

Pipeline Modeling for Timing Analysis

Markus Klein, Johannes Kasberger

Caches & Pipelines

- Überbrücken Geschwindigkeitslücke zwischen CPU und RAM
- Erhöhen Geschwindigkeit
- Verschlechtern Vorhersagbarkeit

Ohne Caches?

- Ohne Caches hat eine 300MHz CPU teilweise Performance einer 20MHz CPU mit Caches
- z.B. fly-by-wire benötigt viel Rechenleistung
- → Caches sind notwendig

Motorola ColdFire 5307

- non-trivial real-life processor
 - instruction prefetching
 - branch prediction
 - caches
 - pipeline
 - memory bus

Motorola ColdFire 5307

- very popular microcontroller
- Verwendung z.B. bei Airbus Flugzeugen

Probleme bei WCET

- Vereinfachende Annahmen verhindern präzise Abschätzung
- Mit Pipeline&Caches Model bekommt man bessere Ergebnisse

Bisherige Ansätze

- ILP wurde überall verwendet
- Zu einfache Betrachtung von Pipelines
- Fehlende Betrachtung von emergenten Effekten von Cache-Pipeline-Kombination
- Branch Prediction wurde ignoriert

Überabschätzung

- Nur ein Model das branch prediction, instruction prefetching, speculative execution und data access zusammen betrachtet, kann gute Ergebnisse liefern

Ansatz

- Zwei Phasen bei Analyse
 - execution modeling
 - program path analysis

Ansatz

- Fertig gelinktes Executable als Ausgangspunkt
- Vorteil: Toolchain für Luftfahrtindustrie muss nicht neu zertifiziert werden
- Control Flow Graph (CFG) aus Executable extrahiert
- Leicht portierbar

Analyse

- Wertanalyse über Registerinhalte/
Speicherzugriffe
- Kombinierte Caches&Pipeline
Zustandsanalyse (mögliche Kombinationen)
- Mit Loop Bounds → ILP Lösung für WCET

Analyse

- CFG in basic blocks teilen
- Semantic der Instruction wird in abstrakte Darstellung umgewandelt
- Knoten im CFG sind Instructions
Kanten sind transfer functions

Transfer Function

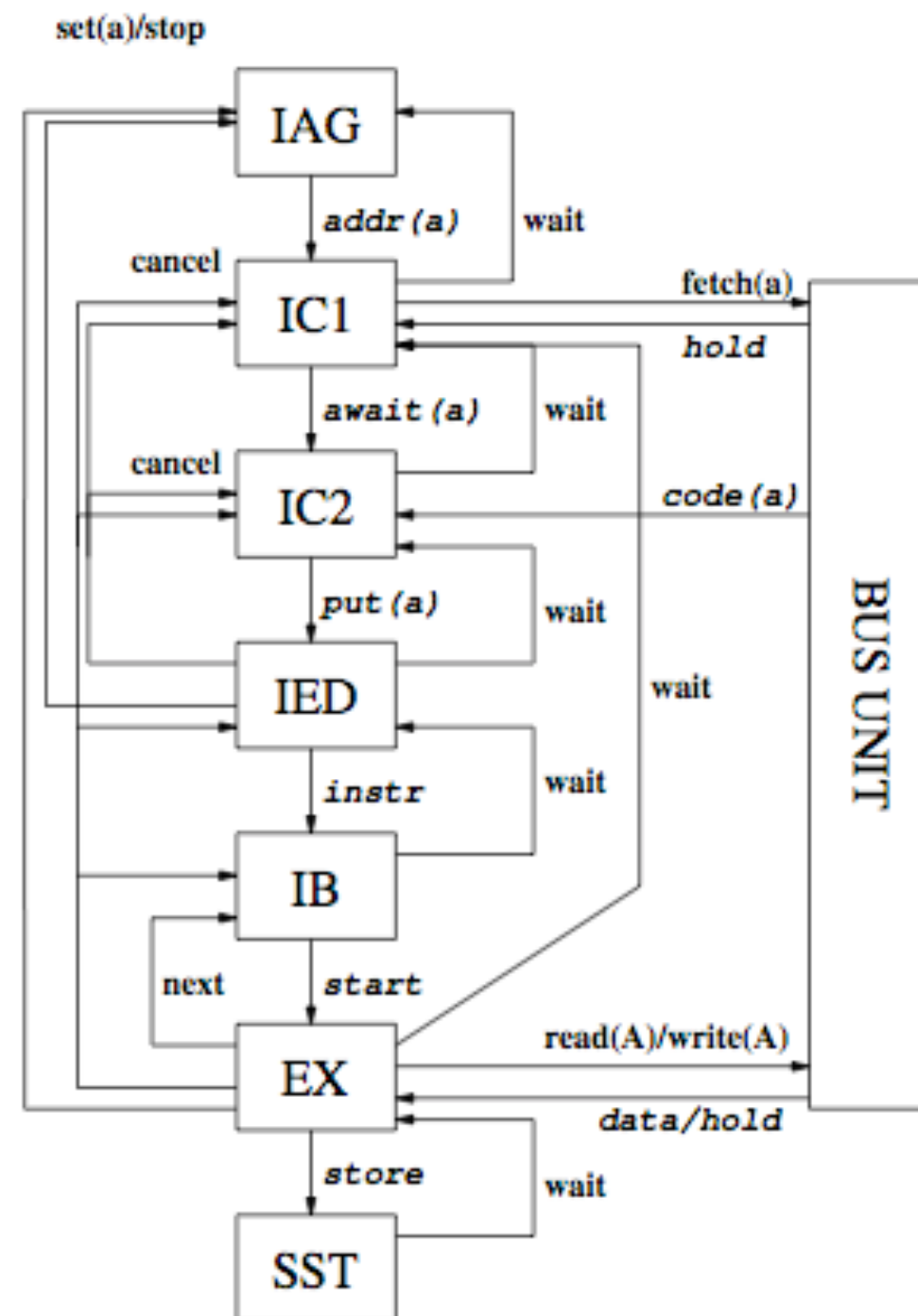
- Simuliert Effekt der Instruction auf Pipeline
- State von Pipeline&Caches wird berücksichtigt



Pipeline Model

- Wird als State Machine modelliert
- State enthält Informationen über das Timing
- Jeder State wird aufgeteilt in Units (mit eigenen State)
- Units kommunizieren über Signale

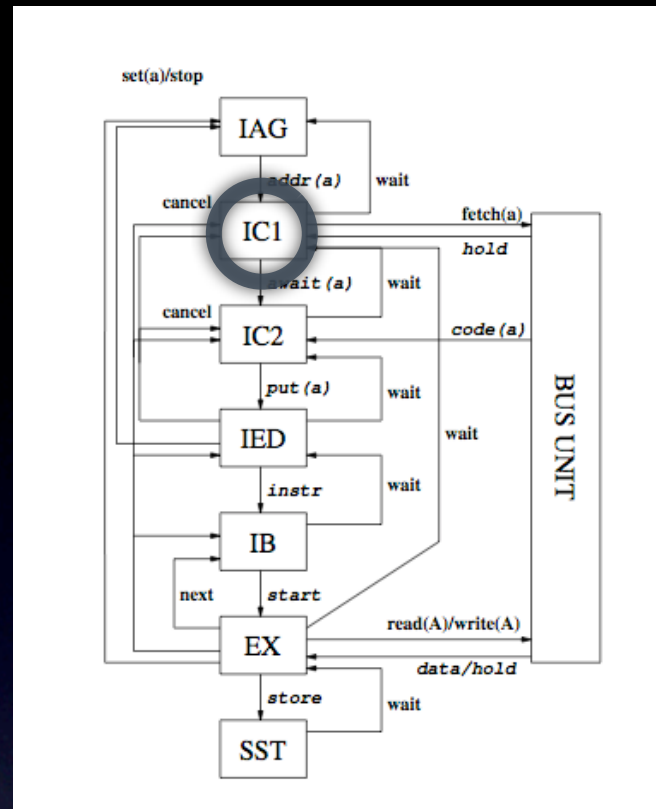
Formales Model ColdFire



Abstract Pipeline State

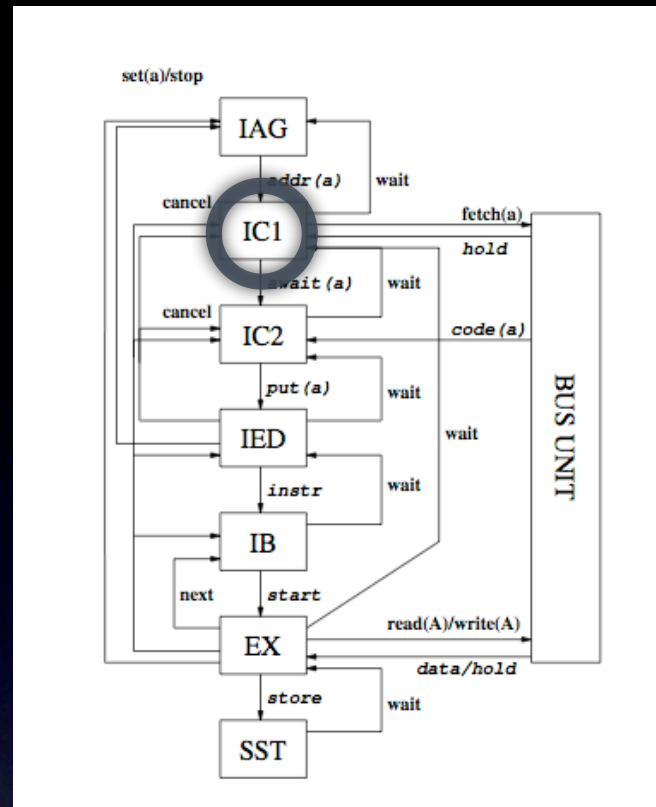
- alle möglichen konkreten Pipeline States für eine Instruction
- z.B. Abhängig von unbekannten Adressen
- Pro konkreter Pipeline State auch ein eigener Cache State

Abstract State

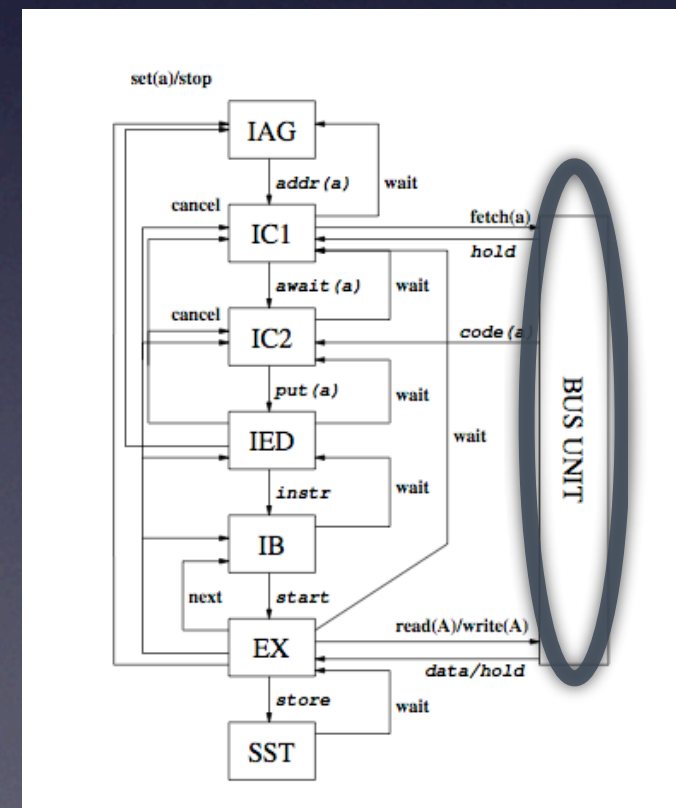
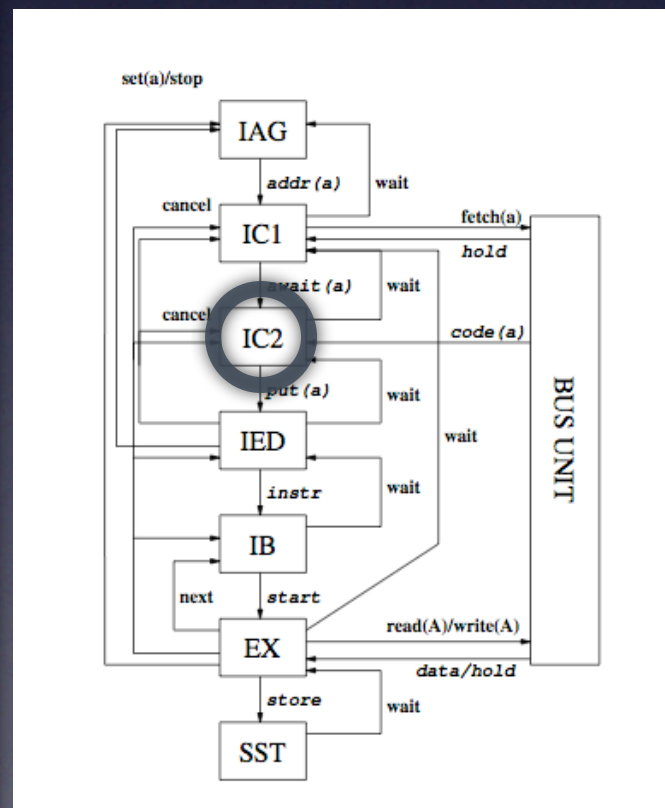


t

Abstract State



Abstract State



Zusammenfassung

- Gute Ergebnisse bei Airbus Benchmark
- Bei Toy Programmen größere Unterschiede zu gemessener Ausführungszeit

Zusammenfassung

- Gute Ergebnisse bei Airbus Benchmark
- Bei Toy Programmen größere Unterschiede zu gemessener Ausführungszeit

Vermutung: Resultiert in aiT