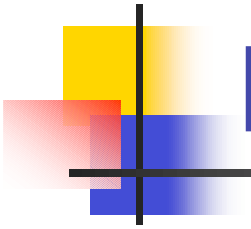
Hardware Loop Buffering

Scott DiPasquale

Khaled Elmeleegy

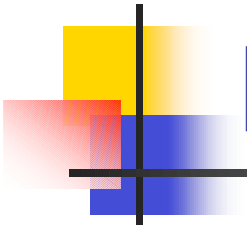
C.J. Ganier

Erik Swanson



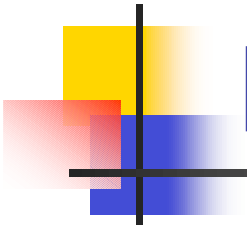
Problem and Hypothesis

- Compilers have more information about loops that they can exploit.
- Hardware loop unrolling is highly complex.
- Buffering loops reduces instruction count and improves performance with moderate complexity.



Loop Definitions

- Dynamic: The number of loop iterations is determined within the loop itself.
- Semi-dynamic: The number of loop iterations is determined prior to entering the loop; the specific value is unknown to the compiler.
- Static: Fixed iteration count for a loop; the value is known to the compiler.

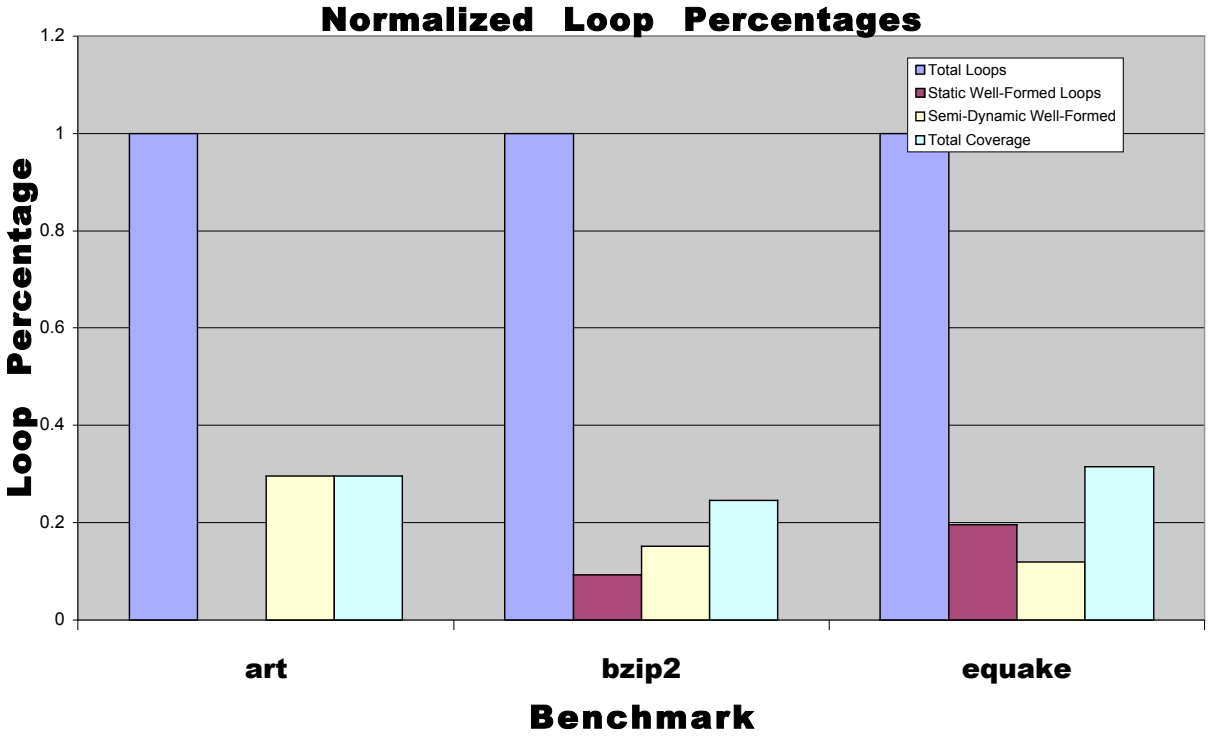


Loop Definitions

- Well-formed: There are no control transfers in the loop other than the exit branch.
- Ill-formed: Anything not well-formed.

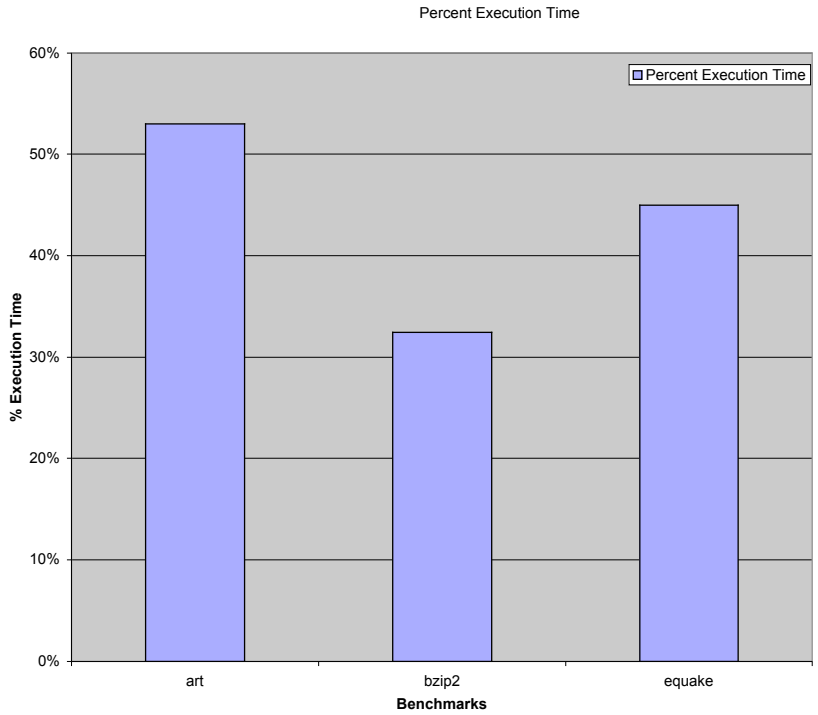
- We are concerned with static and semi-dynamic loops that are well formed.

Motivation

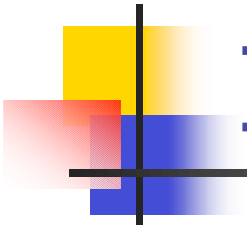


Percentage of loops in benchmark that loop buffering can affect

Motivation



Percentage of total execution time spent in relevant loops



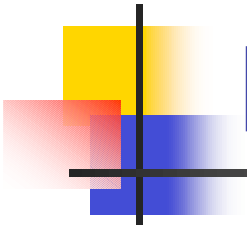
ISA changes – LOOP & ENDL

- LOOP Instruction takes immediate or register with number of iterations.
- Loop repeats without branch overhead
- ENDL Instruction signals end of loop

Example loops

```
Loop:
    SLL r3, r2, 2
    ADD r3, r3, r5
    LW r1, 0(r3)
    ADD r1, r1, r4
    SW   r1, 0(r3)
    ADDI r2, r2, 1
    SGT r5, r2, r6
    BEQ r0, r5, Loop
```

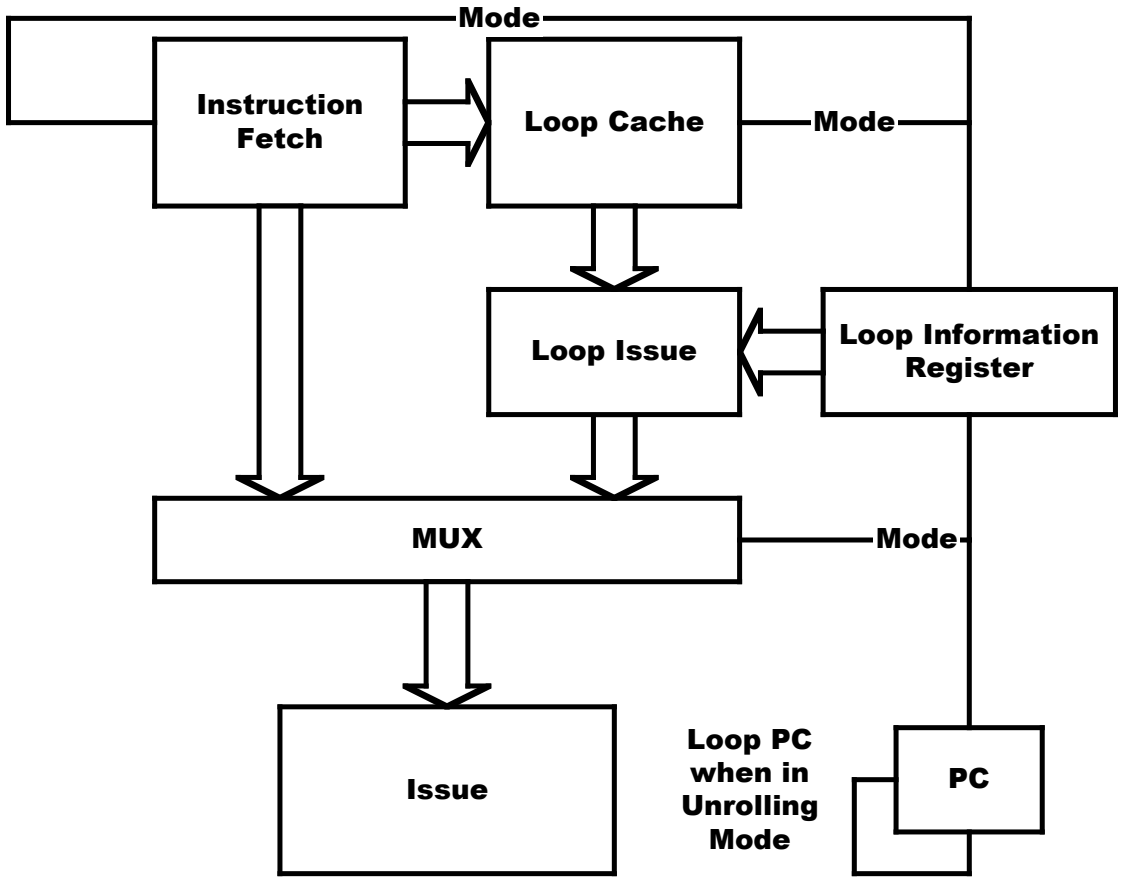
```
    LOOP r6, reg
Loop:
    SLL r3, r2, 2
    ADD r3, r3, r5
    LW r1, 0(r3)
    ADD r1, r1, r4
    SW r1, 0(r3)
    ADDI r2, r2, 4
    ENDL
```



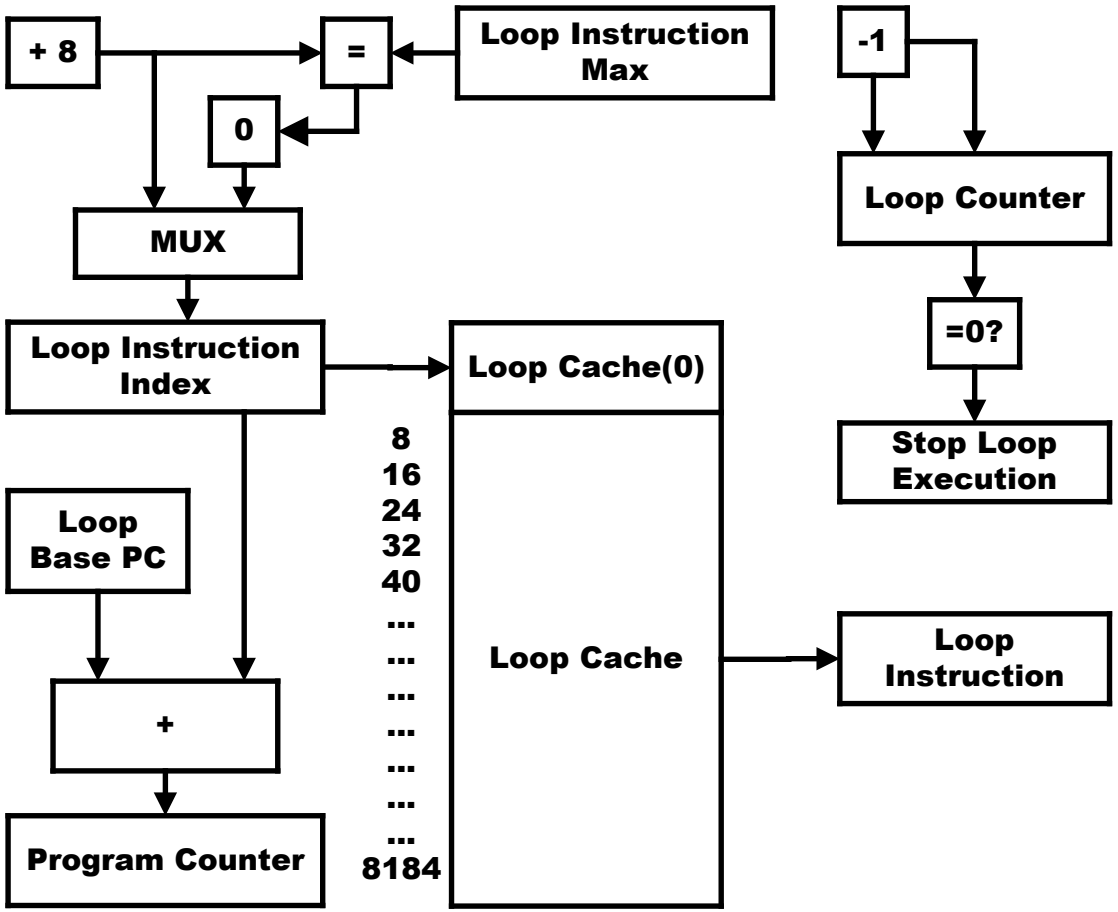
Hardware Implementation

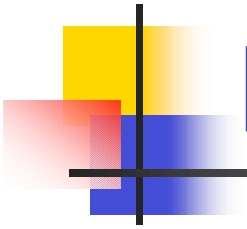
- 64b x 1024 entry buffer
- Loop counter, loop instruction index, loop instruction maximum, number of times to duplicate an instruction.
- If loop bound is in a register, maximum number of iterations is 2^32; an immediate allows 1024 iterations.

Pipeline Block Diagram



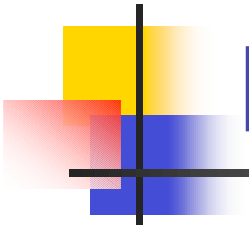
Buffer Block Diagram





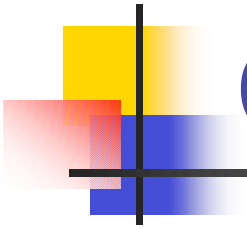
Experimentation

- Simple Scalar modified to bypass instruction fetch and to add instructions
- A small, but relevant, benchmark (183.Equake) was chosen.
- Only one benchmark was used due to the necessity of analyzing and modifying assembly by hand.



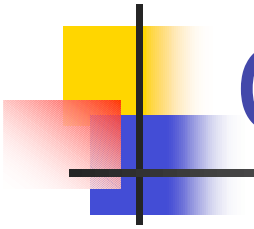
Results

- On average, individual loops in Equake saw a speedup of 1.15.
- Using Amdahl's law, this equates to an overall speedup of 1.062 for Equake.
- Other benchmarks have similar loop characteristics, and are expected to have similar speedups.



Conclusion

- Speedups achieved are appreciable.
- Hardware complexity is moderate.
- Future work can build on the framework developed here to include such elements as loop unrolling.



Questions?