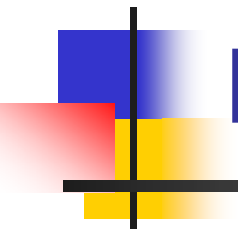
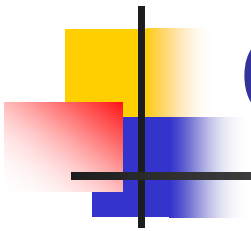
An Accurate Worst Case Execution Timing Analysis for RISC processors



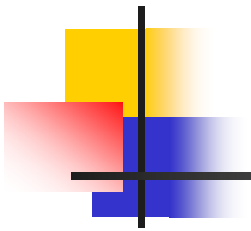
- Presented by

Manoj Sivakumar



Overview

- Introduction
- Issues in estimating WCET for RISC processors
- WCET Timing Analysis
- Preliminary Results
- Conclusion



Introduction

- Estimation of WCET (Worst Case Execution Time) is critical for Real Time Applications
- The tighter the bound on WCET the better for us – as long as all the deadlines are met
- A really pessimistic estimate would surely be a waste of time and resources

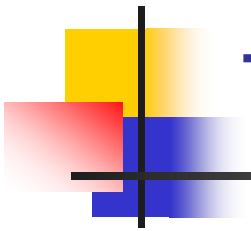
Introduction ..contd

- For RISC (Reduced Instruction Set Computer) processors estimating
  WCET becomes more complex because of issues with
  - Pipelining effects
  - Cache Hits/Misses
- The paper presents one such mechanism that computes WCET including the effect of Pipelining and Cache

Issues with RISC Processors for WCET Estimation

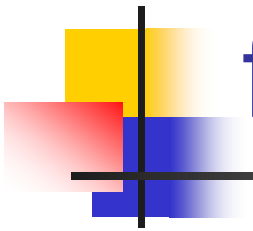
- PIPELINING
  - WCET analysis of individual basic blocks has been presented in literature
  - But effect of Pipelining across Basic blocks has not been analyzed – this is a serious impact.
  - Having a generic method for pipelining which can be applicable to any processor is necessary

Issues with RISC Processors for WCET Estimation



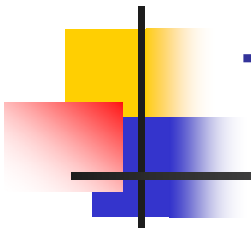
- Effect of Cache – Unpredictable behavior of Cache stems from
  - Inter-task interference
  - Intra-task interference

Issues with RISC Processors for WCET Estimation



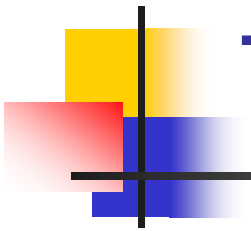
- Intra-task Interference
  - Occurs when more than one memory block of the same task compete with each other for the same cache block
  - These misses can be of 2 types
    - Capacity Misses
    - Conflict Misses

Issues with RISC Processors for WCET Estimation



- Inter-task Interference
  - This occurs when tasks can be pre-empted
  - This type of cache misses cannot be avoided unless we have cache-partitioning
  - But how effective is cache-partitioning – is very much an open issue as the number of tasks increases cache partitioning will become ineffective

Proposed Method – WCET Timing Analysis



- For incorporating effect of Pipelining
  - Because of overlapped execution a block's timing analysis will depend on the surrounding blocks – hence just calculating a simple time bound will not be effective
  - So the authors construct a set of reservation tables

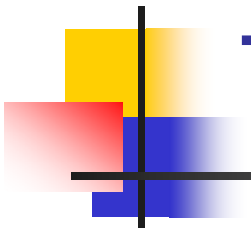
Proposed Method – WCET Timing Analysis

RESERVATION TABLE FOR A GIVEN PIECE OF CODE ON A MIPS PIPELINE

```
lw   $25, 16($23)
nop
lw   $24, 16($22)
nop
mult $25, $24
mflo $25
addu $21, $21, $25
```

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21
IF	X	X	X	X	X	X	X														
RD		X	X	X	X	X	X	X													
ALU			X	X	X	X												X	X		
MD							X	X	X	X	X	X	X	X	X	X	X		X	X	
DIV																					
MEM				X	X	X	X												X	X	
WB					X	X	X	X												X	X

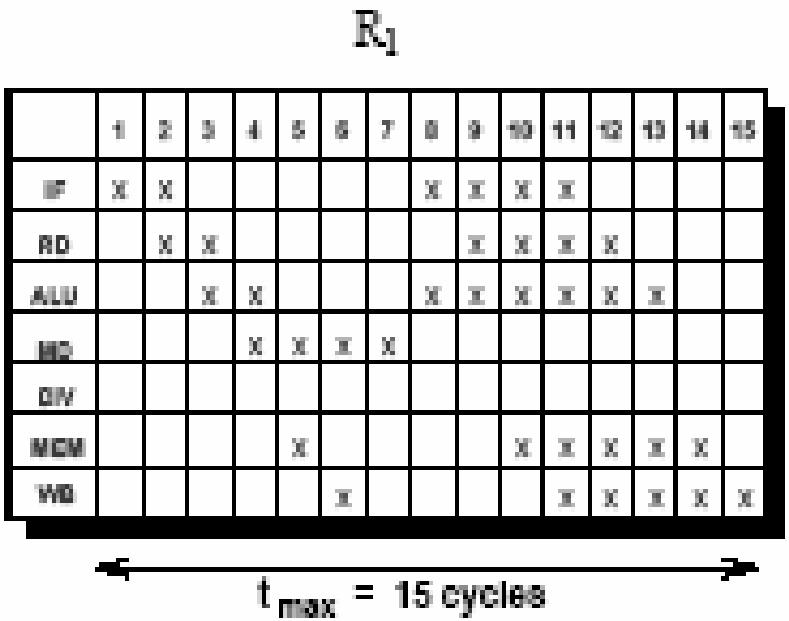
Proposed Method – WCET Timing Analysis



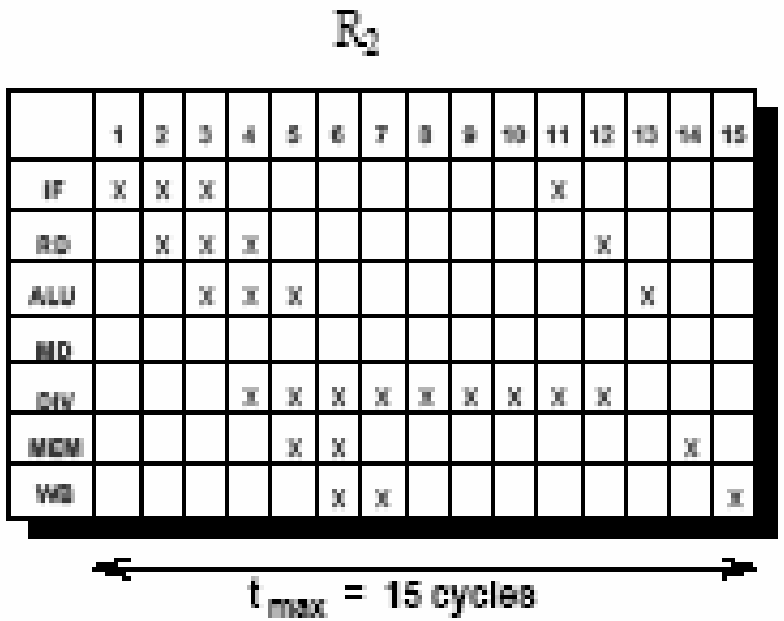
- A program with an IF statement may have more than 1 execution path
- Moreover it is difficult to predict which path corresponds to worst case execution
- Consider the following example

Proposed Method – WCET Timing Analysis

R1 – THEN path



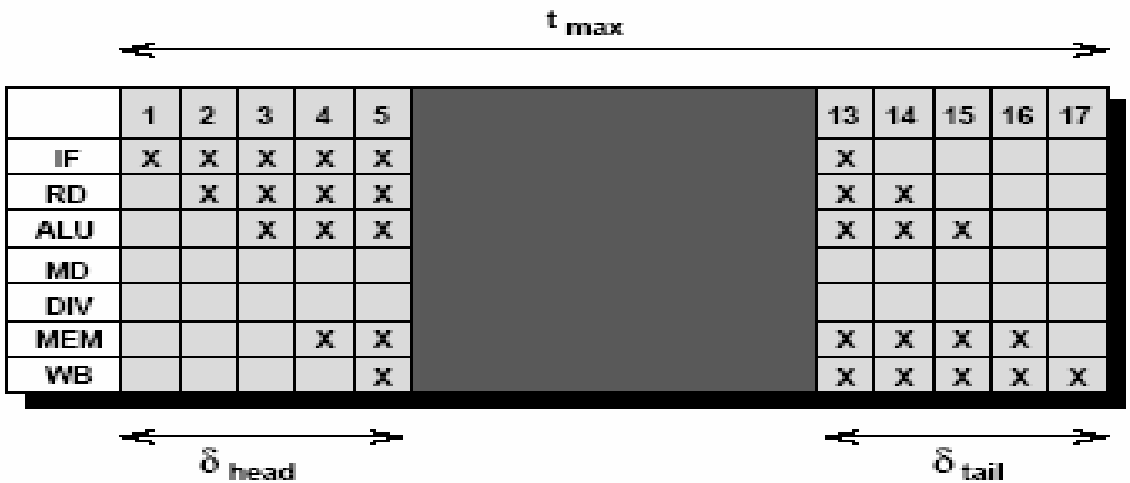
R2 – ELSE path



Proposed Method – WCET Timing Analysis

- Worst Case Timing Abstraction (WCTA)

```
struct pipeline_timing_information {
    time t_max;
    reservation_table head[δ_head];
    reservation_table tail[δ_tail];
}
```



Proposed Method – WCET Timing Analysis

- With each program we maintain a set of reservation tables and this forms the WCTA
- With this at hand consider 2 sequential blocks **S: S1;S2**
- Now the WCET of these 2 blocks together can be calculated using the formula

$$W(S) = W(S_1) \oplus W(S_2)$$

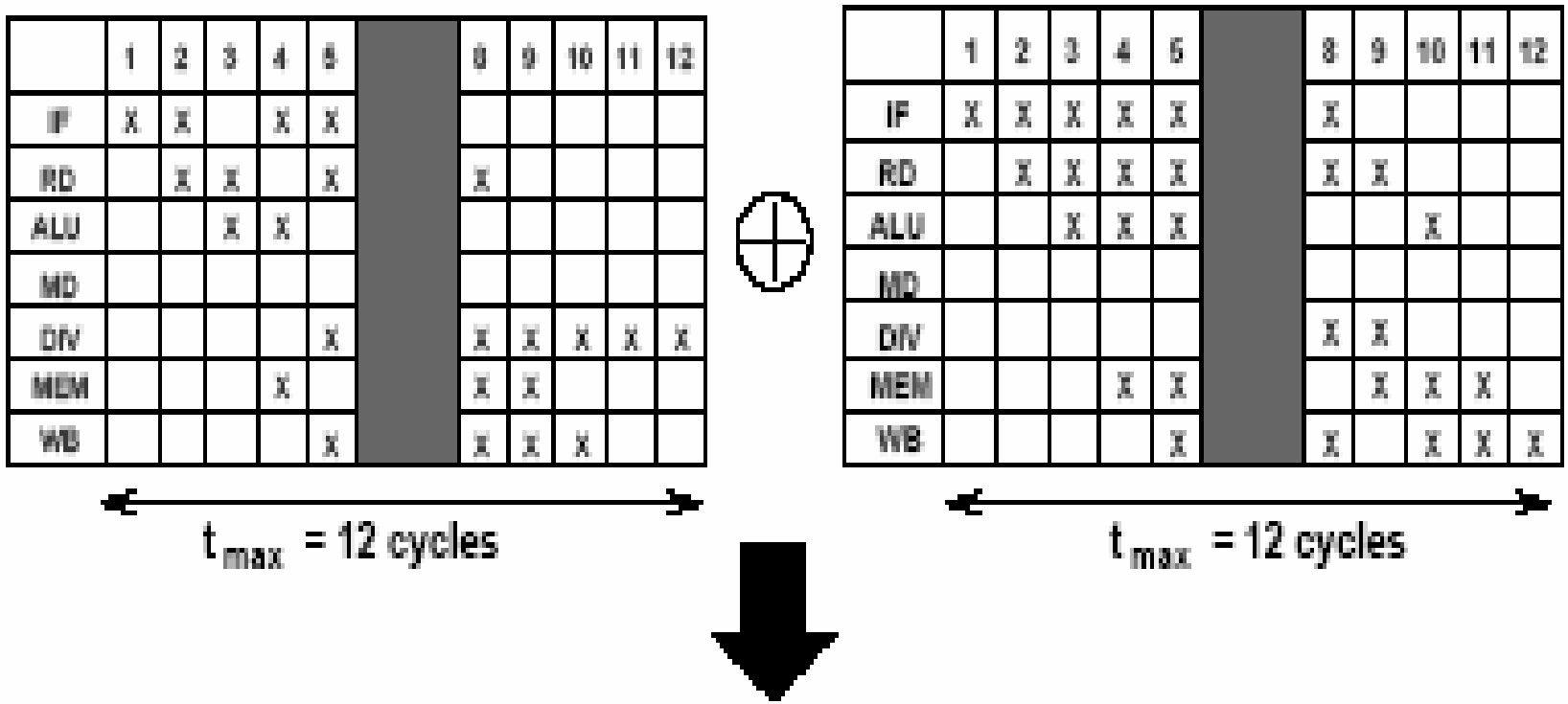
Proposed Method – WCET Timing Analysis

- Where the operation is defined as

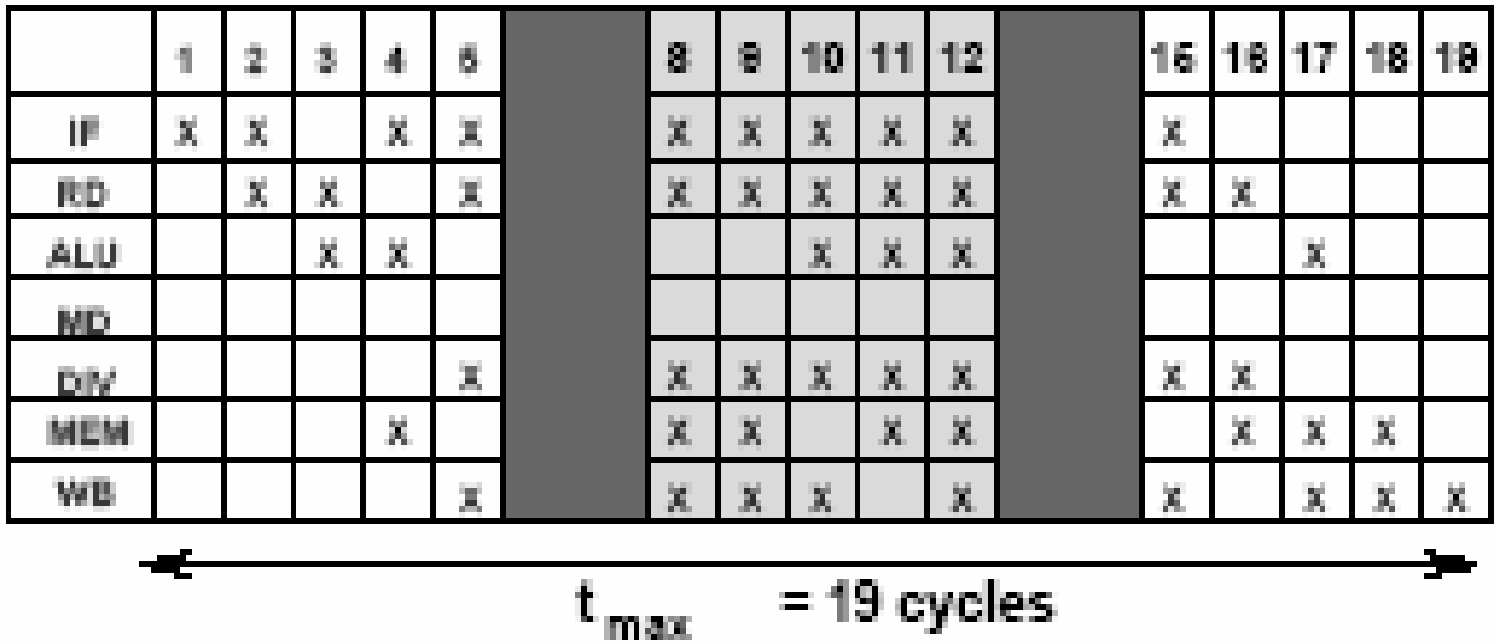
$$W_1 \bigoplus W_2 = \{w_1 \oplus w_2 | w_1 \in W_1, w_2 \in W_2\}$$

- Now instead of trying to understand what his exactly means – it is a simple concatenation with an overlapping merge – it will be clear with this example

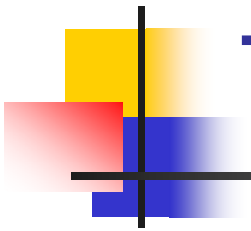
Proposed Method – WCET Timing Analysis



Proposed Method – WCET Timing Analysis

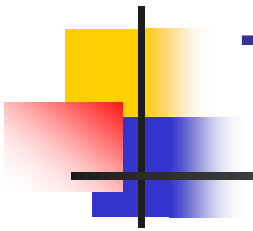


Proposed Method – WCET Timing Analysis



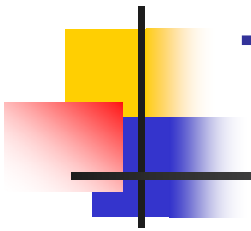
- Pruning of the WCTA
  - An element in a WCTA can be eliminated if we can guarantee that the element's WCET assuming worst case scenario on the surrounding program constructs **IS SHORTER THAN** the WCET of some other element in the same WCTA assuming best case scenario for this element on the surrounding program constructs

Proposed Method – WCET Timing Analysis



- Loop Timing Analysis
  - In the presence of loops and if conditions – the number of execution paths can grow huge and hence the set of WCTA might increase exponentially
  - The authors have shown that by using a dynamic programming approach and show that this is still feasible.

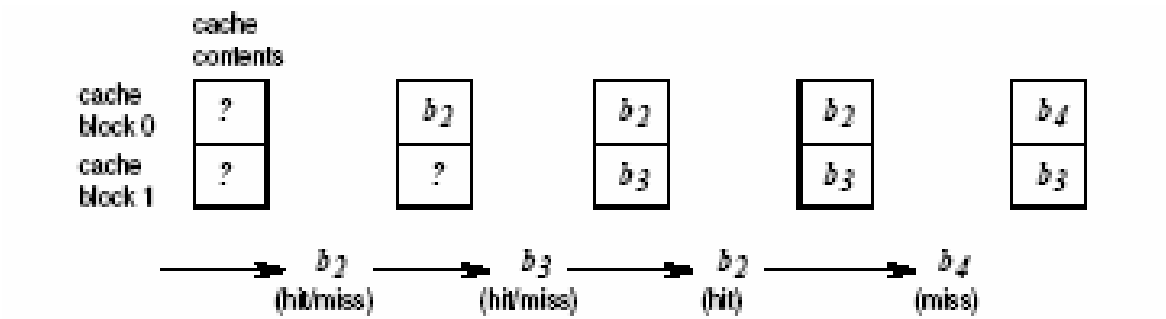
Proposed Method – WCET Timing Analysis



- IMPACT OF CACHE ON TIMING ANALYSIS
- Effects due to
  - Instruction Cache
  - Data Cache

Proposed Method – WCET Timing Analysis

- Instruction Cache
  - Execution of a block will differ depending on which execution path has been taken prior to the block – this is because of the history sensitive nature of the cache



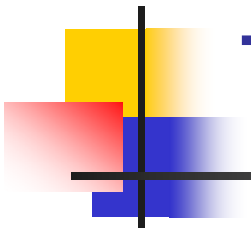
Access order of blocks – {b2, b3, b2, b4}

Proposed Method – WCET Timing Analysis

- In order to include effect of Caching in timing analysis

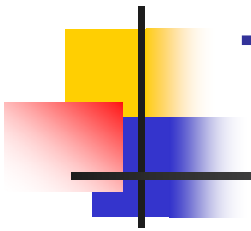
```
struct pipeline_cache_timing_information {
    time t_max;
    reservation_table head[δ_head];
    reservation_table tail[δ_tail];
    block_address first_reference[n_block];
    block_address last_reference[n_block];
};
```

Proposed Method – WCET Timing Analysis



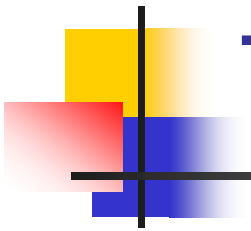
- First_Reference
  - Set of instruction block addresses of the references whose hits or misses depend upon the cache contents prior to the program
- Last_Reference
  - Set of instruction blocks that will remain in cache after this block is finished execution

Proposed Method – WCET Timing Analysis

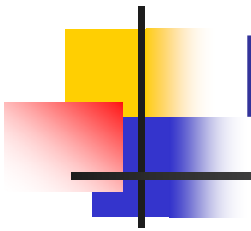


- The concatenation and pruning operations are now modified to update this first_reference and last_reference accordingly
- The paper gives the C code for the update but it is very simple and also we can see that it will surely give a tigher bound

Proposed Method – WCET Timing Analysis



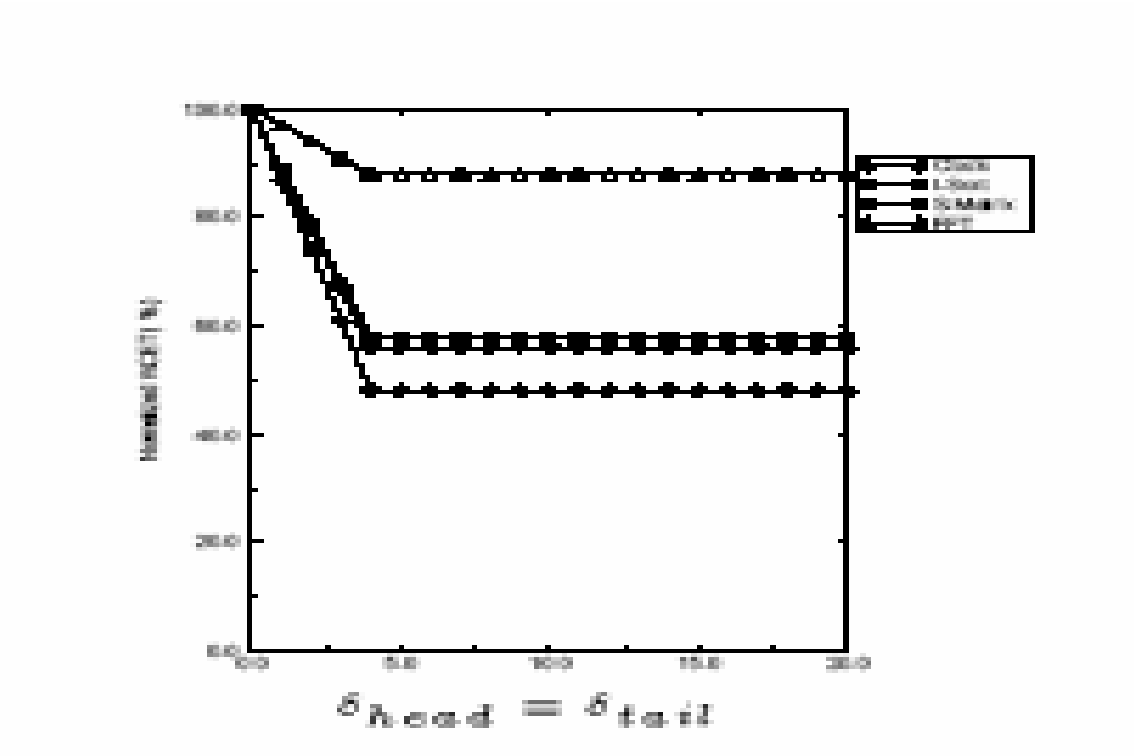
- Data Cache
  - The issue with data cache is that actual address of some data reference will be know only at run time
  - So if special hardware support to handle this is not present the authors say that it is safe to assume all references are misses



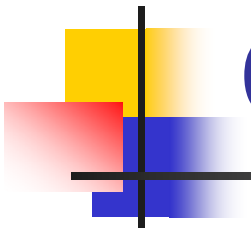
Preliminary Results

- The authors have performed experiments measuring the WCET on four benchmark programs
  - Clock
  - FFT
  - I-Sort
  - S-Matrix

Preliminary Results



$$\delta_{head} = \delta_{tail}$$



Conclusion

- The authors present a mechanism to include effects of Pipelining and Cache in timing analysis of WCET
- The method gives a tighter bound as it includes inter-task an intra-task dependence in estimation