Speculative Predication Across Arbitrary Interprocedural Control Flow

LCPC 1999

Hank Dietz
School of Electrical and Computer Engineering
Purdue University
Department of Electrical Engineering
University of Kentucky

hankd@engr.uky.edu

http://dynamo.ecn.purdue.edu/~hankd

(765) 494 3357

The Current Generation Of Chips

- Superscalar pipelining
- Out-of-order execution
- Lots of circuitry & power
- Off-chip interface dominates performance

Next-Generation Chips Are Different?

- Static scheduling helps...
  - SIMD Within A Register (SWAR)
  - Explicit prefetch
- When the compiler schedules...
  - Small-scale VLIW
  - Explicit speculation with guards
  - Multiprocessor chips

VLIW's A Win, But...

- Obviously a win
  - Much simpler circuitry
    (even meshes well with memory bus)
  - Lower power
  - Compiler technology from the early 1980s
    (with slow & steady improvement)
- Only problem is object code compatibility across generations of implementations with different parallelism

Explicit Speculation With Guards

- What is it?
  - Speculation: schedule instructions to execute before you know if they need to execute
  - Guards: avoid branches and code motion constraints by making instruction results conditional
  - A major feature of IA64...
- Advantages...
  - Efficient, scalable, hardware
  - Scheduling can use a larger "window"

Basic Compilation For Guards

```
if (cond) { c = a + b; } else { c = a - b; }
```

Becomes the branch-free block:

```
g = (cond);
where (g) c = a + b;
where (!g) c = a - b;
```

Better Compilation For Guards

- What limits speedup?
  - Probability speculated instructions are useful drops exponentially
  - Convert only conditional forward jumps
- Common Subexpression Induction (CSI):
  Improves probability that speculated instructions are useful
- Meta-State Conversion (MSC):
  Convert arbitrary flow graphs to speculative form

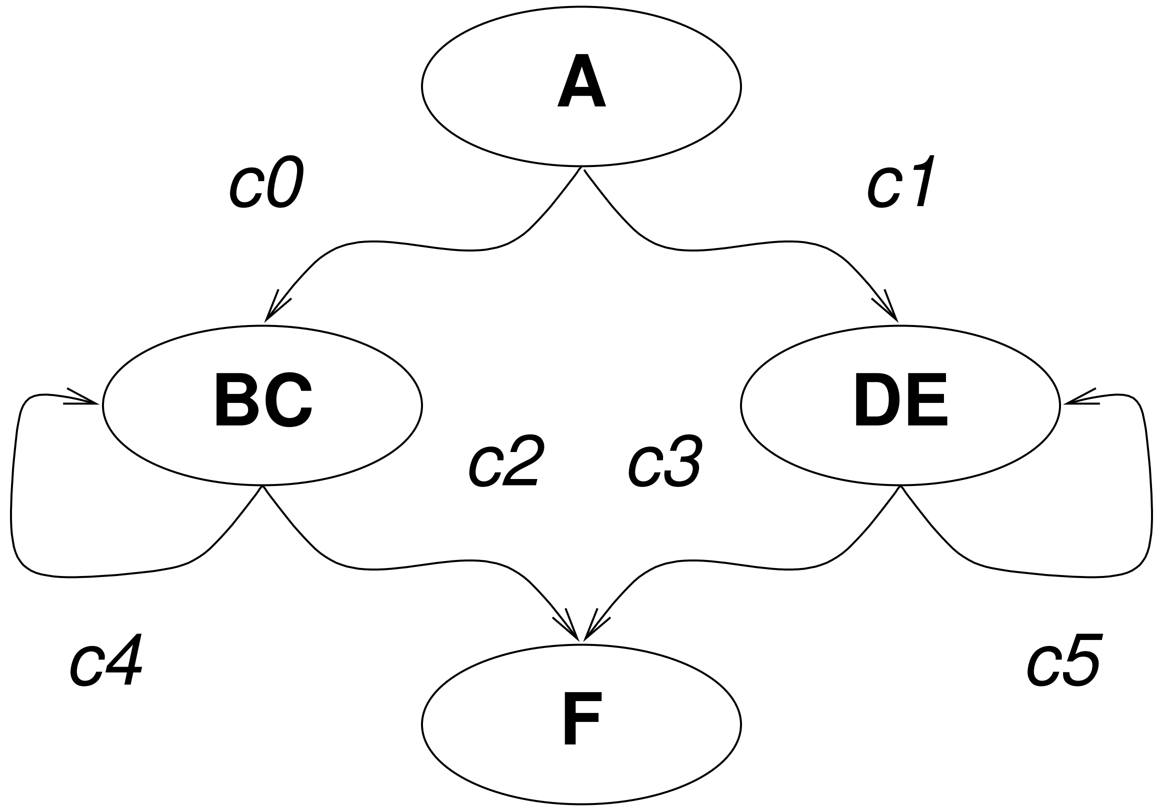
Meta-State Conversion (MSC)

- A state space transformation allowing loops (forward and backward branches), function calls and returns
- In 1993, for MIMD-on-SIMD...
  - SIMD meta state is set of MIMD states that could exist simultaneously
  - Preserves relative timing of MIMD execution
- In 1999, for guarded speculation...
  - Speculative meta state is non-speculative core state plus guarded speculative states
  - Preserves dependence properties

Simple Example Code

```
if (A) {
    do { B } while (C);
} else {
    do { D } while (E);
}
F
```

Simple Example State Graph



What About Function Call/Return?

- CALL is really a jump
  (stack ops do not change control flow)
- RETURN is really an N-way jump
  (jump to after one of the call points)
- Recursion changes nothing!

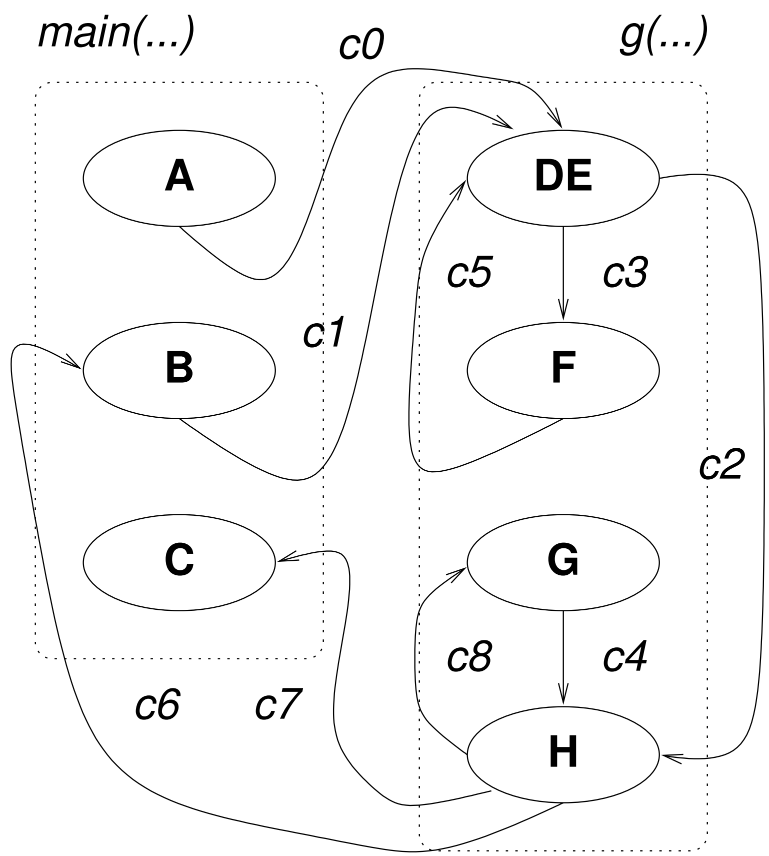
Recursive Function Example

```
main(...)
{
 A
  g(...);
 B
  g(...);
 C
}
```

```
g(...)
{
 D
  if (E) {
  F
   g(...);
  G
  }
 H
  return;
}
```

```
main:              g:
 A                  D
  goto g;            if (E) {
x:                   F
 B                    goto g;
  goto g;          z:
y:                   G
 C                   }
  exit(...);        H
                     switch (...) {
                     Case x: goto x;
                     case y: goto y;
                     case z: goto z;
                     }
```

Recursive Function Example State Graph



Speculative Meta State Conversion

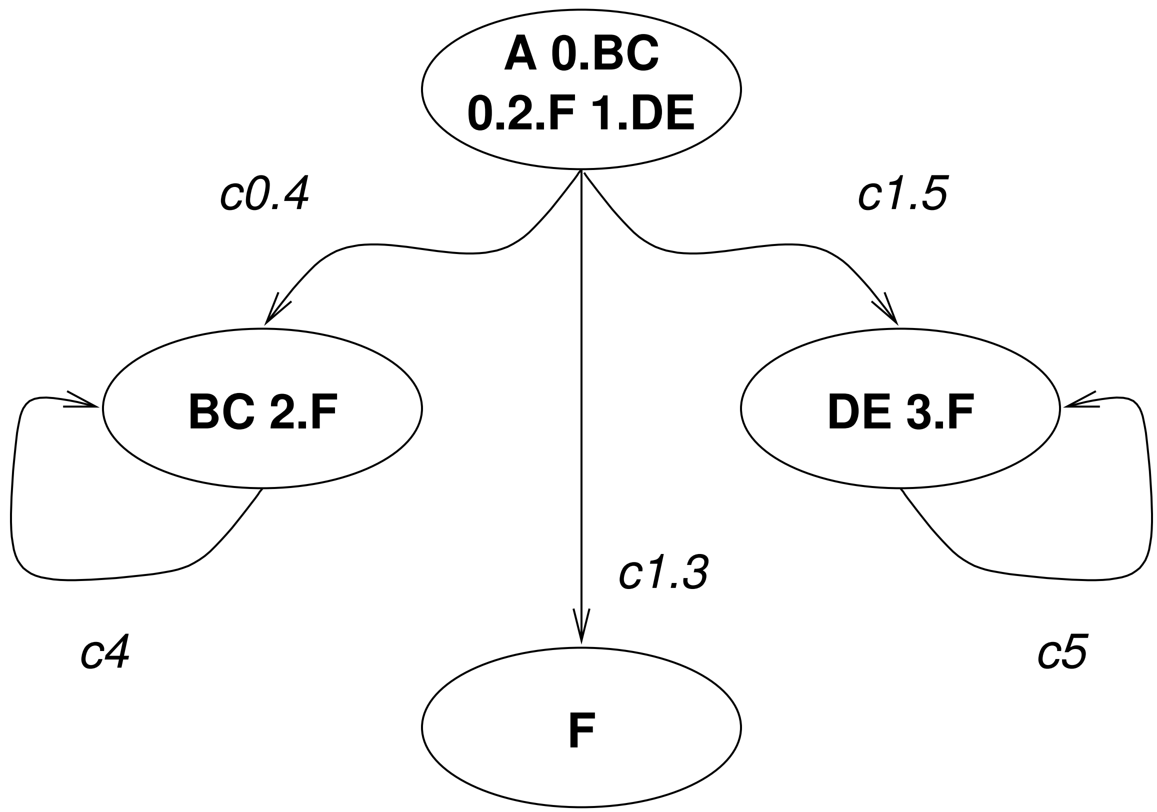
- Similar to NFA-to-DFA or MIMD-to-SIMD conversion
- Algorithm overview:
  - Worklist of states, begins with start state
  - Each state from the worklist is the non-speculative core of exactly one meta state
  - Use a recursive reaching algorithm to add guarded speculative states to the core
  - Where speculation ends, add an exit arc and add the target state to the worklist
- Can mark specific states as non-speculative

Properties Of The Algorithm

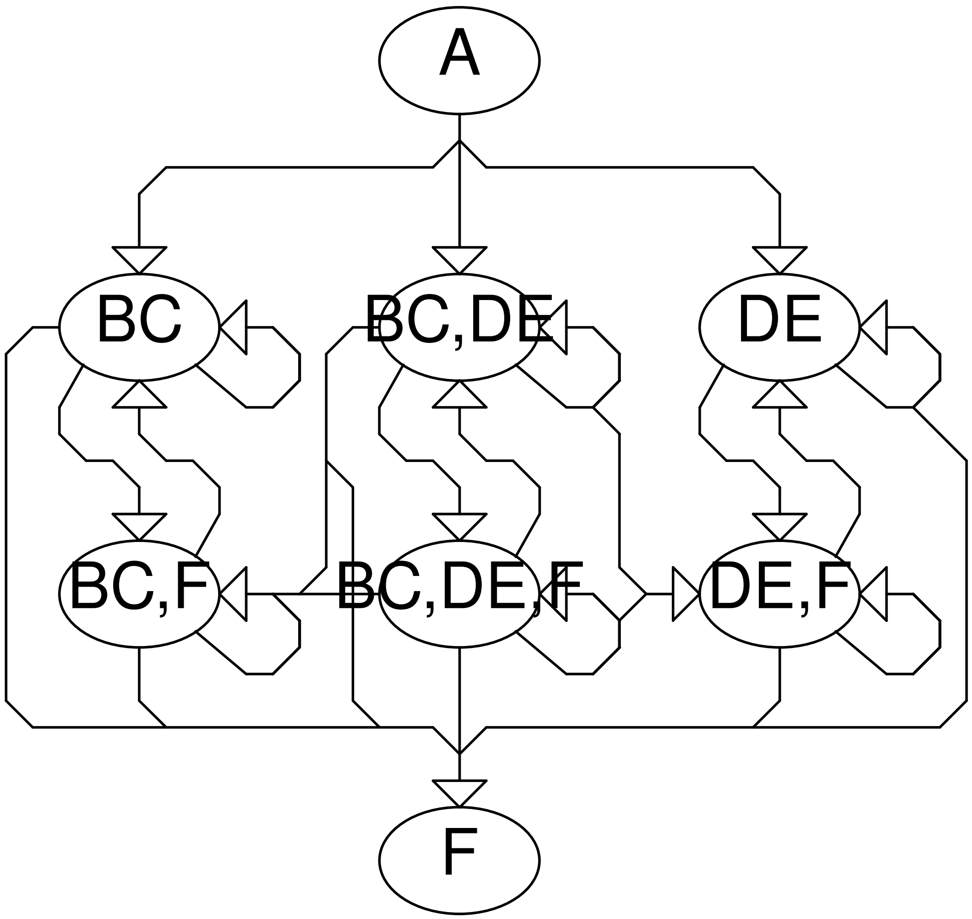
- Each state is a core at most once
  - There are at most N meta states for N states!
  - By forbidding duplication of states within a meta state, complexity of a meta state is O(N) or less
  - Complexity of the complete algorithm is $O(N^2)$ or less
- Tunable maxdepth or cost-based cutoff
- Forbidding state replication blocks speculatively executing loop bodies for multiple iterations, but this can be fixed by partial unrolling

Simple Example Speculative Meta-State Graph

maxdepth = infinity

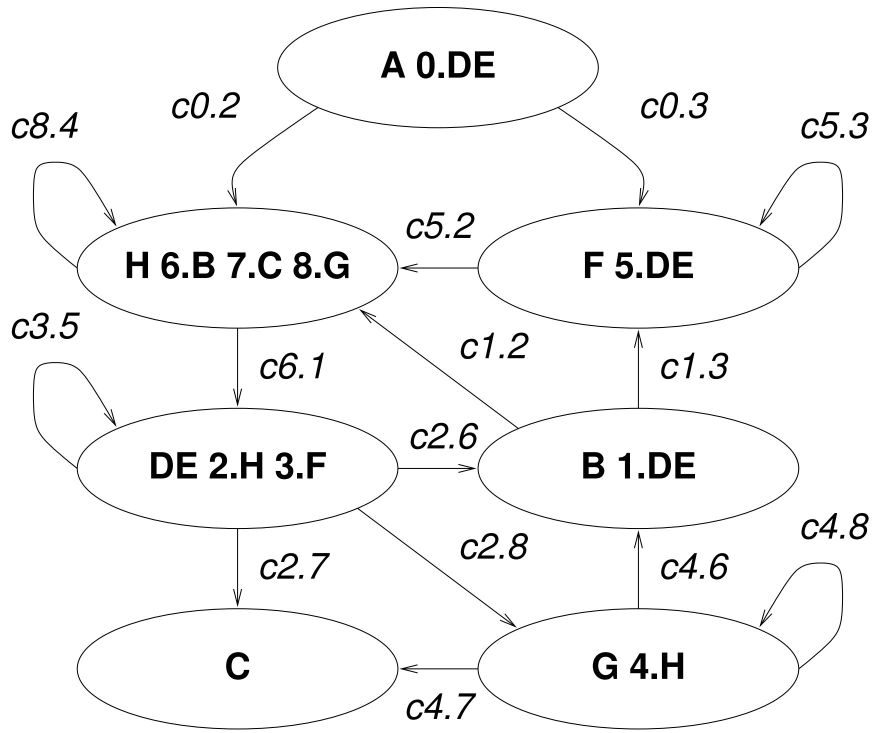


Simple Example SIMD Meta-State Graph



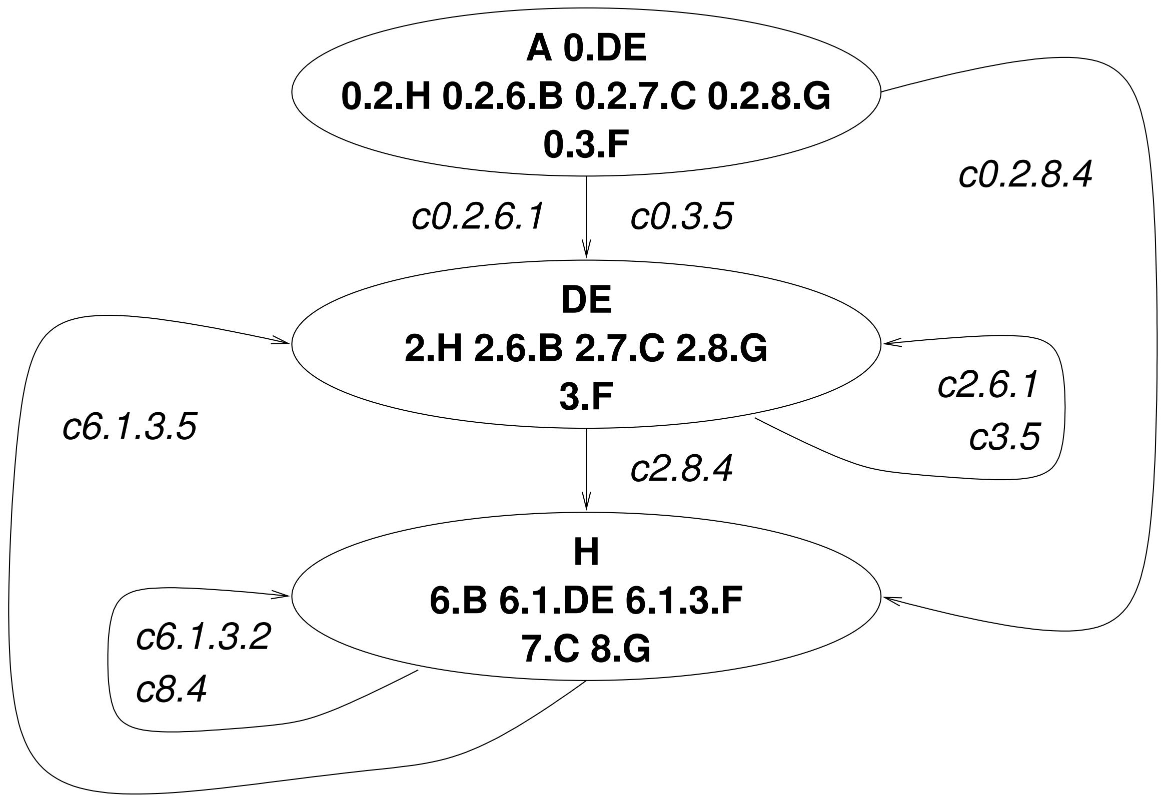
Recursive Function Example Meta-State Graph

maxdepth = one



Recursive Function Example Meta-State Graph

maxdepth = infinity



Coding the Meta State Automaton

- Guard expressions can be optimized (e.g., by algebraic simplifications)
- Multiway branch encoding
  - Hash functions or jump tables
  - Guarded loads of jump target address
- Common Subexpression Induction...

Conclusions

- Next-generation processor chips **require** new compiler technology; Instructions are just as important as data
- Back-to-basics (e.g., state graph) approaches can be simple, very general, and highly efficient
- This paper gives only the "sanitized" theory...
- Predicated speculative designs, such as IA64, are very complex -- much experimental work needs to be done to see how to tune the speculation and coding