<th>arg₁</th>
<th>arg₂</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>*</td>
<td>b</td>
<td>c</td>
<td>t₁</td>
</tr>
<tr>
<td>2</td>
<td>+</td>
<td>a</td>
<td>t₁</td>
<td>t₂</td>
</tr>
<tr>
<td>3</td>
<td>*</td>
<td>b</td>
<td>c</td>
<td>t₃</td>
</tr>
<tr>
<td>4</td>
<td>/</td>
<td>d</td>
<td>t₃</td>
<td>t₄</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>t₂</td>
<td>t₄</td>
<td>t₅</td>
</tr>
</tbody>
</table>

Quadruples

<table>
<thead>
<tr>
<th></th>
<th>op</th>
<th>arg₁</th>
<th>arg₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>*</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>1</td>
<td>+</td>
<td>a</td>
<td>(0)</td>
</tr>
<tr>
<td>2</td>
<td>*</td>
<td>b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>/</td>
<td>d</td>
<td>(2)</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>(1)</td>
<td>(3)</td>
</tr>
</tbody>
</table>

Triples

Syntax tree

DAG

https://nptel.ac.in/content/storage2/courses/106108113/module5/Lecture17.pdf
int foo(char x, char a, char b, char c) {
    char y;
    y = x*a+b+c;
    return y;
}
void foo(int in[3], char a, char b, char c, int out[3]) {
    int x, y;
    for(int i = 0; i < 3; i++) {
        x = in[i];
        y = a*x + b + c;
        out[i] = y;
    }
}
Original Design
Result from HLS
HLS result after refactoring input code
Once Per Compiler

For Every Compilation

PROBES

FRONT-END HLS COMPILER

IR

STATIC PROFILER

REPORT

CODE TRNSFRM

APP HLL

PRE PROCESSOR

APP HLL

FULL HLS COMPILER

HDL
HLL Input → GCC → Optimization Controller → Optimization Sequence
GCC

**Front**

- HLL Input
- Syntax Graph

**Middle**

- Optimization Controller
- Optimization Sequence

** GCC SSA Code**
Open Source HDL tool

GCC

Front
- User HLL Source
- HLL Input
- Syntax Graph Dump

Middle
- Syntax Graph
- Optimization Sequence
- HLL Encoder
- Encoded HLL

Optimization Controller
- GCC SSA Code

GCC-LLVM Converter

User HLL Source

HLL Input

Syntax Graph Dump

Optimization Sequence

HLL Encoder

Encoded HLL

Inference Engine

HDL Generator

Open Source HDL tool

v4
What FPGA-centric clouds and clusters look like*

*Based on a Microsoft figure