Timing Anomalies in Dynamically Scheduled Microprocessors

Real-Time Computing and Communications Lab.
Hanyang University

Yongsoon Lee

25th September, 2006

About a Paper

- Author: Thomas Lundqvist, Per Stenstrom
- University: Department of Computer Engineering
  Chalmers University of Technology
  SE-412 96 Goteborg, Sweden
- Conference: RTSS(Real-time systems Symposium), December 1999

Table of Contents

1. Introduction

- **Previous timing analysis methods have assumed that the worst-case instruction execution time necessarily corresponds to the worst-case behavior.**
- **We show that this assumption is wrong in dynamically scheduled processors.**
- **We propose some simple code modification techniques that eliminate the existence of timing anomalies, thus enabling known analysis methods to estimate WCET**

2. Timing Anomalies in Processors

- Consider the execution of a sequence of instructions.
- Let us study two different cases where the latency of the first instruction is modified.
- In the first case, the latency is increased by i clock cycles.
- In the second case, the latency is decreased by d cycles.
- Let C be the future change in execution time resulting from the increase or decrease of the latency.
- Definition : A timing anomaly is a situation where, in the first case, $C > i$ or $C < 0$, or in the second case, $C < -d$ or $C > 0$.
- If C is guaranteed to be in the interval : $0 \leq C \leq i$ in the first case or $-d \leq C \leq 0$ in the second case, we have no timing anomalies.
- Condition : If a processor only contains in-order resources no timing anomalies can occur.

Timing anomaly examples (1)

Cache hit can result in worst-case timing

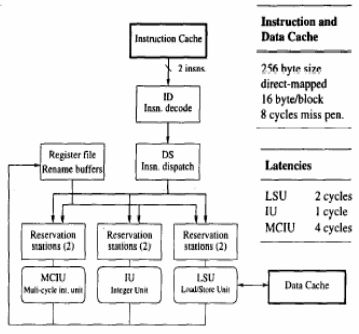
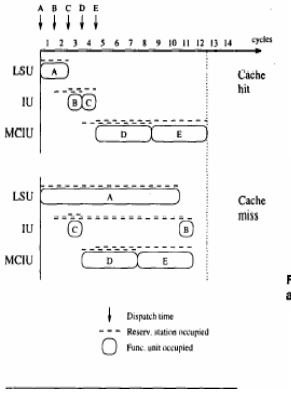


Figure 1. A simplified, yet timing-anomalous, PowerPC architecture.

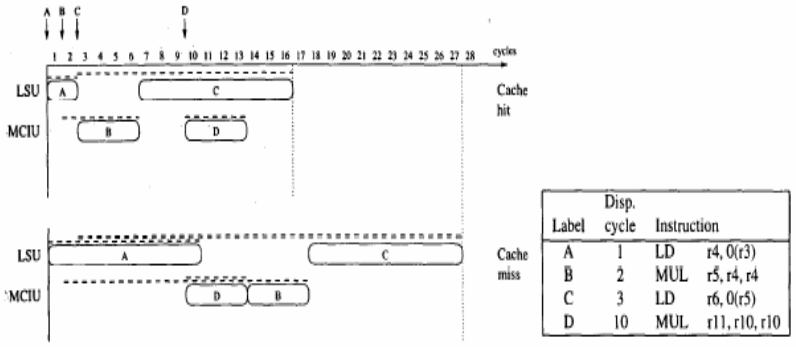


Label	Disp. cycle	Instruction	
A	1	LD	r4, 0(r3)
B	2	ADD	r5, r4, r4
C	3	ADD	r11, r10, r10
D	4	MUL	r12, r11, r11
E	5	MUL	r13, r12, r12

Figure 2. An example when a cache hit causes a longer execution time than a cache miss.

Timing anomaly examples (2)

Miss penalties can be greater than expected.

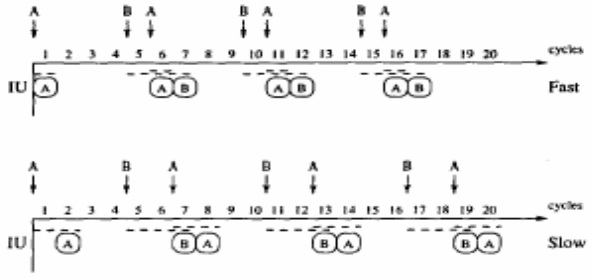


Label	Disp. cycle	Instruction	
A	1	LD	r4, 0(r3)
B	2	MUL	r5, r4, r4
C	3	LD	r6, 0(r5)
D	10	MUL	r11, r10, r10

Figure 3. An example when the cache miss penalty is higher than expected.

Timing anomaly examples (3)

Domino effect when executing loops. WCET may not be bounded.



Label	Disp. cycle	Execute cycle	Instruction	
A	$E_A + 5$	Immediate	ADD	r4, r3, r3
B	$D_A + 4$	$D_A + 6$	ADD	r11, r10, r10

Figure 4. Example of domino effects.

3. Limitations of Previous Methods

- Consider first a program containing only a single feasible path.
- The WCET is then the longest execution time of the instruction sequence along this path.
- Assume that the sequence contains n variable-latency instructions with unknown latency, but we know that each instruction can have k different latencies
- To be safe, we must examine k^n instruction schedules - infeasible
- Consider partial sequence of instructions (basic block) and find longest path. This is impossible without knowledge of the whole instruction sequence.
- All this design is for regarding variable latency instructions

4. Methods for Elimination of Anomalies (1)

- The pessimistic serial-execution method
  - This means that we model all instructions as being executed in-order in the functional units.
  - We sum all instruction latencies in the functional units. In addition to this, we add the miss penalties for all instruction and data cache misses.
  - They can not lead to a greater execution time than the one estimated for serial execution
  - The serial-execution estimate will be safe but maybe too pessimistic. However, a big advantage is that unknown events in the system are handled in a safe way.

4. Methods for Elimination of Anomalies (2)

- The program modification method
  - The serial-execution method is very pessimistic.
  - If we want a tighter estimated WCET we must model the pipelined execution accurately and deal with the problem of timing anomalies.
  - All variable-latency instructions with unknown latency and other unknown events still result in a predictable pipeline state
  - One way of accomplishing this is to modify the program so that we can rely on safe local decisions.
  - To force an **in-order resource use** when executing the variable-latency instruction. Then, the pipeline state must be predictable.
  - Unfortunately, no supports for in-order resource scheduling is present in processors today, but other instructions may be used for this purpose. Ex)sync

4. Methods for Elimination of Anomalies (3)

- Sync inhibits dispatching until the sync instruction completes. This instruction can be used as a way to force serialization together with a variable-latency instruction.
- In order resource scheduling, maximum latency will be the worst-case latency.
- For cache handling, set the state of the caches corresponding to the two paths are made equal to each other.
- No unnecessary pessimism is added due to additional cache misses.
- Then, use timing analysis methods to estimate the WCET for programs running on a dynamically scheduled processor.

4. Methods for Elimination of Anomalies (4)

Name	Description
matmult	Multiplies two 50x50 matrices
bsort	Bubblesort of 100 integers
isort	Insertsort of 10 integers
fib	Calculates n:th element of the Fibonacci sequence for $n \leq 30$
DES	Encrypts 64-bit data
jfdctint	Does a discrete cosine transform of an 8x8 pixel image
compress	Compresses 50 bytes of data (downscaled version of compress from SPEC CPU95 benchmark suite)

	Measured	Estimated WCET						
	Actual	Unsafe program		Serial method		Modified program		Modified
Program	WCET	WCET	Ratio	WCET	Ratio	WCET	Ratio	slowdown
matmult	5283287	5283287	1	10566574	2	6323287	1.20	1.20
bsort	230490	230490	1	460981	2	256854	1.11	1.11
isort	2085	2085	1	4170	2	2325	1.12	1.12
fib	797	797	1	1594	2	797	1	1
DES	186166	186358	1.001	372716	2.002	186358	1.001	1
jfdctint	9409	9409	1	18819	2	9921	1.05	1.05
compress	16486	54583	3.31	109167	6.62	69291	4.20	1.27

Table 2. The estimated WCET using the serial method and when using modified programs.

Conclusions

- Previous worst-case proposed estimate method cannot estimate correct WCET in dynamic scheduling.
- Propose to make program modifications to make unknown instruction latencies predictable.
- We applied these program modifications to seven benchmark programs.
- We found that the pessimism imposed by the program modifications is less than 27% for the programs in our benchmark.

Quiz

- Give an example that the situation occurs timing anomaly.
  - ( ) can result n worst-case timing.
  - Miss penalties can be ( ) than expected.
  - Impact on WCET may not be ( )
- Give a method for eliminating to anomalies in this paper suggested
  - The pessimistic ( ) method
  - The ( ) method