This paper presents Twill, a compiler framework that automatically derives optimal software pipelining (SWP) and warp specialization (WS) strategies for modern Tensor Core GPUs. Recent NVIDIA architectures such as Hopper and Blackwell expose powerful fixed-function units (Tensor Cores, Tensor Memory Accelerator, Tensor Memory), but fully utilizing them requires intricate scheduling decisions that are currently made using weak heuristics or expert intuition. In practice, this has led to long delays between hardware releases and performant kernels.

The authors observe that software pipelining and warp specialization are tightly coupled on modern GPUs and cannot be optimized independently. While SWP can be formulated as a classic modulo scheduling problem to maximize throughput, many resulting schedules are unrealizable on Tensor Core GPUs without WS due to cooperative warp execution, register pressure, blocking synchronization, and highly variable memory latencies. Treating SWP and WS as separate compilation phases often produces suboptimal or infeasible code.

To address this, the paper introduces a joint constraint-based formulation that simultaneously determines an optimal modulo schedule and a warp assignment. The approach first computes a throughput optimal initiation interval using modulo scheduling, then encodes realizability constraints which includes functional unit capacities, memory lifetimes, register limits, cross-warp communication costs, and blocking synchronization, into an SMT (Satisfiability Modulo Theories) problem. Solving this system yields a globally optimal SWP schedule and WS strategy, or proves that no such schedule exists.

The authors implement this approach in Twill, a heuristic-free system that operates on tile-level dependence graphs extracted from Triton programs. Twill is easily extensible to new GPU architectures by changing machine constraints and guarantees optimality for singly-nested loops without control flow. The paper introduces additional techniques such as cost normalization to make constraint solving tractable and specialized handling of variable-latency operations.

Evaluation on Flash Attention forward and backward passes shows that Twill automatically rediscovers the expert-designed pipelines and warp specializations used in Flash Attention 3 (Hopper) and Flash Attention 4 (Blackwell). The generated schedules achieve performance within 1–2% of hand-tuned implementations such as cuDNN and official Flash Attention kernels. The results also demonstrate that approaches which optimize SWP or WS in isolation fail to reach peak performance.

Overall, this work provides the first principled, architecture-portable framework for jointly optimizing software pipelining and warp specialization on Tensor Core GPUs, replacing ad-hoc heuristics with a sound optimization model and offering strong optimality guarantees.