Concurrent Processors

- Vector processors
- SIMD and small clustered MIMD
- Multiple instruction issue machines
  - Superscalar (run time schedule)
  - VLIW (compile time schedule)
  - EPIC
  - Hybrids
Speedup

- let \( T_1 \) be the program execution time for a non-concurrent (pipelined) processor
  - use the best algorithm
- let \( T_p \) be the execution time for a concurrent processor
  (with \( ilp = p \))
  - use the best algorithm
- the speedup, \( Sp = T_1/T_p \) \( Sp \leq p \)

SIMD processors

- Used as a metaphor for sub word parallelism
- MMX (Intel), PA Max (HP), VIS (Sun)
  - arithmetic operations on partitioned 64b operands
  - arithmetic can be modulo or saturated (signed or unsigned)
  - data can be 8b, 16b, 32b
  - MMX provides integer ops using FP registers
- Developed in '95-96 to support MPEG1 decode
- 3DNow and VIS use FP operations (short 32b) for graphics
SIMD processors

- More recent processors target H.263 encode and improved graphics support
- More robust SIMD and vector processors together with attached or support processors
  - examples form “Green Book” Intel AMP; Motorola Altivec; Philips TM 1000 (a 5 way VLIW)
  - to support MPEG2 decode, AC3 audio, better 3D graphics
  - see also TI videoconferencing chip (3 way MIMD cluster) in Green Book.

Vector Processors

- Large scale VP now still limited to servers IBM and Silicon Graphics- Cray usually in conjunction with small scale MIMD (up to 16 way)
- Mature compiler technology; achievable speedup for scientific applications maybe 2x
- More client processors moving to VP, but with fewer Vector registers and oriented to smaller operands
Vector Processor Architecture

- Vector units include vector registers
  - typically 8 regs x 64 words x 64 bits
- Vector instructions

```
    VOP   VR3   VR2   VR1

VR3 <-- VR2 VOP VR1 for all words in the VR
```

Vector Processor Operations

- All register based except VLD and VST
- VADD, VMPY, VDIV, etc. FP and integer
  - sometimes reciprocal operation instead of division
- VCOMP compares 2 VRs and creates a scalar (64b) result
- VACC (accumulate) and other ops are possible
  - gather/scatter (expand/compress)
Vector Processor Storage

Vector Function Pipeline

VADD VR3,VR2,VR1
VP Concurrency

- VLDs, VSTs and VOP can be performed concurrently. A single long vector VOP needs two VLDs and a VST to support uninterrupted VOP processing.
  - Sp = 4 (max)

- A VOP can be chained to another VOP if memory ports allow
  - Need another VLD and maybe another VST
  - Sp = 6 or 7 (max)

- Need to support 3-5 memory access each $\Delta t$.
  - $T_c = 10 \Delta t \Rightarrow$ must support 30+ memory accesses

Vector Processor Organization
Vector Processor Summary

- **Advantages**
  - Code density
  - Path length
  - Regular data structures
  - Single-Instruction loops

- **Disadvantages**
  - Non-vectorizable code
  - Additional costs
    - vector registers
    - memory
  - Limited speedup

Vector Memory Mapping

- **Stride**, \( \Delta \), is the distance between successive vector memory accesses.
- **If** \( \Delta \) and \( m \) are not relatively prime (rp), we significantly lower \( BW_{ach} \)
- **Techniques**:
  - hashing
  - interleaving
    - with \( m = 2^k + 1 \)
    - or \( m = 2^k - 1 \)
**Vector Memory Modeling**

- Can use vector request buffer to bypass waiting requests.
  - VOPs are tolerant of delay

- Suppose
  - \( s \) = no of request sources
  - \( n \) is total number of requests per \( T_c \)
  - then \( n = (T_c / \Delta t)s \)

- Suppose \( \gamma \) is mean no of bypassed items per source, then by using a large buffer (TBF) where \( \gamma < TBF/s \) we can improve \( BW_{ach} \)

---

**\( \gamma \)-Binomial Model**

- A mixed queue model. Each buffered item acts as a new request each cycle in addition to \( n \)
  - new request rate is \( n + n\gamma = n(1+\gamma) \)

- \( B(m,n,\gamma) = m + n(1+\gamma) - 1/2 - \sqrt{(m+n(1+\gamma)-1/2)^2 - 2nm(1+\gamma)} \)
Finding $\gamma_{opt}$

- We can make $\gamma$ and TBF large enough we ought to be able to bypass waiting requests and have $BW_{offered} = n/Tc = BW_{ach}$
- We do this by designing the TBF to accommodate the open Q size for $M_B/D/1$
  - $mQ = n(1+\gamma)-B$
  - $Q=(\rho^2-\rho)/2(1-\rho)$
  - $n=B, \rho = n/m$
  - $\gamma_{opt}=(n-1)/(2m-2n)$

TBF

- TBF must be larger than $n \gamma_{opt}$
- A possible rule is make $TBF = \lceil 2n \gamma_{opt} \rceil$ rounded up to the nearest power of 2.
- Then assume that the achievable $\gamma$ is $\min(0.5 \gamma_{opt}, 1)$ in computing $B(m,n,\gamma)$
Example

Processor Cycle = 10 ns
Memory pipes (s) = 2
Tc = 60 ns
m = 16

Inter-Instruction Bypassing

- If we bypass only within each VLD then when the VOP completes there will still be outstanding requests in the TBF that must be completed before beginning a new VOP.
  - this adds delay at the completion of a VOP
  - reduces advantage of large buffers for load bypassing
- We can avoid this by adding hardware to the VRs to allow inter-instruction bypassing
  - adds substantial complexity
Vector Processor Performance Metrics

- **Amdahl’s Law**
  - Speedup limited to vectorizable portion

- **Vector Length**
  - $n_{1/2} = \text{length of vector that achieves } 1/2 \text{ max speedup}$
  - Limited by arithmetic startup or memory overhead

High-Bandwidth Interleaved Caches

- **High-Bandwidth caches needed for multiple load/store pipes to feed multiple, pipelined functional units**
  - Vector processor or multiple-issue

- **Interleaved caches can be analyzed using the $B(m,n,\delta)$ model**, but for performance the writes and (maybe) some reads are buffered.

- **Need the $B(m,n,\delta,\gamma)$ model where $\gamma$ is derived for at least the writes**
  - $n = n_r + n_w$
  - $\delta_w = n_w/\text{write sources}$
  - $\gamma_{opt} = (n_w - \delta_w)/(2m - 2n_w)$
Example

Superscalar with 4LD/ST pipes
processor time = memory cycle time
4-way interleaved
Refs per cycle = 0.3 read and 0.2 write
writes fully buffered, reads are not

VP vs Multiple Issue

- **VP +**
  - good \( S_p \) on large scientific problems
  - mature compiler technology.
- **VP -**
  - limited to regular data and control structures
  - VRs and buffers
  - memory BW!!
- **MI +**
  - general-purpose
  - good \( S_p \) on small problems
  - developing compiler technology
- **MI -**
  - instr decoder HW
  - large D cache
  - inefficient use of multiple ALUs
Summary

- Vector Processors
  - Address regular data structures
  - Massive memory bandwidth
  - Multiple pipelined functional units
  - Mature hardware and compiler technology

- Vector Performance Parameters
  - Max speedup ($S_p$)
  - % vectorizable
  - Vector length ($n_{1/2}$)

Vector Processors are reappearing in Multi Media oriented Microprocessors