ECE 695R: SYSTEM-ON-CHIP DESIGN

Module 2: HW/SW Partitioning
Lecture D: Automatic Custom Instruction Generation
- Tensilica XPRES

Anand Raghunathan
raghunathan@purdue.edu

Fall 2014, ME 1052, T Th 12:00PM-1:15PM
Automatic Instruction Set Extension in Practice: Tensilica XPRES

Source: Grant Martin, Tensilica/Cadence
XPRES: Automating ASIP generation

**Compile Original C/C++ Code with XCC**

**Run XPRES Compiler**

**Designer selects “best” configuration**

**Run Xtensa Processor Generator**

**Compile & run original, unmodified C code**

```
int main()
{
    int i;
    short c[100];
    for (i=0;i<N/2;i++)
    {
```

```
Option: manually refine configuration
```

Build processor, including RTL, SW, etc.

Build chip, system
Techniques Employed by XPres Compiler: 3 kinds of parallelism

- Instruction-Level Parallelism: FLIX
  - Bundling independent operations

- Data-Level Parallelism: SIMD / Vector
  - Single instruction, multiple-data operations

- Pipeline Parallelism: Instruction Fusion
  - Merging of compound, dependent operations
  - E.g. a multiply-add or a multiply feeding a shift

- All three techniques can be combined
  - E.g. Parallel FLIX execution of Fused-SIMD operations
Example

```c
for (i=0; i<n; i++) c[i] = (a[i] * b[i]) >> 4;
```

- **VL** – vector length
- **IW** – issue width
- **L/S** – memory units
Example: SAD (Sum of Absolute Differences)  
Automatic Solution Search

Wide Range of Choices of Performance Increase vs Hardware Cost

<table>
<thead>
<tr>
<th>i</th>
<th>Speedup -v- Base processor</th>
<th>Gates Added</th>
<th>SIMD Factor</th>
<th>FLIX Width (Slots)</th>
<th>Load / Store Units</th>
<th>Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18.6x</td>
<td>49,681</td>
<td>8</td>
<td>3</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>2</td>
<td>14.1x</td>
<td>35,439</td>
<td>8</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>13.9x</td>
<td>27,082</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>10.1x</td>
<td>19,547</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>5</td>
<td>9.0x</td>
<td>15,527</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>Yes</td>
</tr>
<tr>
<td>6</td>
<td>6.8x</td>
<td>8994</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>7</td>
<td>5.3x</td>
<td>7158</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>8</td>
<td>1.5x</td>
<td>301</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>Yes</td>
</tr>
</tbody>
</table>
Lectures #10: Summary

• Automatic Instruction Set Extension is a challenging problem
  – Size and complexity of design space
  – Phased approach: Local generation, re-use identification, and global selection
  – Commercial tools starting to appear (Tensilica’s XPRES)

• Automatic Instruction Set Extension in Practice
  – Tensilica’s XPRES