Summary
Multiscalar Processors

Venkatesan Packirisamy

February 2, 2005

1 To be completed before class

What are the problems solved by this paper? (50 words)

Current superscalar processors use hardware speculation and other techniques to extract ILP in a program. It can issue multiple instructions from the same program in the same cycle. But the parallelism that can be extracted by this method is very limited. The main reasons are:

- branch prediction accuracy limits ILP (There are 1 branch in every 5 instructions)
- complexity of issue logic to issue n instructions is n^2.
- to extract more parallelism we need a very large window of instructions to know which are ready to issue. Because of this the superscalar approach is not scalable.

What are the approaches attempted by this paper? (50 words)

The paper tries to cross these limits and extract more parallelism in the program using a novel technique. They try to get more parallelism by extracting thread level parallelism (TLP). The multiscalar processor contains many processing elements, each of which can fetch and execute instructions independently. A sequencer walks over the Control Flow Graph (CFG) and assigns tasks to these processing elements. The tasks execute in parallel (speculatively) while maintaining sequential semantics. In effect multiscalar splits the complexity of the superscalar processor and distributes across the processing element. Due to this the processor can extract parallelism from a very large window of instructions and the processor is also not as complex as the superscalar processor.

What are the main conclusions of this paper? (50 words)

The multiscalar processor has various advantages to the existing paradigms of computer architecture. The multiscalar approach can give very good speedup over a superscalar processor which uses the same amount of hardware. Apart from performance, the multiscalar processor is comparatively less complex (eg. issue logic). The paper discusses various performance bottlenecks which indicates that further speedup is possible by using more advanced hardware and compiler approaches.

2 To be completed after class

Did this paper address an important issue? Explain. (100 words)

There is a limit on the amount of parallelism that can be extracted with the superscalar architecture. Other architectures like VLIW also have lots of limitations. Studies conducted on the limits of ILP in integer programs indicate that the maximum ILP that can be extracted using superscalar processors is less than 5 (less than 3 in some studies). The major limitations of the superscalar processors are:

- branch prediction:
  In integer programs there is a branch for every 5 instructions. So if we execute an instruction across 5 branches and of the accuracy of branch prediction is 90%, then only 60% of the time
the instruction executed will be useful. So if we try to aggressively issue instructions across multiple branches, most of the time it will be wasted. So there is a limit on how much we can look ahead.

- **issue logic:**
  When we issue an instruction, we have to compare it with other instructions issued simultaneously. So if n instructions are issued the number of comparisons is \( O(n^2) \). This means if we want to issue more instructions per cycle, the complexity of the issue logic will increase thus increasing the pipeline length.

- **window size:**
  In order to issue more instructions per cycle, superscalar processors need to have a huge window of instructions from which it can select instructions with dependencies resolved. But increased window size means, increased number of cycles to lookup the window.

- **load store queue:**
  A load or store instruction with unresolved address can block the progress of the entire pipeline.

The paper proposes the multiscalar paradigm which solves all limitations of a superscalar processor.

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

In a Multiscalar processor, multiple processing elements (with reduced complexity than the original single processor) are arranged in a single die. The program is divided into tasks with the help of both software and hardware. These tasks are the distributed among the processing elements. Each processing element fetches its own instructions and execute them independently in parallel with other processing elements. The register dependences between the tasks are maintained through explicit synchronization. The memory dependences are maintained by hardware support (ARB). If there is any violation, the thread is squashed and restarted.

**Strength:**
The approach crosses all the limitations of the superscalar processor. The problem with superscalar is that it tries to extract parallelism from a single thread of instructions which are usually dependent on each other. There are portions of a program which has a different type of parallelism - thread level parallelism (TLP), eg. loops. The superscalar processor will not be able to extract this 'far away parallelism', because it can see only a small portion of the program. The Multiscalar uses software to partition the program into tasks so that in between tasks there is parallelism. If these parallel (semi-parallel) tasks are executed in parallel, we can extract the parallelism between them (inter thread) as well as parallelism within them(intra-thread). By distributing the execution, multiscalar is able to extract new parallelism which is not possible in superscalar. Also the complexity of the design is very less when compared to superscalar.

**Weakness:**
The sequencer which distributes tasks among the processing elements is a centralized structure. This limits the scalability of the design. Also the ARB - address resolution buffer is centralized.

**Do the results support the conclusions? Explain. (100 words)**
The results show good improvement in speedup in most of the benchmark programs. Some programs even show 4 times speedup. But many programs have only moderate improvement. One of the main reasons is the percentage of code where parallelization is applied could be less. So according to amdahl's law we cannot get speedup beyond a certain limit. In spite of low coverage, the programs show good improvement. This indicates the potential of the multiscalar approach. The paper identified some areas of improvement for the multiscalar. The results confirm that the reduced improvement in some of the programs is due to these bottlenecks. For example, in the benchmark sc, it is found that the loops are not well balanced.

**Describe the potential future works? (100 words)**
The paper identifies different areas which could be bottlenecks for the multiscalar approach. All these bottlenecks gives a lot of potential for future work.
• Reducing violations:
  If there are frequent memory dependences between the threads, it will lead to frequent squashes. To avoid this, the compiler can identify the dependences that will always happen (global variables and other scalar variables), and insert explicit synchronizations. But this can serialize the execution. So they must be used carefully. Advanced compiler and hardware techniques can be developed to identify the suitable synchronization point.

• Reducing dependences:
  Inter-thread dependences can lead to lot of wait time. So in addition to synchronization, we can schedule the code such that the amount of overlap is increased. So there is lot of scope for advanced compiler optimizations.

• Load balancing:
  If the work among the tasks are not well distributed, then some processing elements will be very busy will others will remain idle. The results also show that this is an important issue. So we need advanced compiler optimizations to divide the program into equal 'sized' tasks.

• Better design:
  The hardware design of the multiscalar can be further improved. The centralized structures like the sequencer and the ARB should be replaced with distributed structures so that the design is more scalable.