Microprocessor Microarchitecture
Multithreading

Lynn Choi

School of Electrical Engineering

Limitations of Superscalar Processors

- ***Hardware complexity of wide-issue processors***
  - Limited instruction fetch bandwidth
    - Taken branches and branch prediction throughput
  - Quadratic (or more) increase in hardware complexity in
    - Renaming logic
    - Wakeup and selection logic
    - Bypass logic
    - Register file access time

- ***On-chip wire delays prevent centralized shared resources***
  - End-to-end on-chip wire delay grows rapidly from 2-3 clock cycles in 0.25μ to 20 clock cycles in sub 0.1μ technology
    - **This prevents centralized shared resources**

- ***Limitations of available ILP***
  - Even with aggressive wide-issue implementations
    - The amount of ILP exploitable is less than 5~6 instructions per cycle

Today's Microprocessor

- **CPU 2013 – looking back to year 2001 according to Moore's law**
  - 256X increase in terms of transistors
  - 256X performance improvement, however,
    - Wider issue rate increases the clock cycle time
    - Limited amount of ILP in applications
- ***Diminishing return in terms of***
  - Performance and resource utilization

Intel Corp. All rights reserved

- **Intel i7 Processor**
  - Technology
    - 32nm process, 130W, 239 mm^2 die, 1.17B transistors
    - 3.46 GHz, 64-bit 6-core 12-thread processor
    - 159 Ispec, 103 Fspec on SPEC CPU 2006 (296MHz UltraSparc II processor as a reference machine)
    - 14-stage 4-issue out-of-order (OOO) pipeline optimized for multicore and low power consumption
    - 64bit Intel architecture (x86-64)
    - 256KB L2 cache/core, 12MB L3 Caches
- ***Goals***
  - Scalable performance and more efficient resource utilization

Approaches

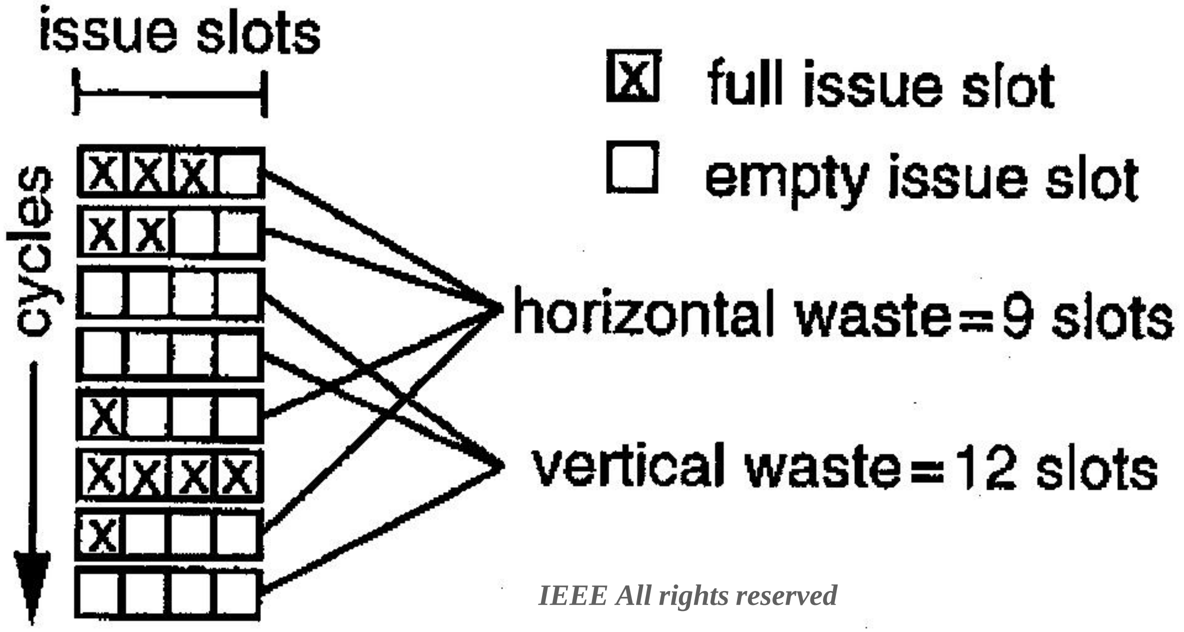
- ***MP (Multiprocessor) approach***
  - Decentralize all resources
  - Multiprocessing on a single chip
    - Communicate through shared-memory: Stanford Hydra
    - Communicate through messages: MIT RAW
- ***MT (Multithreaded) approach***
  - More tightly coupled than MP
  - Dependent threads vs. independent threads
    - Dependent threads require HW for inter-thread synchronization and communication
      - Examples: Multiscalar (U of Wisconsin), Superthreading (U of Minnesota), DMT, Trace Processor
    - Independent threads: Fine-grain multithreading, SMT
  - Centralized vs. decentralized architectures
    - Decentralized multithreaded architectures
      - Each thread has a separate pipeline
      - Multiscalar, Superthreading
    - Centralized multithreaded architectures
      - Share pipelines among multiple threads
      - TERA, SMT (throughput-oriented), Trace Processor, DMT (performance-oriented)

MT Approach

- ***Multithreading of Independent Threads***
  - No inter-thread dependency checking and no inter-thread communication
  - Threads can be generated from
    - A single program (parallelizing compiler)
    - Multiple programs (multiprogramming workloads)
  - Fine-grain Multithreading
    - Only a single thread active at a time
    - Switch thread on a long latency operation (cache miss, stall)
      - MIT April, Elementary Multithreading (Japan)
    - Switch thread every cycle – TERA, HEP
  - Simultaneous Multithreading (SMT)
    - Multiple threads active at a time
    - Issue from multiple threads each cycle
- ***Multithreading of Dependent Threads***
  - Not adopted by commercial processors due to complexity and only marginal performance gain

SMT (Simultaneous Multithreading)

- ***Motivation***
  - Existing multiple-issue superscalar architectures do not utilize resources efficiently
    - Intel Pentium III, DEC Alpha 21264, PowerPC, MIPS R10000
  - Exhibit *horizontal* and *vertical pipeline wastes*



IEEE All rights reserved

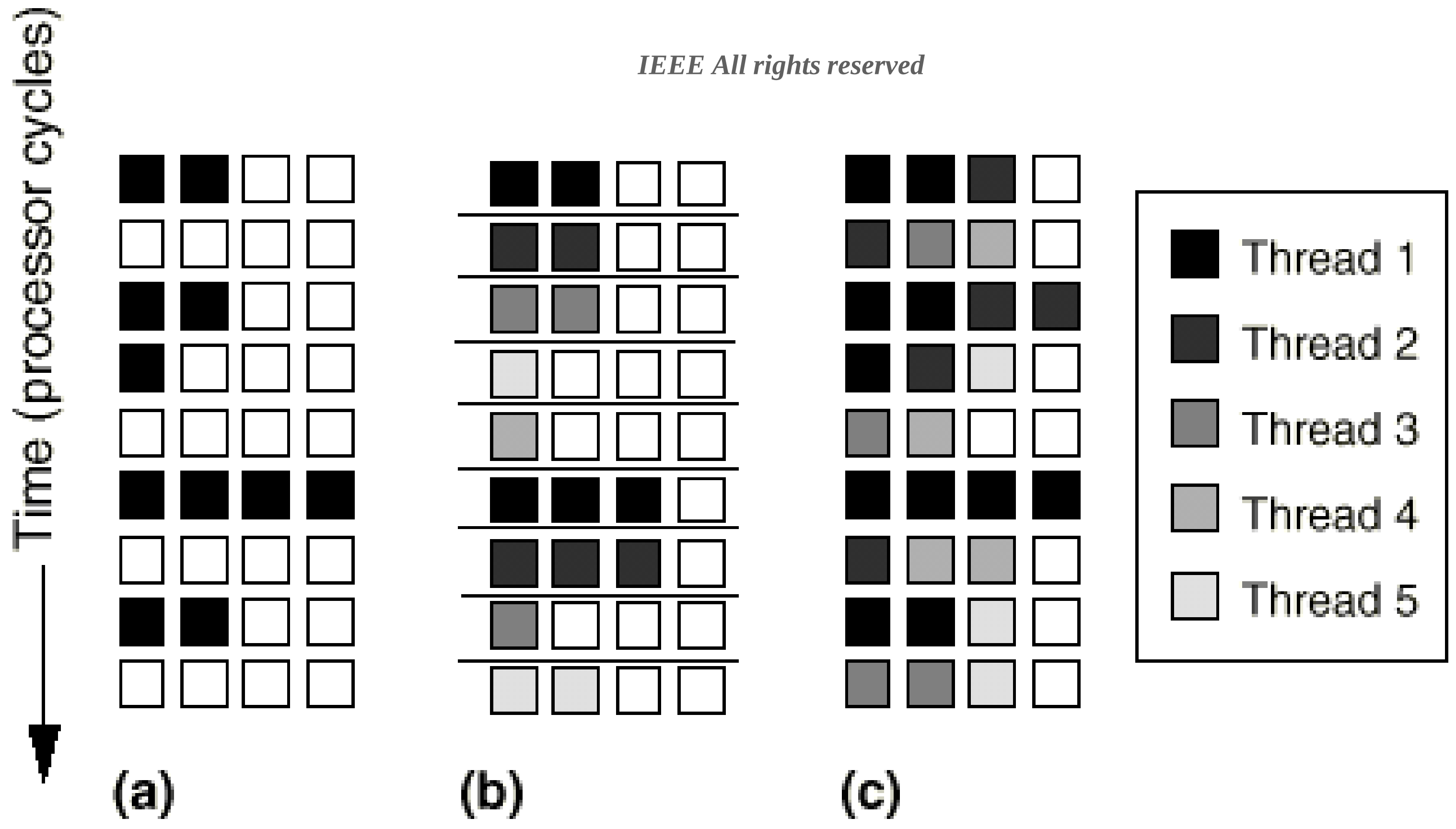
SMT Motivation

- ***Fine-grain Multithreading***
  - HEP, Tera, MASA, MIT Alewife
  - Fast context switching among multiple independent threads
    - Switch threads on cache miss stalls – Alewife
    - Switch threads on every cycle – Tera, HEP
  - Target vertical wastes only
    - At any cycle, issue instructions from only a single thread
- ***Single-chip MP***
  - Coarse-grain parallelism among independent threads in a different processor
  - Also exhibit both vertical and horizontal wastes in each individual processor pipeline

SMT Idea

- ***Idea***
  - Interleave multiple independent threads into the pipeline every cycle
  - Eliminate both horizontal and vertical pipeline bubbles
  - Increase processor utilization
  - Require added hardware resources
    - Each thread needs its own PC, register file, instruction retirement & exception mechanism
      - How about branch predictors? - RSB, BTB, BPT
    - Multithreaded scheduling of instruction fetch and issue
    - More complex and larger shared cache structures (I/D caches)
  - Share functional units and instruction windows
    - How about instruction pipeline?
  - Can be applied to MP architectures

Multithreading of Independent Threads



Superscalar

Fine-grained Multithreading

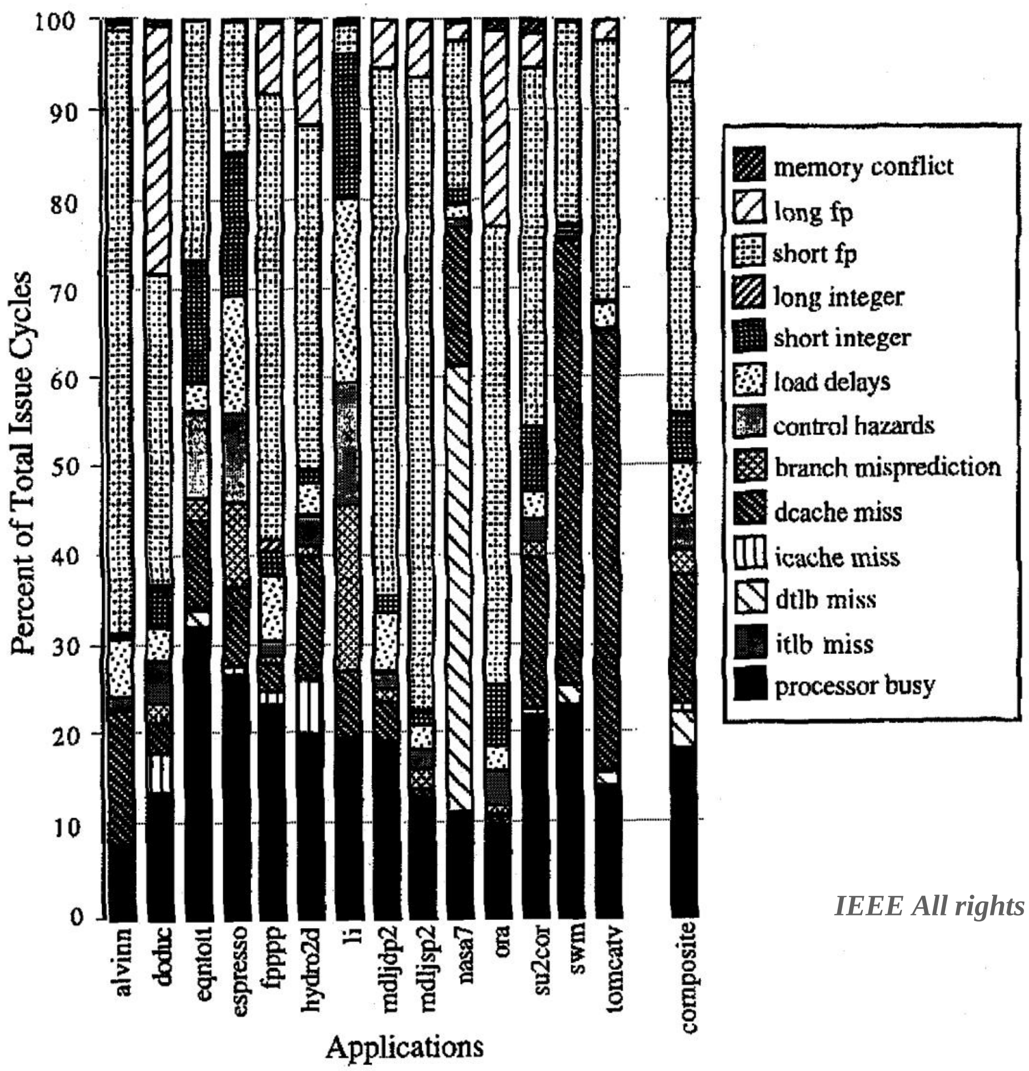
Simultaneous Multithreading

Comparison of pipeline issue slots in three different architectures

Experimentation

- ***Simulation***
  - Based on Alpha 21164 with following differences
    - Augmented for wider-issue superscalar and SMT
    - Larger on-chip L1 and L2 caches
    - Multiple hardware contexts for SMT
    - 2K-entry bimodal predictor, 12-entry RSB
  - SPEC92 benchmarks
    - Compiled by Multiflow trace scheduling compiler
  - No extra pipeline stage for SMT
    - Less than 5% impact
      - Due to the increased (1 extra cycle) misprediction penalty
  - SMT scheduling
    - Context 0 can schedule onto any unit; context 1 can schedule on to any unit unutilized by context 0, etc.

Where the wastes come from?

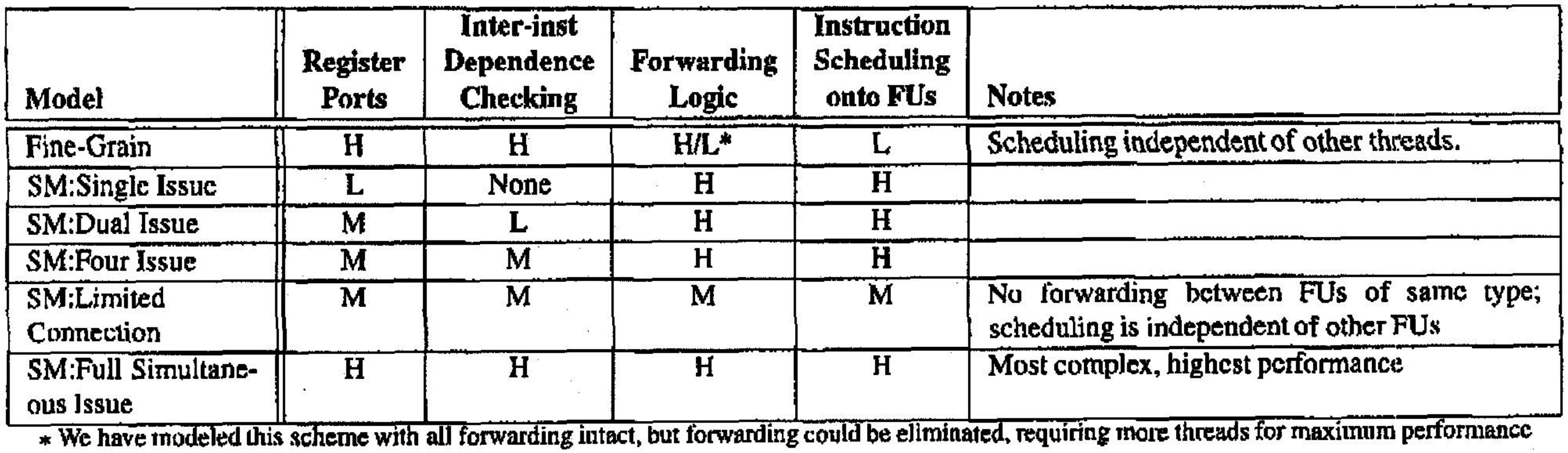


8-issue superscalar processor execution time distribution

- *19% busy time* (~ 1.5 IPC)
(1) 37% short FP dependences
(2) Dcache misses
(3) Long FP dependences
(4) Load delays
(5) Short integer dependences
(6) DTLB misses
(7) Branch misprediction
- *1+2+3 occupies 60%*
- *61% wasted cycles are vertical*
- *39% are horizontal*

Machine Models

- ❑ ***Fine-grain multithreading - one thread each cycle***
- ❑ ***SMT - multiple threads each cycle***
  - – full simultaneous issue - each thread can issue up to 8 each cycle
  - – four issue - each thread can issue up to 4 each cycle
  - – dual issue - each thread can issue up to 2 each cycle
  - – single issue - each thread issue 1 each cycle
  - – limited connection - partition FUs to threads
    - 8 threads, 4 INT, each INT can receive from 2 threads

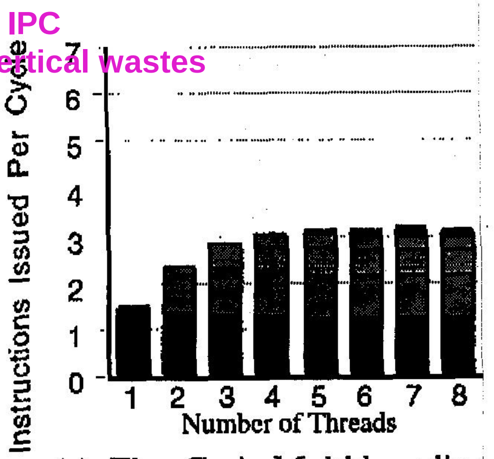


Model	Register Ports	Inter-inst Dependence Checking	Forwarding Logic	Instruction Scheduling onto FUs	Notes
Fine-Grain	H	H	H/L*	L	Scheduling independent of other threads.
SM:Single Issue	L	None	H	H	
SM:Dual Issue	M	L	H	H	
SM:Four Issue	M	M	H	H	
SM:Limited Connection	M	M	M	M	No forwarding between FUs of same type; scheduling is independent of other FUs
SM:Full Simultaneous Issue	H	H	H	H	Most complex, highest performance

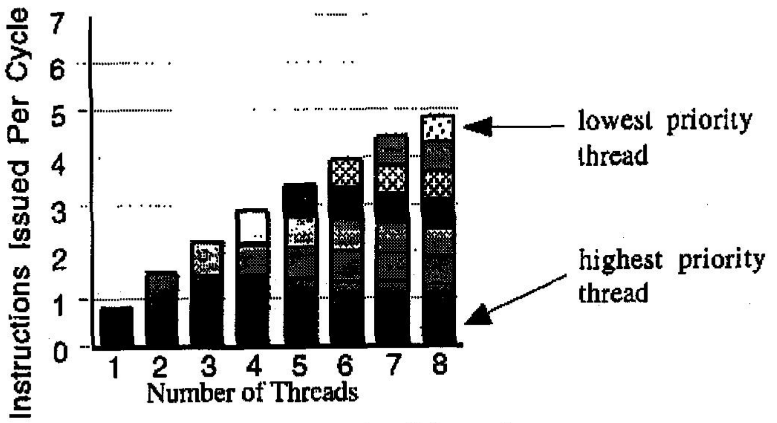
* We have modeled this scheme with all forwarding intact, but forwarding could be eliminated, requiring more threads for maximum performance

Performance

Saturated at 3 IPC
bounded by vertical wastes

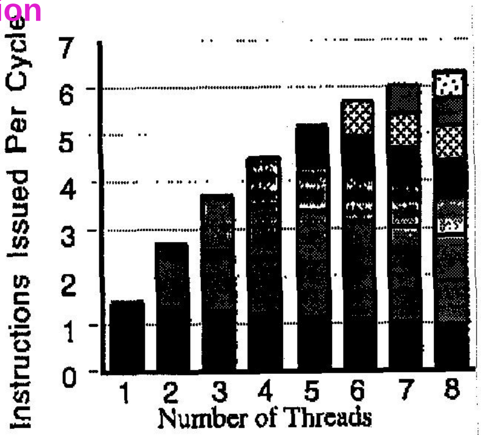


(a) Fine Grain Multithreading

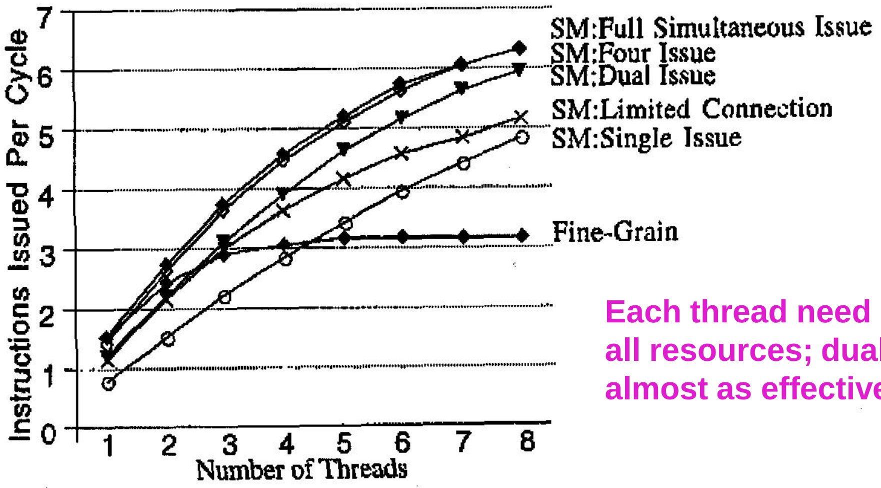


(b) SM: Single Issue Per Thread

Sharing degrades performance:
35%slow down of 1st priority thread
due to competition



(c) SM: Full Simultaneous Issue

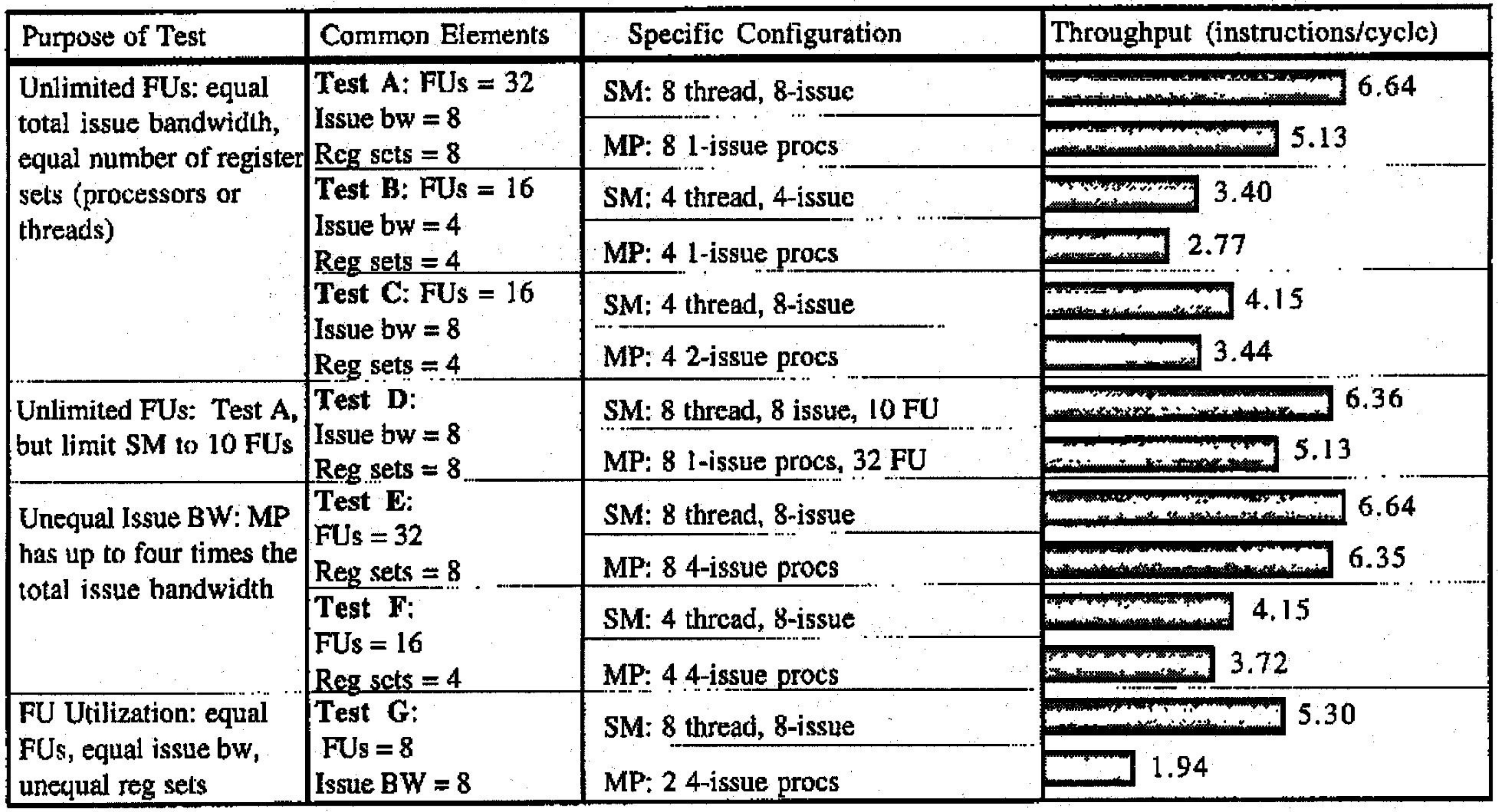


(d) All Models

Each thread need not utilize
all resources; dual issue is
almost as effective as full issue

SMT vs. MP

MP's advantage: simple scheduling, faster private cache access - both are not modeled



Purpose of Test	Common Elements	Specific Configuration	Throughput (instructions/cycle)
Unlimited FUs: equal total issue bandwidth, equal number of register sets (processors or threads)	Test A: FUs = 32 Issue bw = 8 Reg sets = 8	SM: 8 thread, 8-issue	6.64
		MP: 8 1-issue procs	5.13
	Test B: FUs = 16 Issue bw = 4 Reg sets = 4	SM: 4 thread, 4-issue	3.40
		MP: 4 1-issue procs	2.77
	Test C: FUs = 16 Issue bw = 8 Reg sets = 4	SM: 4 thread, 8-issue	4.15
		MP: 4 2-issue procs	3.44
Unlimited FUs: Test A, but limit SM to 10 FUs	Test D: Issue bw = 8 Reg sets = 8	SM: 8 thread, 8 issue, 10 FU	6.36
		MP: 8 1-issue procs, 32 FU	5.13
Unequal Issue BW: MP has up to four times the total issue bandwidth	Test E: FUs = 32 Reg sets = 8	SM: 8 thread, 8-issue	6.64
		MP: 8 4-issue procs	6.35
	Test F: FUs = 16 Reg sets = 4	SM: 4 thread, 8-issue	4.15
		MP: 4 4-issue procs	3.72
FU Utilization: equal FUs, equal issue bw, unequal reg sets	Test G: FUs = 8 Issue BW = 8	SM: 8 thread, 8-issue	5.30
		MP: 2 4-issue procs	1.94

Figure 5: **Results for the various multiprocessor vs. simultaneous multithreading comparisons. The multiprocessor always has one functional unit of each type per processor. In most cases the SM processor has the same total number of each FU type as the MP.**

Exercises and Discussion

- ***Compare SMT versus MP on a single chip in terms of cost/performance and machine scalability.***
- ***Discuss the bottleneck in each stage of a OOO superscalar pipeline.***
- ***What is the additional hardware/complexity required for SMT implementation?***