Region Formation for ILP Processors: An Overview

Recap...Instruction Scheduling and Register Allocation

- List scheduling
  - Deterministic scheduling for instructions in a basic block
  - Create priority list and greedily schedule by parsing down the list
  - An instruction that is ready and has resources available is scheduled as early as possible (based on its rank/priority)

- Register allocation
  - Model as a graph coloring problem
  - Construct interference graph, and color the nodes
  - May need to ‘spill’ nodes if not enough registers

Scheduling for ILP: Outline

- Basic block has little parallelism
  - Very little ILP extracted as a result
  - Need to provide larger regions to the scheduler

- Definitions
  - Basic block
  - Superblock
  - Hyperblock

- Region Formation Background
**Background**

- Implications of function based compilation
  - hidden memory aliasing behavior
  - optimization
  - control flow structure
- Aggressive optimizing compiler reduces the side effect by using function inlining

**Problem with function based compilation**

- In function based compilation..
  - While compiling function A, the scope of the compilation is limited to it.
  - While compiling function B, the fact that function B is part of a cycle is hidden
- Solution
  - Function Inlining
  - Code expansion is large

**Example:**

**Function based compilation**

![Function A and Function B Diagram](image)

**The effect of compilation size**

![Graph showing compilation time and execution performance](image)
Definitions: Region Formation

- A region is a unit of program instructions that will be examined together for scheduling.
- The smallest region used is called the “basicblock”
- There are three other types of regions explored. They are both composed of basicblocks. They are the trace, the superblock and the hyperblock.

Features of region based compilation

- Compiler is in complete control over the size and contents of the compilation unit
- The size of the compilation unit is typically smaller than functions
  - reducing the impact of the algorithmic complexity
- The use of profiling information to select regions allows the compiler to select compilation units than more accurately reflect dynamic behavior of the program
  - allow the compiler to produce more compact optimized code
- Each region may be compiled completely before compilation proceeds to next region
  - all function-oriented compiler transformations may be applied.

Types of region

Arbitrary Region

- Trace
- Super Block
- Hyper Block
- Function

Arbitrary region formation

- Select a seed block, s, which is the most frequently executed block not yet in a region
- find all desirable successor y of a block x for all x in a region where Succ(x,y) satisfies,
  \[ \frac{W(x \rightarrow y)}{W(x)} \geq T_j \land \frac{W(y)}{W(s)} \geq T_i \]

W(x) : frequency
x → y : control flow edge from x to y
- How to decide T & Ts?
**Example: Arbitrary region**

Block B is the most frequently executed. “Seed”

**Trace**

- Ordered sequence of basic blocks connected by control flow
- Choose a “seed” block with the highest expected frequency
- From seed, expand forward/backward in the control flow graph, picking the unscheduled block with the highest expected frequency

**Definitions: The BasicBlock**

- “A basic block is a straight-line code sequence with no transfers in or out, except at the beginning or end.” Hennessy and Patterson

**Trace Picking**

E is not included, since F has higher frequency

Very Long Instruction Word Format
Traces


- Main Ideas:
  - Choose a program segment that has no cyclic dependences.
  - Choose one of the paths out of each branch that is encountered.

  more...

Traces (Contd.)

- Use statistical knowledge based on (estimated) program behavior to bias the choices to favor the more frequently taken branches.

- This information is gained through profiling the program or via static analysis.

- The resulting sequence of basic blocks including the branch instructions is referred to as a trace.

Region Based Scheduling

- Treat a region as the input to the scheduler
  - How to schedule instructions in a region?
  - Can we move instructions to any “slot”?
  - What do we have to watch out for?

Scheduling Algorithm

- Input is the Region (superblock, trace, etc.)
- Use List Scheduling algorithm!
  - Treat movement of instructions past branch and join points as “special cases”
Detecting Speculation and Replication Structurally

- Need tests that can be performed quickly to determine which of the side-effects have to be addressed after code-motion.

- Preferably based on structured information that can be derived from previously computed (and explained) program analysis.

- Decisions that are based on the Control (sub) Component of the Program Dependence Graph (PDG).

The Four Elementary but Significant Side-effects

- Consider a single instruction moving past a conditional branch:

The First Case

- This code movement leads to the instruction executing sometimes when the instruction ought not to have: speculatively.

The Second Case

- Identical to previous case except the pseudo-dependence edge is from A to the join instruction whenever A is a “write” or a def.

- A more general solution is to permit the code motion but undo the effect of the speculated definition by adding repair code

An expensive proposition in terms of compilation cost.
The Third Case

- Instruction A will not be executed if the off-trace path is taken.
- To avoid mistakes, it is replicated.

more...

Super Block

- A trace with a single entry but potentially many exits
- Simplifies code motion during scheduling
  - upward movements past a side entry within a block are pure replication
  - downward movements past a side entry within a block are pure speculation
- Two step formation
  - Trace picking
  - Tail duplication

The Fourth Case

- Similar to Case 3 except for the direction of the replication as shown in the figure above.

Definitions: The Superblock

- The superblock is a scheduling region composed of basic blocks with a single entry but potentially many exits
- Superblock formation is done in two steps
  - Trace selection
  - Tail duplication
Seed block

Background: Region Formation

The SuperBlock

Super block formation and tail duplication

Background: Region Formation

The HyperBlock

- Single entry/ multiple exit set of predicated basic blocks (if-conversion)
- There are no incoming control flow arcs from outside basic blocks to the selected blocks other than the entry block
- Nested inner loops inside the selected blocks are not allowed
- Hyperblock formation procedure:
  - Trace selection
  - Tail duplication
  - Loop peeling
  - Node splitting
  - If-conversion
Definitions: The Hyperblock

- Single entry/multiple exit set of predicated basic block (if conversion)
- There exit no incoming control flow arcs from outside basic blocks to the selected blocks other than the entry block
- There exist no nested inner loops inside the selected blocks

A larger scheduling region exposes more instructions that may be executed in parallel.

Hyper block formation procedure

- Tail duplication
  - remove side entries
- Loop Peeling
  - create bigger region for nested loop
- Node Splitting
  - eliminate dependencies created by control path merge
  - large code expansion
- After above three transformations, perform if conversion
If conversion

Experiment shows that 85% of the execution time was contained in regions with fewer than 250 operations, when region size is not limited.

There are some regions formed with more than 10000 operations. (May need limit)

How can I decide the size limit?
  – Open Issue

Summary: Region Formation

In general, the opportunity to extract more parallelism increases as the region size increases. There are more instructions exposed in the larger region size.

The compile time increases as the region size increases. A trade-off in compile time versus run-time must be considered.

Region Size Control

Region Formation in Trimaran

A research infrastructure used to facilitate the creation and evaluation of EPIC/VLIW and superscalar compiler optimization techniques.

Forms 3 types of regions:
  – Basic blocks
  – Superblocks
  – Hyperblocks

Operates only on the C language as input

Uses a general machine description language (HMDES)

This infrastructure uses a parameterized processor architecture called HPL-PD (a.k.a. PlayDoh)

All architectures are mapped into and simulated in HPL-PD.