Estimating Timing Profiles for Simulation of Embedded Systems

Andrew H. Chan
May 15, 2009

Introduction

- Timing estimation
- Timing and Concurrency
  - Many parts working together
  - Need to be aware of timing characteristics
- Given hardware, software and control flow, estimate execution time

Motivation

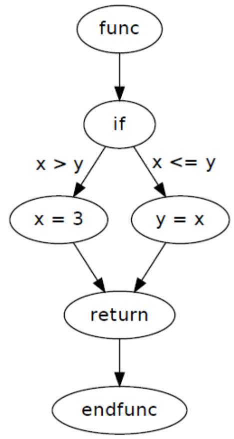
- Access Point Event Simulation of Legacy Embedded Software Systems (APES)
  - Discrete event model in Ptolemy
  - Needs timing estimates of code fragments
- Easier/less expensive to simulate in software without maintaining actual hardware
- Might not have direct access to hardware

Motivation

- Explore timing behavior of hardware
- Guided testing to assess timing characteristics
- Simplify problem by deciding on both software and hardware

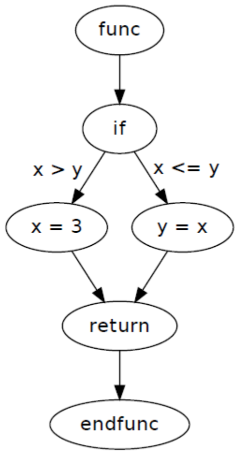
Timing Model

- Control Flow Graph (CFG)
  - Conditionals, branches
  - Basic blocks
  - Directed acyclic graph



Example

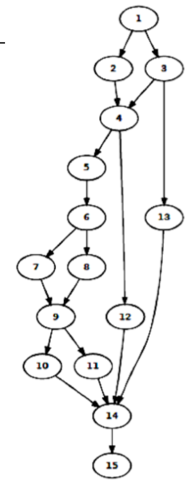
```
void func(int x, int y)
   {
   if(x > y) {
     x = 3;
}
   else {
     y = x;
}
   return;
}
```



Algorithm

- Represent paths with an "edge" vector
- Vector of 0s and 1s, where each component corresponds to an "edge"

- $\mathbf{x} = \begin{bmatrix} 1. \\ 1. \\ 0. \\ 1. \\ 0. \\ 0. \end{bmatrix}$



Algorithm

- Calculate 2-barycentric spanner
  - Good representative set of basis paths
  - Every path can be represented by a linear combination with coeff less than 2
- Sample on these paths to obtain timing information
- Only get total path execution times
  - No execution times for individual basic blocks

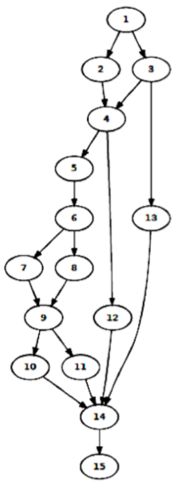
Algorithm

- B is a matrix with basis vectors as rows
- **v** is a vector of the average execution times for the basis vectors
- **w** is the "edge weight" vector
- $B\mathbf{w} = \mathbf{v}$ ⇨ $\mathbf{w} = B^{-1}\mathbf{v}$
- Estimate for path **x**
  - $\mathbf{w}^T\mathbf{x}$ = estimated time

Algorithm

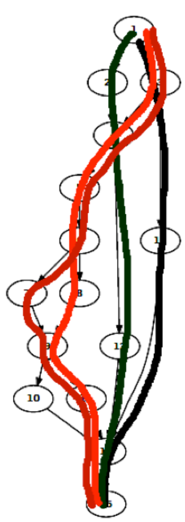
- Estimate "edge" weights
- Path lengths are more important
  - Not all edges are assigned accurate weights, but it doesn't matter
  - Only the overall path length matters

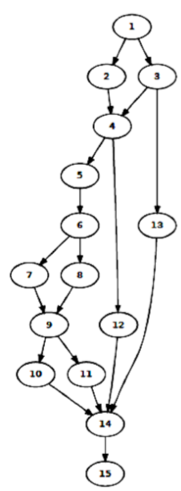
Example



- CFG for Altitude
- 11 paths
- 6 basis paths

Example





Example

- Record average time for each basis path
- Place in vector
- **v** = $\begin{pmatrix} 221. \\ 872. \\ 228. \\ 1213. \\ 185. \\ 1196. \end{pmatrix}$
- Basis Matrix
- B =

$$\begin{pmatrix} 1. & 0. & 1. & 0. & 0. & 0. \\ 0. & 1. & 0. & 1. & 0. & 1. \\ 0. & 0. & 1. & 0. & 0. & 1. \\ 0. & 0. & 0. & 1. & 0. & 1. \\ 0. & 0. & 0. & 0. & 1. & 1. \\ 0. & 1. & 0. & 0. & 0. & 1. \end{pmatrix}$$

Example

- Calculate B^{-1}
- Compute $B^{-1}\mathbf{v} = \mathbf{w}$
  - **w** = $\begin{pmatrix} 1530. \\ -341. \\ -1309. \\ -324. \\ -1352. \\ 1538. \end{pmatrix}$
- Take the "edge" vector of the path you want to estimate
  - **x** = $\begin{pmatrix} 1. \\ 1. \\ 0. \\ 1. \\ 0. \\ 0. \end{pmatrix}$
- Estimated time = $\mathbf{w}^T\mathbf{x}$
  - 865 estimated cycles
  - 851 actual cycles

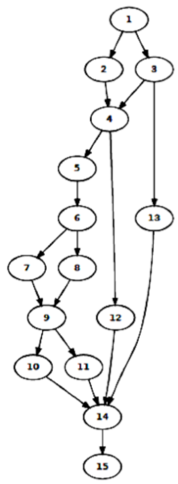
Implementation

- SimIt ARM simulator
- CREST: branch coverage
- CIL: C front-end, instrumentation
- Yices: satisfiability solver
- SciPy and Numpy

Implementation

- SimIt ARM 2.1 simulator
  - Cycle-accurate simulator for the StrongARM microprocessor
  - ARM V4 instruction set architecture
  - 206MHz processor

Experiments



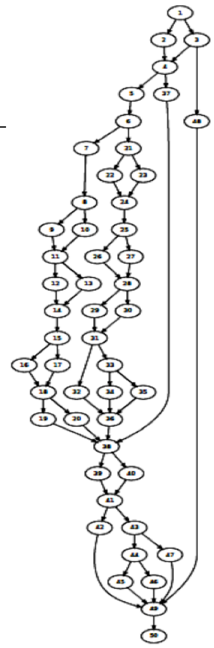
- CFG for Altitude
- 11 paths
- 6 basis paths

Experiments

- Altitude, 12 lines of code
  - 15 nodes, 19 edges
  - 6 basis paths, 11 possible paths in CFG
  - 5 non-basis paths tested
  - Mean: 794 cycles, Std Dev: 54%
  - Average estimation diff: 0.9 %
  - Max estimation diff: 1.62 %

Experiments

- Climb Control
  - CFG



Experiments

- Climb Control
  - 50 nodes, 66 edges
  - 18 basis paths, 657 possible paths in CFG
  - 94 non-basis paths tested
  - Mean: 1178 cycles, Std Dev: 35%
  - Average estimation diff: 2.5 %
  - Max estimation diff: 12.7 %

Conclusion

- Important to choose a good set of basis paths
- Context-switching effects are not handled
- Data-dependent effects
- Can be used to estimate time given trace of execution

Questions?