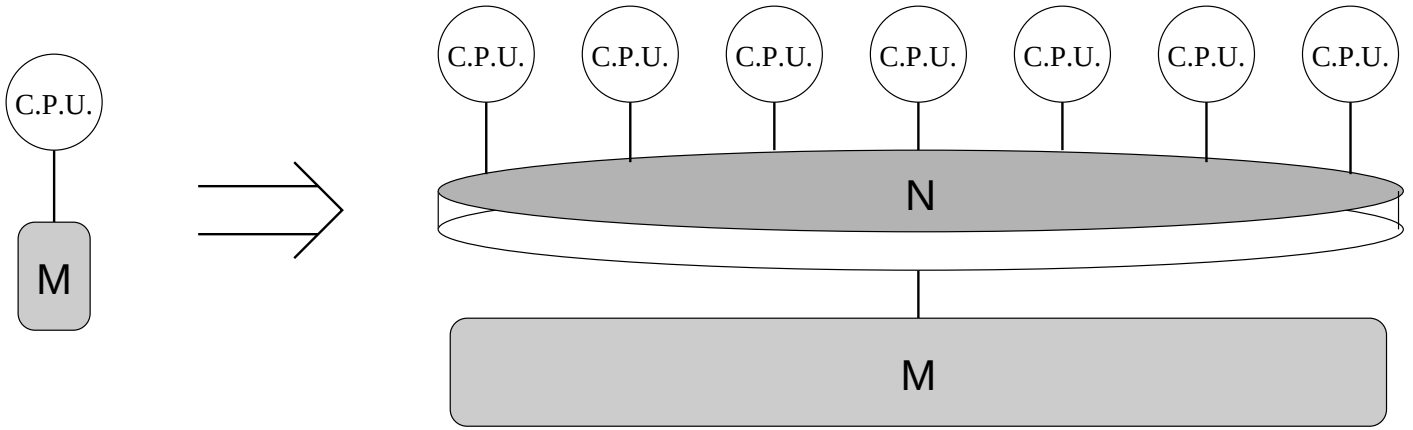
MIMD - Shared Memory Multiprocessors



- Replicated Processing Elements
- Each PE is a full CPU including IPU & ALU [1]
- Single Memory
- Communications Network - Processor to Memory

[1] No extra charge for TLAs

MIMD - Shared Memory Multiprocessors

For years we have multi-tasked on single processors.

We have been giving the user the illusion of one processor per process now we can do it for real!

Unix offers facilities such as -

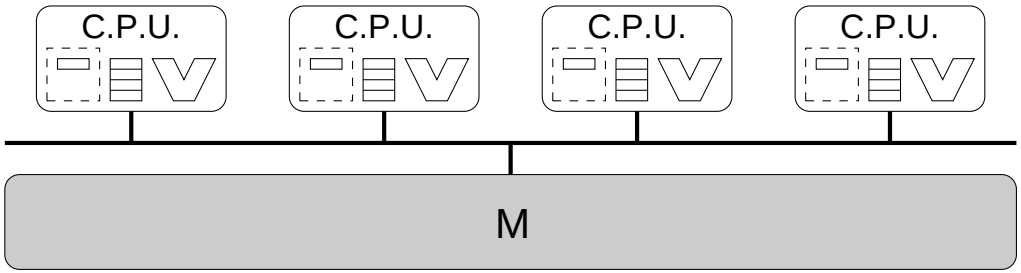
- *fork()* - forks a new process.
- *wait()* - waits for the new process to die.
- *pipe()* - allows inter-process data transfer.
- *semctl()* - semaphores allow process synchronization.

All communication and synchronization is via shared access to memory.

With a few changes to the operating system we can run existing software with significant speed increase.

MIMD - Shared Memory Multiprocessors

Bus Architectures



- Bus Contention

  Multiple requests for bus access.

- Bus Arbitration

  Centralised control over the allocation of bus cycles.

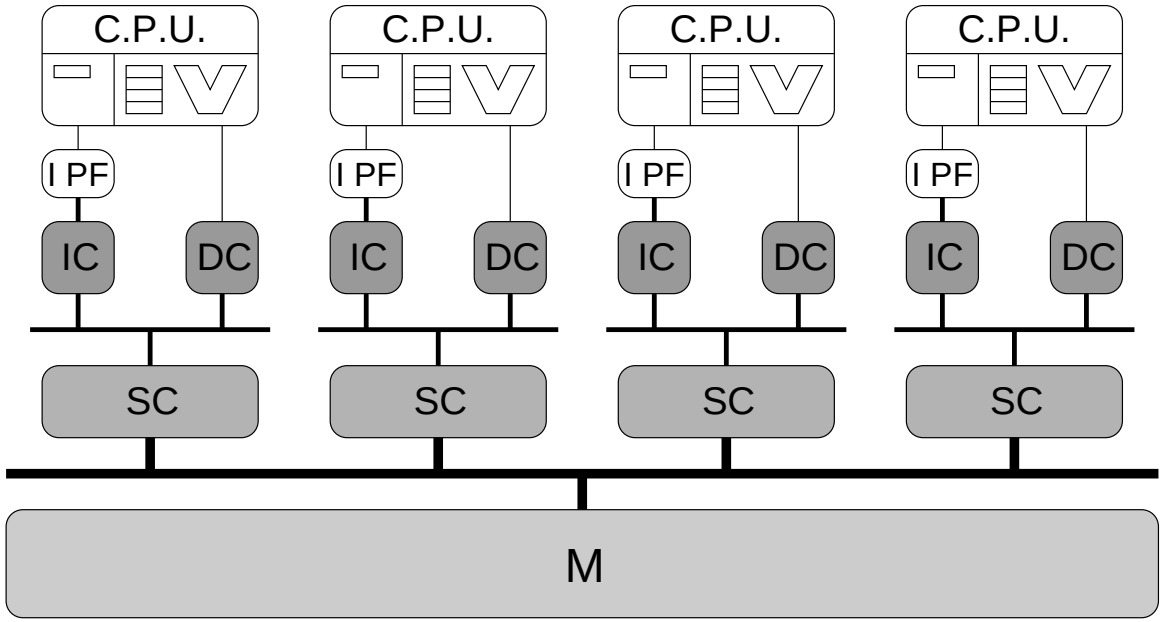
- Delay due to Contention and Arbitration

  Collisions will be more the norm than the exception.

 We have an imbalance between Processing Power and Memory Bandwidth.

Caches

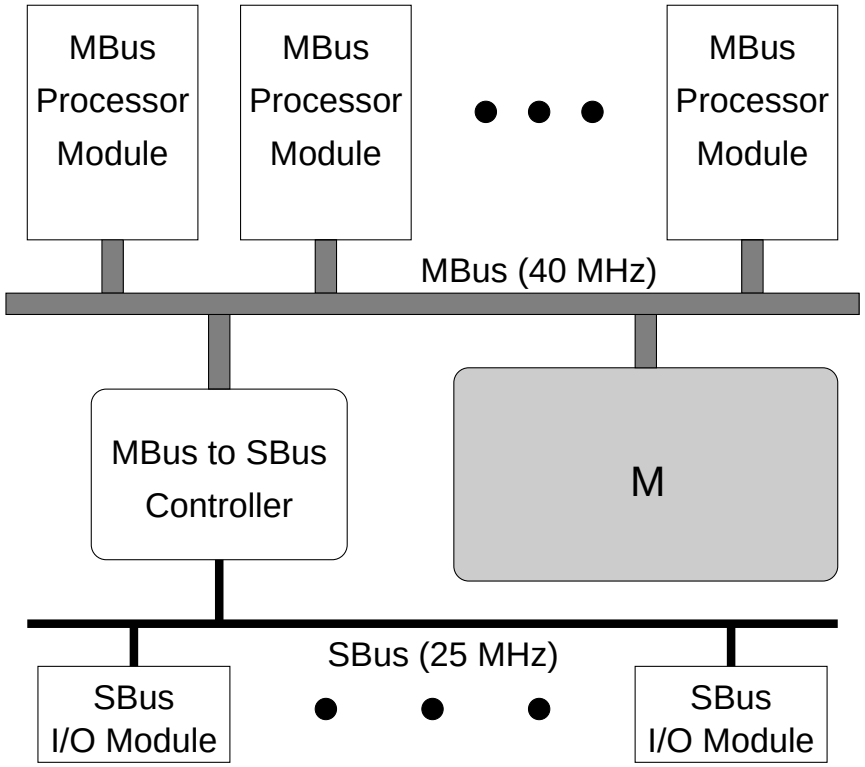
Caches can be used to reduce the number of global memory accesses.

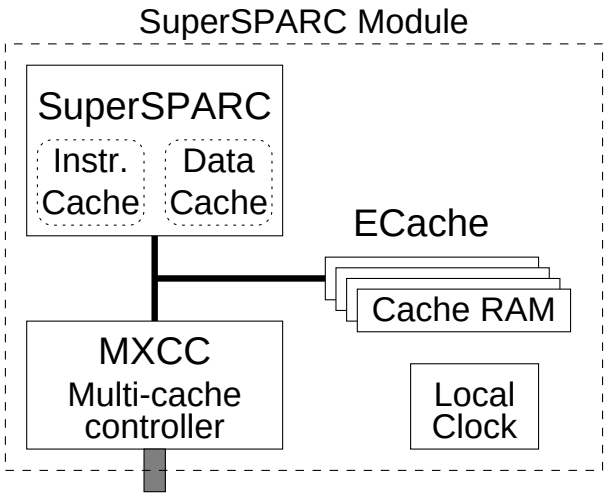


Using the full range of caching techniques available, we can balance the Processing Power and Memory Bandwidth of small bus based systems.

SPARC MBus Architecture

e.g. SPARCstation 20





- MBus is synchronous at 40 MHz.
- Processor Modules have local clocks (50 MHz SuperSPARC, 150 MHz HyperSPARC).
- MBus to SBus controller provides all centralized control functions.

SPARC MBus Architecture

- *Write Back* and *Fetch on Write*
  These strategies help to reduce bus traffic.
- Cache States
  To support Cache Coherence protocols, each cache line is tagged with its state:
  - Invalid
  - Exclusive Clean
  - Shared Clean
  - Exclusive Modified
  - Shared Modified
- Ownership
  - A cache line may be owned by at most one cache.
    Ownership is indicated by a *modified* state.
    Only a *modified* line need be written back on discard.
  - A line which is not owned by any cache is owned by the main memory.

SPARC MBus Architecture

- *Direct Data Intervention* - cache to cache data transfer

  In order to support the *write back* strategy, the owner of a modified cache line must supply it to any processor which requests it from main memory.

  – All caches must *snoop* MBus for read.

  In fact a modified line may be can be handed over to a new owner without it being written to main memory.

- *Write Invalidate*

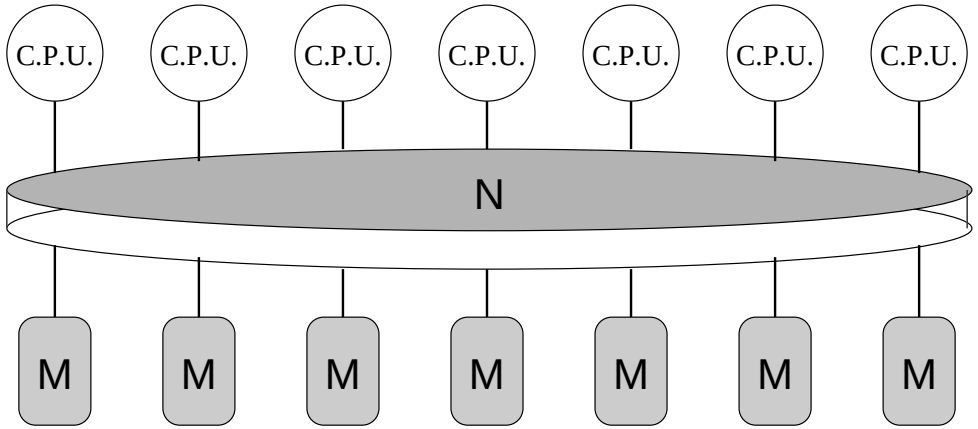
  Must invalidate all other cache copies before a write to cache.

  – All caches must *snoop* MBus for invalidate.

  This maintains *cache to cache* coherence. The *write back* strategy precludes *cache to main memory* coherence.

MIMD Shared Memory Multiprocessors

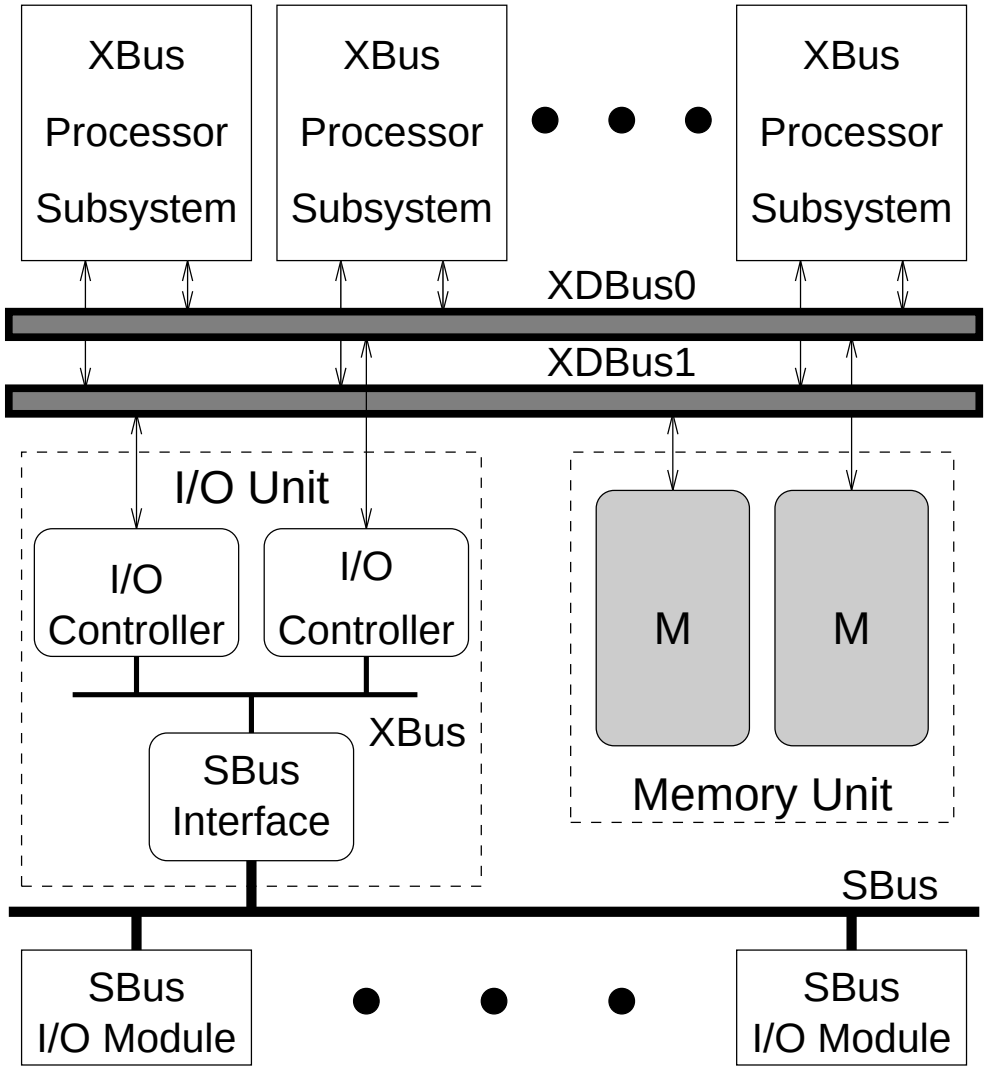
Multi-Bank Memories

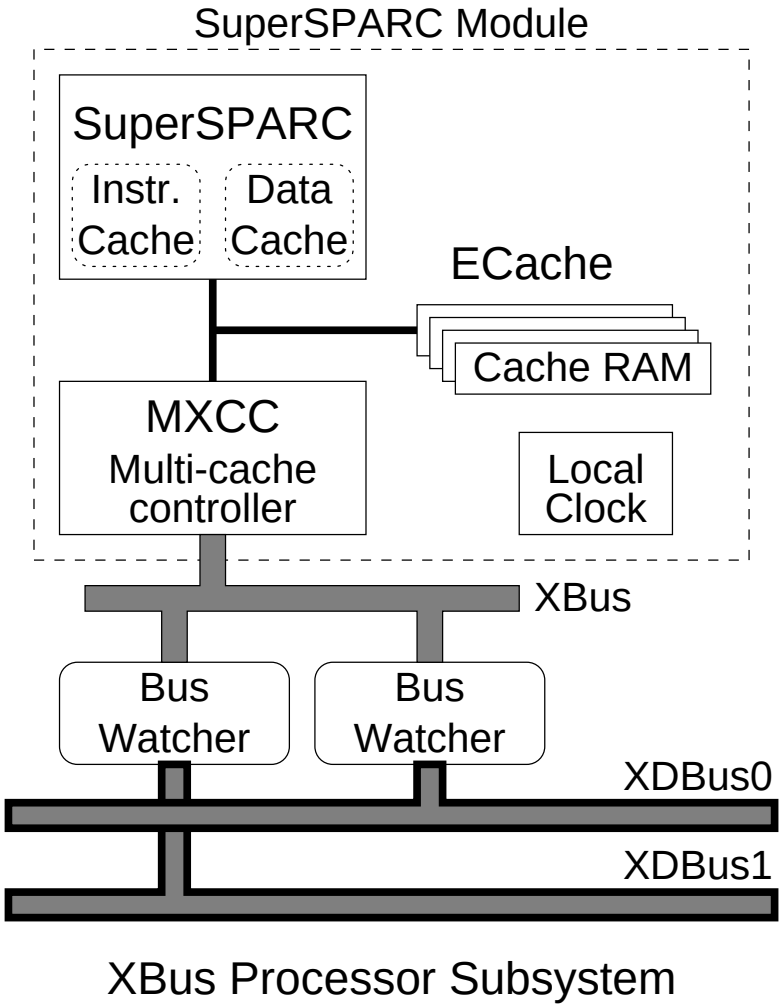


- Within a single memory cycle, we allow one memory access per bank.
- In this way we can dramatically increase effective Memory Bandwidth.

SPARC XBus Architecture

e.g. SPARCserver 2000





XBus Processor Subsystem

SPARC XBus Architecture

- XDBusses are packet switched rather than exclusive use.
  - Exclusive use bus:
    A processor controls the bus throughout a transaction. Although the bus may be idle, no other processor may make use of it.
  - Packet switched bus:
    A processor sends an addressed packet to request a transaction.
    The addressee replies when the transaction is complete.
    Other requests and replies may occur between request and reply.
    Transactions requests are queued allowing for efficient bus usage and pipelined memory access.
- SPARCserver 2000
  - Twin XDBus connects to 2 Interleaved Memory Banks.
  - Supports up to 20 processors

Multi-Bank Memories

We still have problems with Contention.

- We have *pathological* programs which require all processors to access the same bank simultaneously.
- Distribute data such that adjacent memory locations are on different banks.

  Use a prime number of banks and a power of two of processors.
- Distribute Randomly

  Use a hashing scheme to determine the location of a data value.

The cost of the system can be very high[2] in order to allow wide bandwidth connection between any processor and any memory bank.

[2]multiprocessor CRAY X-MP systems employ this approach.

A Success Story

With few software problems these machines are becoming popular:

- Single Bus systems have found their way onto the desktop.
- Multi-Bank systems with up to 20 processors are used as servers, replacing more traditional mainframes.

N.B. In most cases these systems are not being used for parallel programs. A single program will usually use only one processor. The performance of such programs is limited by the performance of an individual processor.

The Limits

The number of processors in a shared memory system is restricted by the following factors.

- Central Control
- Memory Collisions
- Bus Physical Size
- Interconnect Complexity

With between 2 and 20 processors these are not Massively Parallel Systems.

Todays supercomputers tend to use other architectures.