Summary
Multiscalar Processors

Kiran Yellajyosula
April 30, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)
Most programs are written for sequential execution, with the assumption that instructions are executed strictly in the order they appear. Out-of-order processors suffer with increased instruction window size and issue logic. This paper tries to parallelize the execution with the help of the multiscalar paradigm and thereby improve the ILP. This is done while maintaining the sequential semantics assumed in the program.

What are the approaches attempted by this paper? (50 words)
The multiscalar processor has multiple processing units(PUs), each with its own fetching, executing and sequencing mechanism. Data is forwarded from one PU to the other in a cyclic ring fashion.
The CFG of the program is divided into tasks. The multiscalar processor uses branch prediction to speculatively execute the tasks on different PUs, while preserving inter-task data dependences through registers or memory. In case of a violation, i.e., a task uses wrong data values, the task is squashed.
The compiler breaks the CFG into tasks and creates masks which depict the inter-task dependences. The compiler directs the processor as to when to release a particular value. The processor uses synchronization to wait for register data from the previous PU. All memory dependences are tracked by means of a central Address Resolution Buffer(ARB). The ARB tracks all the memory dependences and detects violations.

What are the main conclusions of this paper? (50 words)
The authors simulated the multiscalar processor using a cycle-by-cycle simulator that is based on MIPS instructions. GCC 2.5.8 was extended to produce the code for the multiscalar processor. There is an average increase of 11 – 12% in the code size of the benchmarks tested.
The paper shows the IPC achievable using the multiscalar processor on in-order and out-of-order processors without explicit software support. The authors believe that with the help of increased software support and streamlined hardware, the multiscalar processor would be able to extract more ILP.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)
This paper improves the performance of applications by making the compiler create threads by breaking the CFG of the program. The work exploits and exploits the inherent parallelism available in sequential applications. It addresses an issue of how to utilize the extra transistors available in the chip so as to improve performance.
The paper also alleviates the design complexity of very wide issue window or a large in-flight execution window. It also shows how to look past many branch instructions. The concept of the hardware assuring the correctness of the code relaxes the work of the compiler and paves the way for some future aggressive research.

Are the proposed approaches valid? Describe its strength and weakness. (100 words)

Yes, the approach used in the paper is valid.

The strengths of the approach include:

1. improved performance
2. reduced complexity in design. The interfaces between the processors and the cache are to be designed.
3. reduction in the design complexity of a very wide issue logic
4. reduced size of the ROB and the corresponding hardware to check dependences at run time.
5. a different technique of estimating branch probabilities.

The weakness of the approach include:

1. a centralized hardware (ARB) for assuring that the memory dependences are not violated
2. Increased power consumption
3. Increase complexity in cache.
4. The size of each thread is small, corresponding to the thread generation.
5. The hardware has to issue instructions and there is no scheduling to improve the performance of threads
6. increased program size.

Do the results support the conclusions? Explain. (100 words)

Yes, the results support the conclusions. The experiments conducted on a multiscalar simulator show improvement in the performance of applications in the case of in-order and out-of-order processors. The simulated processors were either one-issue or two-issue processors which have fairly simple issue logic. The authors compare the IPC of the multiscalar to the IPC of the superscalar processor and explain the speedups and task prediction accuracies achieved in SPECint92, SPECfp92 benchmarks, GNU textutils 1.9 and cmp from the GNU difutils2.6

Describe the potential future works? (100 words)

The compiler can improve the task creation process and then reschedule these threads to reduce the number of violations. The compiler can be optimized to reduce the increase in code size and overhead while the hardware can be streamlined for communicating data. The ARB can be distributed across the processors so as to decrease the access bandwidth. Since all speculative data is stored in a separate cache, we need to design a protocol to ensure the data is forwarded to speculative tasks on demand. The tasks might be of different sizes and hence load balance will become an issue. A longer thread would delay smaller threads from retiring.