ECE 695R:
SYSTEM-ON-CHIP DESIGN

Module 2: HW/SW Partitioning
Lecture A1: Automatic Custom Instruction Generation

Anand Raghunathan
raghunathan@purdue.edu

Fall 2014, ME 1052, T Th 12:00PM-1:15PM
Lecture #10 Overview

• Application Specific Instruction Processors (ASIPs) are a promising approach to HW/SW partitioning

• However, designing ASIPs requires significant design effort

• How do we reduce the design effort?
  – Clean interfaces and automation
  – Specifically focus on the problem of automatic Instruction Set Extension (ISE)
Custom Instructions: Manual Approach

1. Identify potential new instructions
2. Design custom instruction HW
3. Invoke custom instructions
4. Verify functional correctness, performance, power
5. Profile, read source code
6. Understand selected code regions
7. Rewrite source code

Slow and error-prone
Automating Instruction Set Extension

- Starting with a program, automatically create custom instructions and a modified program that uses them.
- Constraints:
  - Performance/Area/Power
  - Architectural constraints on the interface of the custom instruction.
Preliminaries: Program Representation

• Programs of real-world complexity need to be represented hierarchically!
  – Top-level: Function call graph
  – Function level: Control/Data Flow Graphs
Function Call Graph: Example

- Vertices: Functions, Edges: Caller-Callee relationship

Example: MPEG2 Decoder
Control Flow Graph: Example

• Code within each function is usually divided into **Basic Blocks**
  – BB can have only one entry point

• Control-flow graph (CFG) for a function
  – Vertex = basic block, edge = control dependency
Data Flow Graph: Example

- Code within a basic block can be expressed in terms of the primitive operations within it
- Data flow graph for a basic block
  - Vertex = operation/variable, edge = data dependency
Search Space for Custom Instructions

- Size of local search space
  - Q: How many custom instructions can be generated from a basic block with n operations? \( ^nC_1 + ^nC_2 + \ldots + ^nC_n = O(2^n) \)

- Complexity of search space
  - Seemingly “similar” custom instructions may widely differ in performance

```c
BYTESWAP(unsigned s, unsigned char do_swap)
{
    unsigned ss, s1, s2, s3, s4,
            s5, s6, s7, s8, result;
    s1 = s<<24;
    s2 = s<<8;
    s4 = s>>8;
    s6 = s>>24;
    s3 = s2 & 0xff0000;
    s5 = s4 & 0xff00;
    s7 = s1 | s3;
    s8 = s5 | s6;
    ss = s7 | s8;
    /* Global count of #words processed */
    if (do_swap) SWAPPED_COUNT++;
    result = do_swap ? ss : s;

    return result;
}
```
Search Space for Custom Instructions

• Size of global search space
  – $\prod N_i$, where each function $f_i$ has $N_i$ instructions
  – Area/power constraints may dictate that a sub-set be selected