Traditional architectural approaches for increasing microprocessor performance rely on the use of large, complex, highly-speculative out-of-order cores to extract Instruction-Level Parallelism (ILP) from single-threaded applications. In order to realize high performance, these designs employ a myriad of speculative techniques, ranging from branch prediction to load-latency prediction and memory-dependency prediction. While these techniques are absolutely essential for realizing high performance in single-threaded machines, it is conceivable that they could be less important in processors that exploit Thread-Level Parallelism.

The goal of this thesis is to investigate the impact of disabling or limiting the branch and load-hit speculation, as well as the impact of pipelining the scheduling logic on the performance of SMT processors. We begin by examining the synergy of speculative execution with multithreading. If sufficient TLP exists, then disabling the speculative execution on SMT can, at least in theory, result in the allocation of resources to only non-speculative instructions thus possibly even increasing the performance. We quantify the impact of completely disallowing speculative execution as well as only resorting to speculative execution when no non-speculative instruction from any thread is available for fetching.

We then study and quantify the impact of disabling load-hit speculation on SMT machines. The motivation is that the bubbles created as a result of such restriction are likely to be filled by the load-independent instructions from other threads on SMT, thus possibly mitigating the performance impact.

Lastly, the impact of pipelining the instruction scheduling logic into separate wakeup and selection stages is quantified. Just as with load-hit speculation, if the pipeline bubbles are filled with the instructions from other threads, then the impact on the performance can be negligible. Various instruction selection schemes specifically targeted towards this goal are proposed and analyzed.