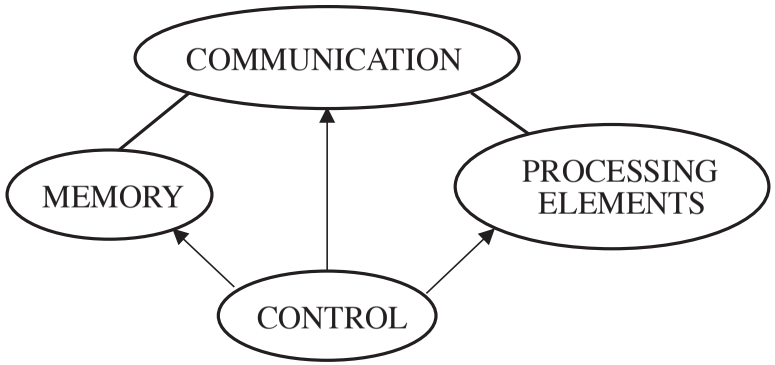
DSP ARCHITECTURES

IDEAL DSP ARCHITECTURES



Architectural components

An *ideal DSP architecture* belongs to a class of architectures that implements the static schedule.

An ideal architecture has processing elements that can execute the operations according to the schedule and is supported with appropriate communication channels and memories.

Processing Elements

Processing elements (*PE*) usually perform simple, memoryless mappings of the input values to a single output value. The arithmetic operations commonly used in DSP algorithms are

add/sub, add/sub–and–shift

multiply, multiply–and–accumulate

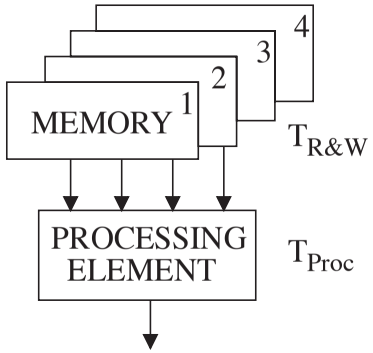
vector product

two-port adaptor

butterfly

We will reserve the more general term *processor* to denote a PE with its internal memory and control circuitry.

Hence, a processor is able to perform a task independently of other processors.



Processing element with multiple inputs

At this point it is interesting to note that the execution time for processing elements and the cycle time (read and write) for memories manufactured in the same technology are of the same order.

Hence, to fully utilize a multiple-input processing element one memory or memory port must be provided for each input and output value.

Synchronous and Asynchronous Systems

Timing philosophy is one of the most important attributes characterizing a computing system. There are two types of possible timing schemes—*synchronous* and *asynchronous*.

Synchronous timing techniques are characterized by the existence of a global clock that defines the basic unit of time.

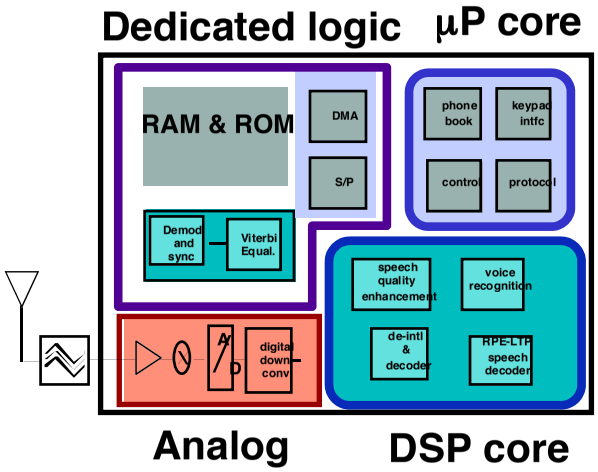
This time reference is used to order the activities into a proper sequence.

The whole system is an isosynchronous domain. This is in practice accomplished by a central, global clock that broadcasts clock signals to all units in the system so that the entire system operates in a lock-step fashion.

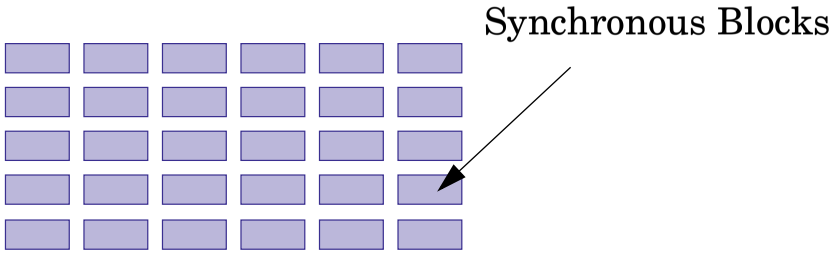
Asynchronous timing techniques operate without a global clock.

Communication among operational units is performed by means of inter-lock hand shaking schemes.

PLATTFORM-BASED IMPLEMENTATIONS



Globally Asynchronous and Locally Synchronous Architectures
GALS



MULTIPROCESSORS AND MULTICOMPUTERS

General-purpose parallel or distributed computer systems can be divided into two categories: *multiprocessors* and *multicomputers*.

The main difference between these two categories lies in the way in which communication between the processors is organized.

All processors share the same memory space in a multiprocessor system while in a multicomputer system each processor has its own private memory space.

Multiprocessor systems can be subdivided into tightly coupled and loosely coupled systems.

A tightly coupled system has a common main memory so that the access time from any processor to memory is the same.

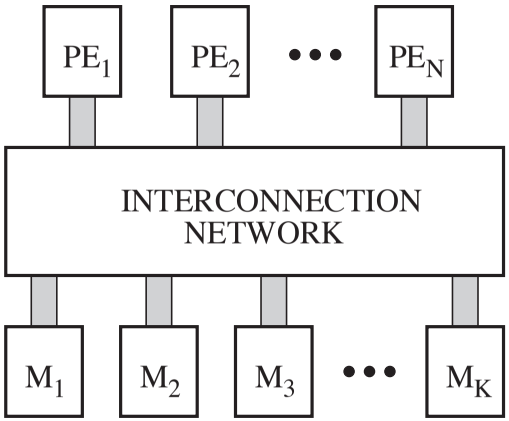
In a loosely coupled system the main memory is distributed among the processors, although the processors share the same memory space.

The term *processor array* is used when the processing elements are connected in a regular manner.

An *array processor* is an array of processing elements doing the same operation but on different data items.

SHARED-MEMORY ARCHITECTURES

A special case of a multiprocessor architecture, henceforth called *shared-memory architecture*



The shared-memory architecture can accommodate only a small number of processors due to the memory bandwidth bottleneck.

Memory Bandwidth Bottleneck

Each processor must be allocated two memory time slots: one for receiving inputs and the other for storing the output value into the memories.

To fully utilize N processors with execution time T_{PE}, the following inequality must hold:

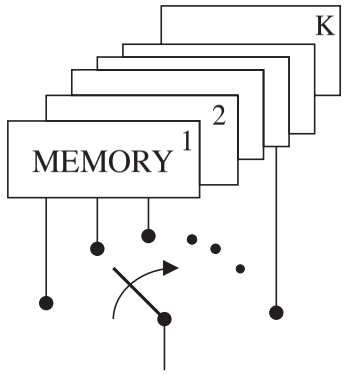
$$T_{PE} \geq 2NT_M$$

where T_M is the cycle time for the memories.

However, T_{PE} and T_M are of the same order. Hence, very few processors can be kept busy because of this *memory bandwidth bottleneck*.

Thus, there is a fundamental imbalance between computational capacity and communication bandwidth in shared-memory architecture.

Reducing the Memory Cycle Time



Interleaving of *K* memories

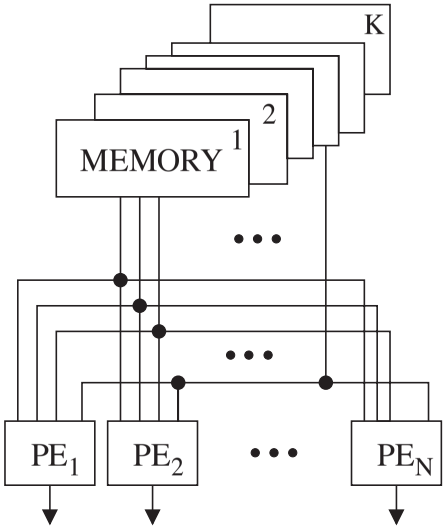
Cache/Buffer Memories

The communication demand can be reduced by providing the processors with fast private memories.

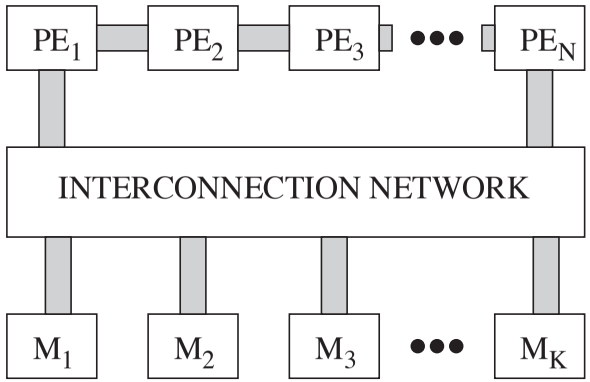
The processor can therefore access its cache memory without interference from the other processors.

Reducing Communications

Broadcasting

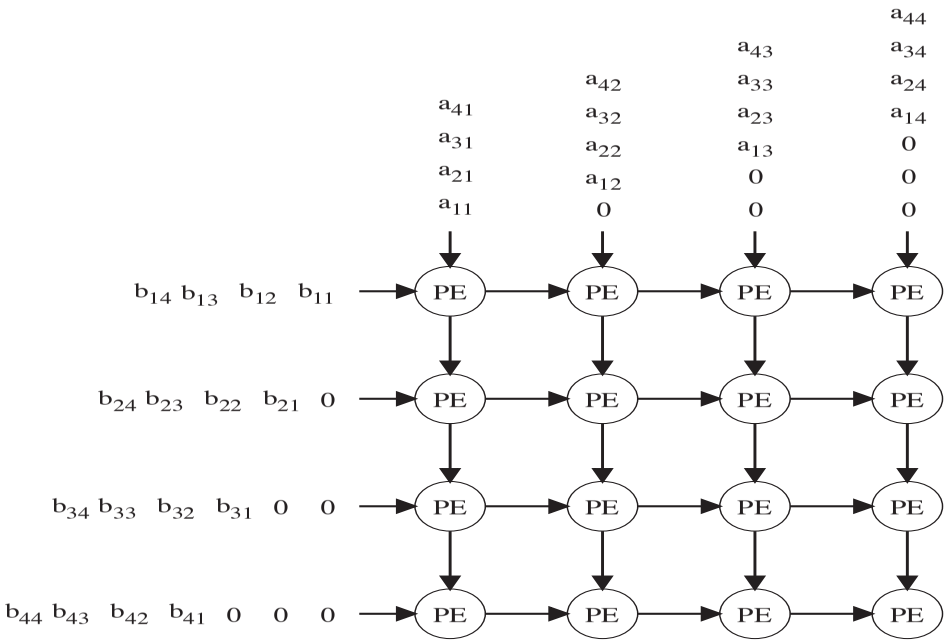


Interprocessor Communication



SYSTOLIC ARRAYS

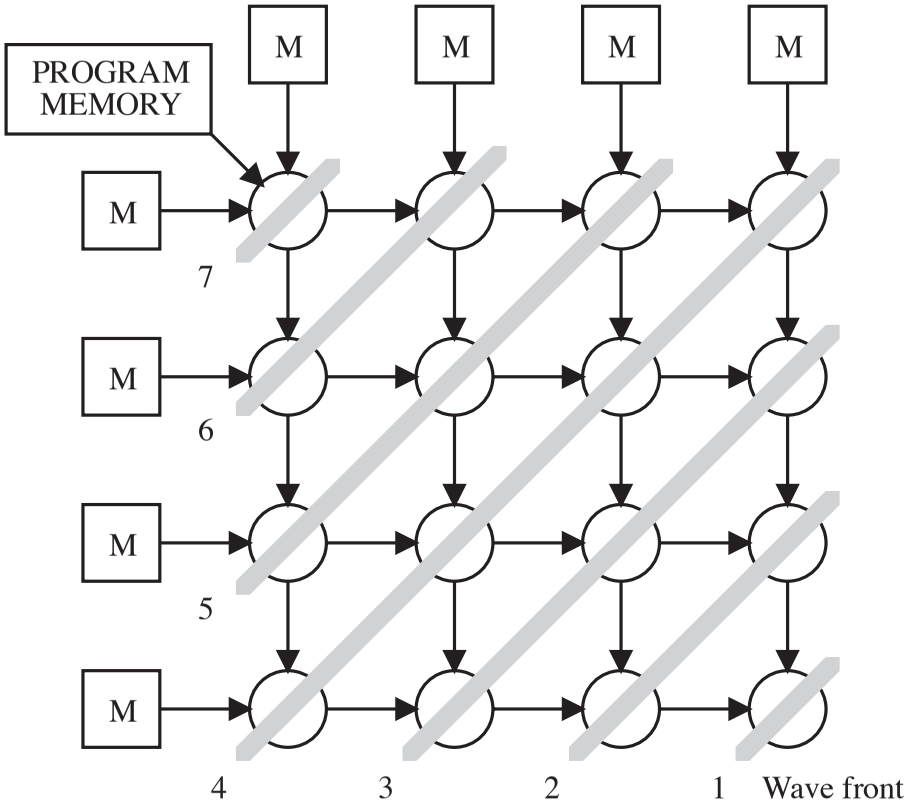
A *systolic array* is defined as a lattice of synchronous and locally connected PEs that can perform iterative algorithms with regular data dependencies.



A systolic array is an *n*-dimensional structural pipeline with synchronous communication between the PEs.

WAVE FRONT ARRAYS

A wave front array is an *n*-dimensional structural pipeline with asynchronous communication between the PEs.



Large Basic Operations

The third factor in Inequality affecting architectural balance is execution time for the PEs.

Obviously, if we use PEs with a large granularity, execution time will be longer, however more useful work will be done.

For example, a butterfly PE is preferred over separate PEs that perform simpler operations such as add, subtract, and multiply.

Further, fewer memory transactions may be needed if direct interprocessor communications are allowed.