SIMD Parallel Computers

Although the term SIMD can be applied to pipelined vector computers, we are concerned here with replicated parallel machines only.

The most significant machines in this area are

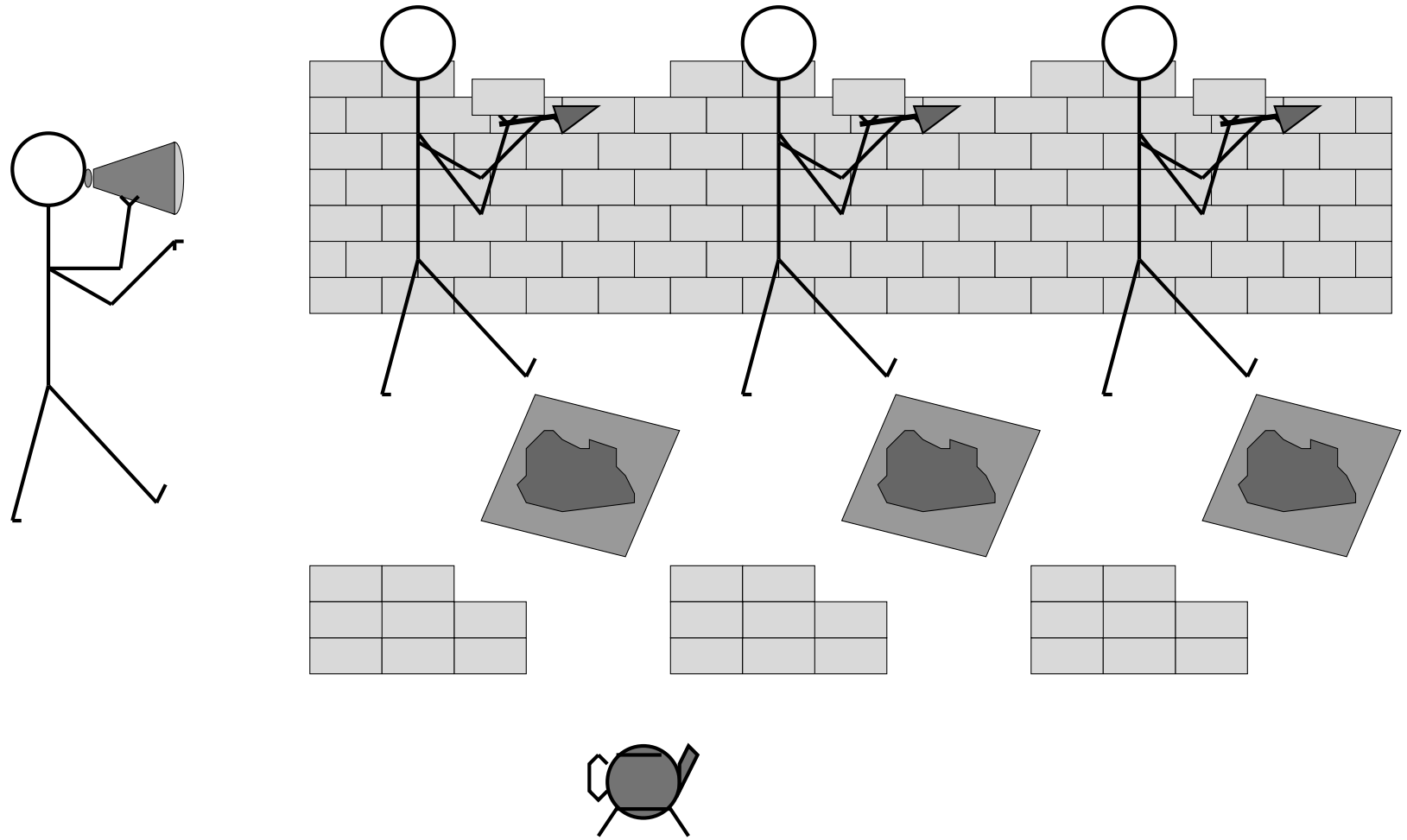
- DAP (Distributed Array of Processors) from Active Memory Technology
- CM-1 and CM-2 Connection Machines from Thinking Machines

SIMD machines typically have a single master *Instruction Processing Unit* (IPU) and a number of slave *Processing Elements* (PEs).

The master IPU reads instructions from memory, makes decisions on flow of control and broadcasts instructions to the slave PEs. The slave PEs obey these instructions in lockstep.

In order to look at the benefits and pitfalls of this system let us look at wall building by 'SIMD Construction Ltd.'.

SIMD Construction Ltd. - *Wall builders to the Pharaohs*



8002

Benefits of SIMD Architectures

- Co-ordination is centralised, Control is easy.

  Because the slaves work in lockstep under the control of a single master, their behaviour is predictable and synchronized. The broadcast mechanism is the only mechanism required for co-ordination and control.

- Slave labour is cheap.

  The slave PEs are very simple calculating engines, there is no requirement for them to be able to read an instruction list or make decisions for themselves. This simplicity reduces the marginal cost of extra slaves.

- Communication is simplified.

  If the slaves are each told to pass a brick to the left and receive one from the right, this can happen immediately and without handshaking, because the slaves remain synchronized at all times.

Problems with SIMD Architectures

Limits of a single instruction stream

We don't always want all the slaves to do the same task.

- Boundary Effects
  We would like slaves at the boundaries to behave differently.
  - Placing a half brick at the end of a row.
- Insufficient Work
  Sometimes we have more slaves than we can use.
  - 1000 Slaves building a 500 brick wide wall.
- Sequential Tasks
  Certain tasks cannot be divided amongst the slaves.
  - Making the Tea.

Problems with SIMD Architectures

Coping with a single instruction stream:

- ## Activity Control

  We arrange for certain slaves to ignore the instructions from the master, dependent on local data.

  Although we cope with the problems we do not make efficient use of our resources. Many slaves may sit idle.[1]

- ## Specialist Scalar Processing

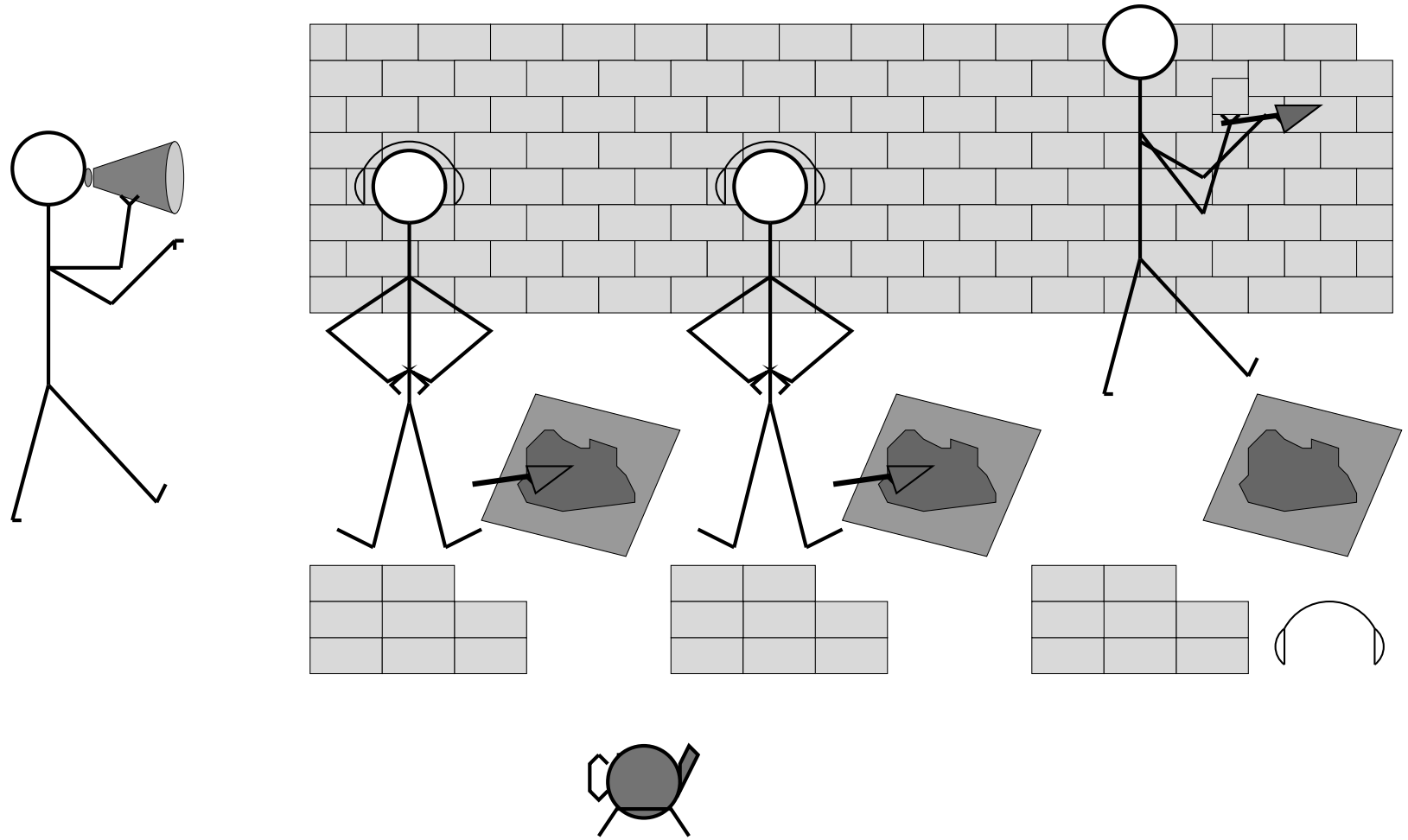
  We might arrange for certain sequential tasks to be done by the master processor.

  We can optimize the master to carry out these tasks without increasing the complexity of the replicated processors.

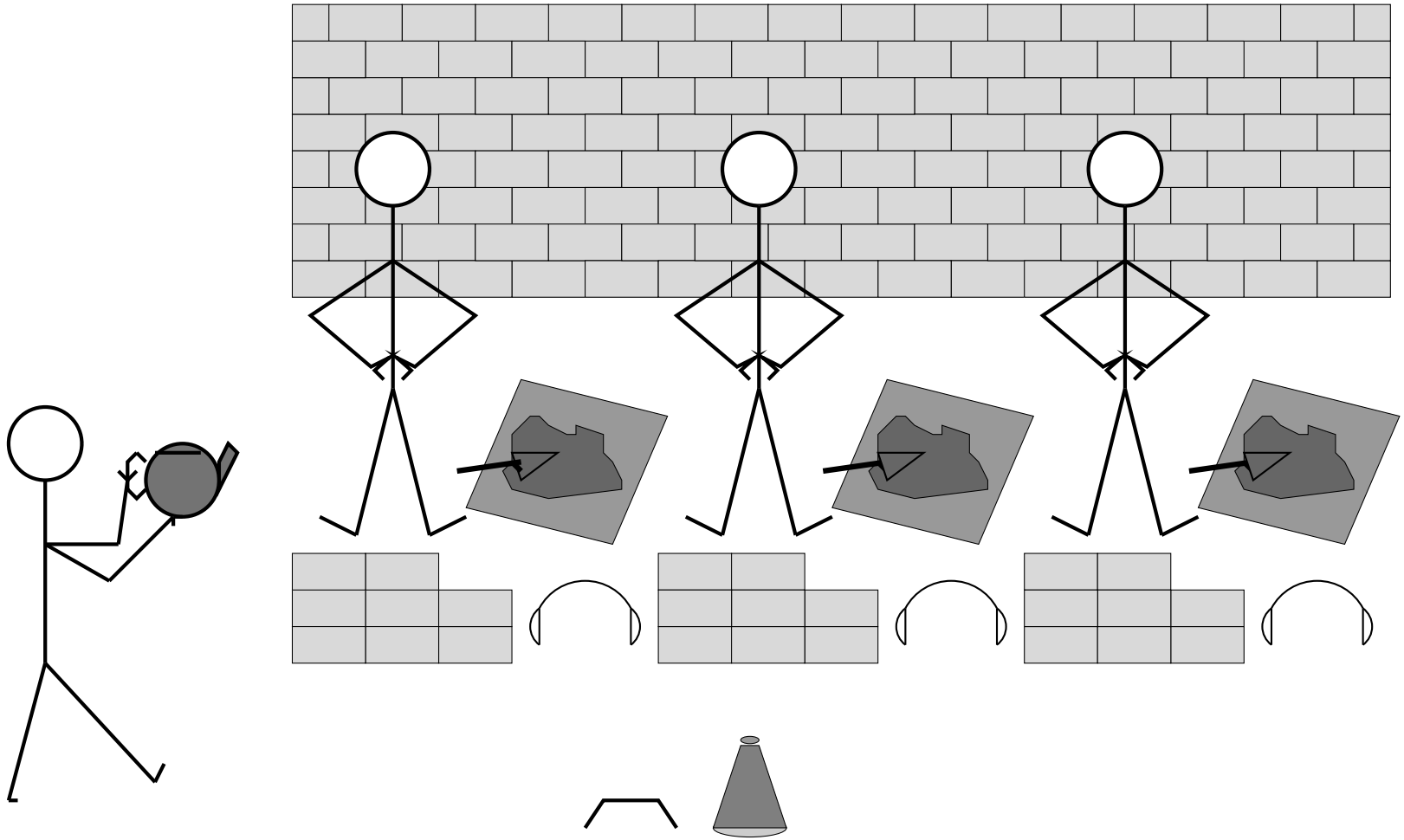
  Note that our slaves will still be idle during sequential tasks.

[1] c.f. memory in sequential machines – large amounts of memory will speed up processing by being available when it is needed.

Activity Control for SIMD machines

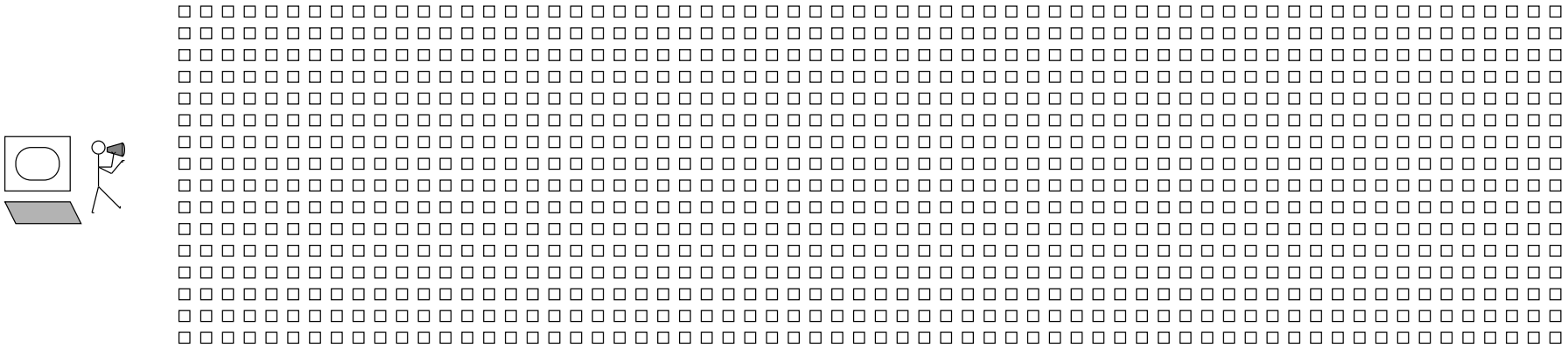


Sequential Processes on SIMD machines



SIMD - Processor size and number

Due to the simplicity of the slaves, with no overheads for co-ordination and very simple communication requirements, SIMD machines frequently have a very large number of very simple processors.



- The first versions of both the DAP and the Connection Machine employ *single bit* processing elements.
  - The DAP 510 has 1024 PEs, while the CM-1 has between 4096 and 65536 PEs.

SIMD - Exploiting Parallelism

We have suggested that parallelism within a problem or algorithm may be identified and classified. We have so far mentioned data parallelism and the more general process parallelism.

- ## Data Parallelism

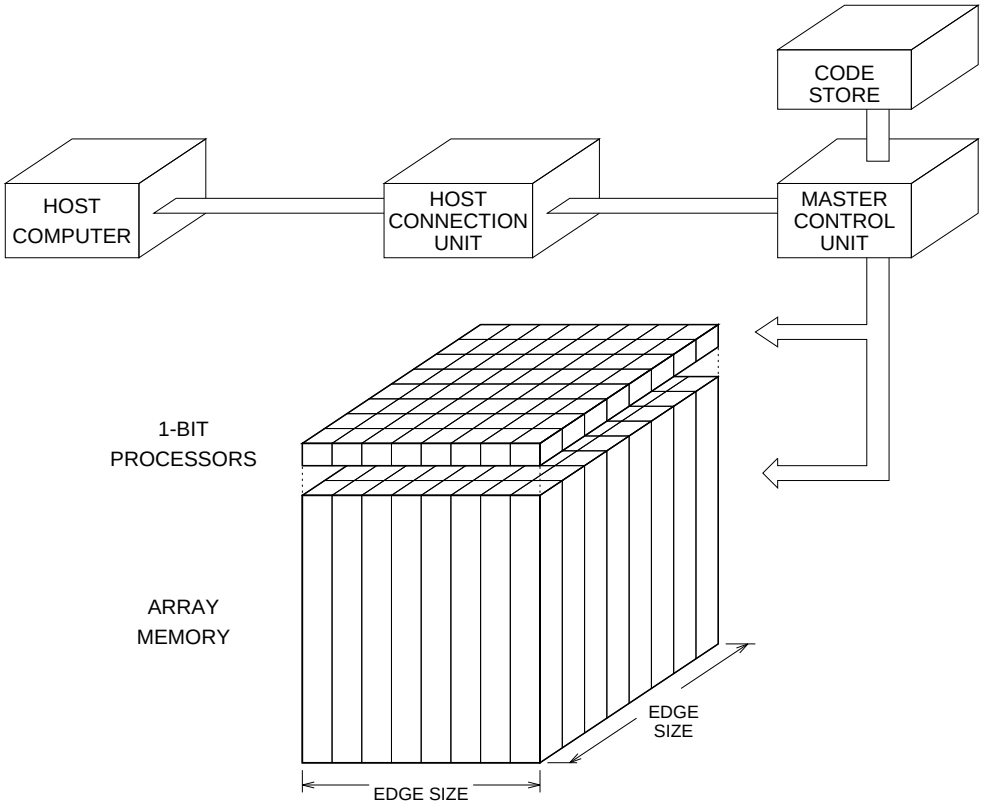
  SIMD machines are ideally suited to solving problems which exhibit *Data Parallelism* because they perform the same operation simultaneously on all Processing Elements.

  Thus they can perform the same operation over the whole of a data structure distributed amongst the processors.

- ## Process Parallelism

  In fact due to the restriction of a single IPU, they are unable to exploit more general *Process Parallelism*. *Process Parallel* constructs must be re-sequenced before execution.

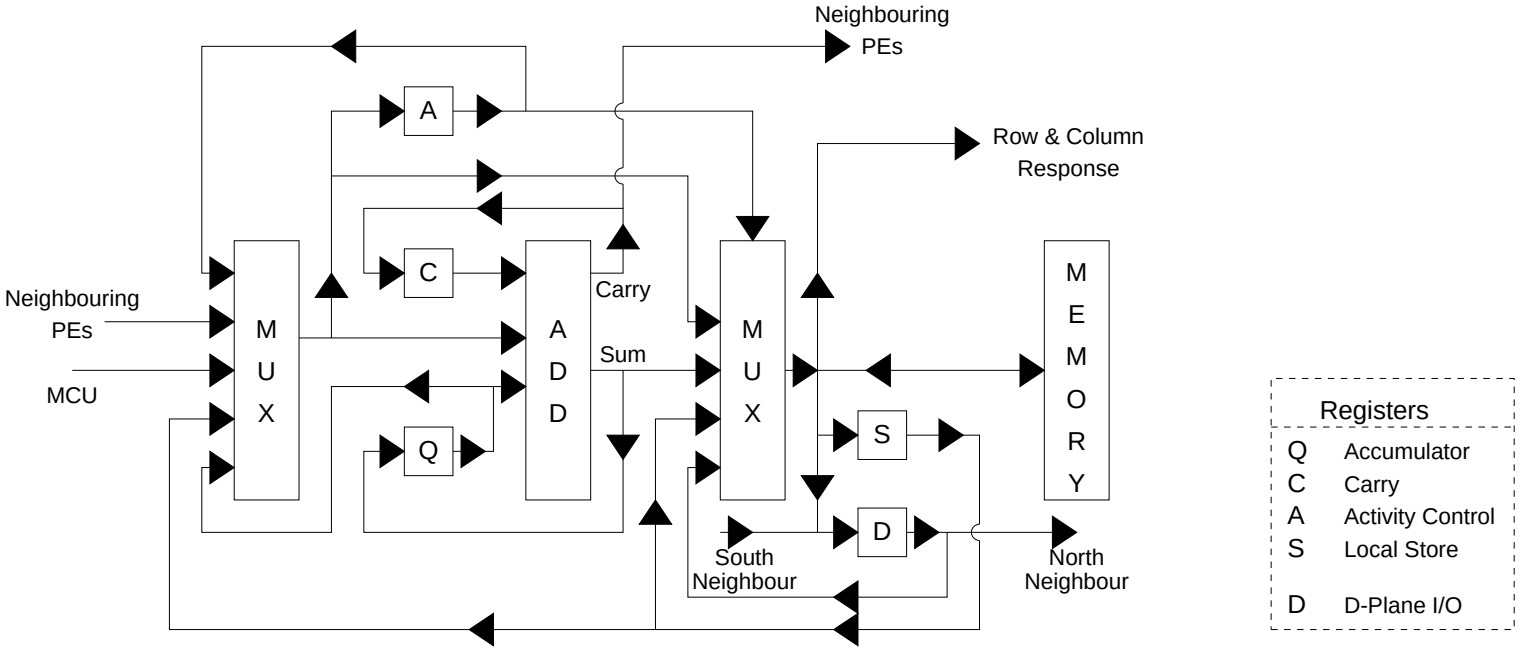
DAP - Distributed Array of Processors



- Originally developed by ICL (1976).
- VLSI DAP 500 (1024PEs) & DAP 600 (2048PEs) built by AMT (1986-)
- Specializes in vision processing and finite element problems.

DAP

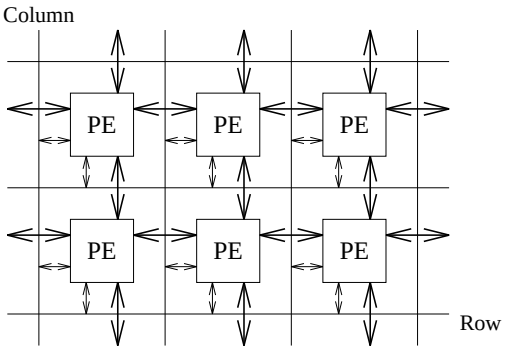
- Processing Element Architecture



– Simple single bit PE
  - - 64 PEs per VLSI chip
– Single bit data path to memory
  - - 32 or 64 kbits SRAM per PE

DAP

- Host Computer
  - Runs sequential parts of code (Fortran 77 or C)
- Master Control Unit (Instruction Processing Unit)
  - Runs Fortran Plus array processing code
  - Controls PEs & Addresses Memory
  - Makes all global decisions
- PE connectivity
  - N S E & W Nearest neighbour connections
  - Row and Column Highways

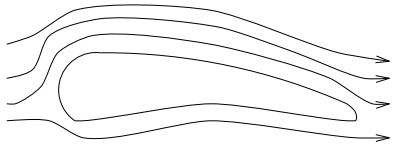


8012

Data Mapping for an Array Processor

Map data to make full use of the limited connectivity offered by the PE array.

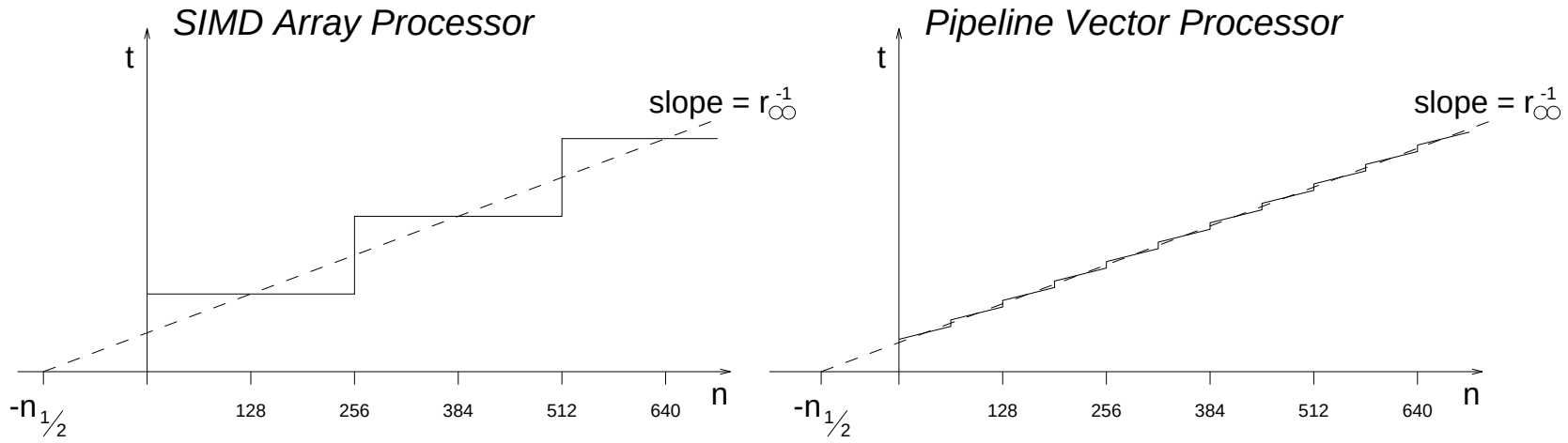
The DAP's nearest neighbour connectivity makes it very good at solving *1D*, *2D* and *3D* Finite Element Analysis problems.



- Array sized data
  - *1D* Array
    All PEs can be configured as a long vector (1024/4096 elements)
  - *2D* Array
    Direct mapping to PE array.
  - *3D* Array
    Third dimension is stored in memory.

Array Processor vs Pipeline Vector Processor

The array processor performance curve may be approximated to that of a generic pipeline vector machine, giving us a direct means of machine comparison. Here we see the curves for an array processor and a real vector processor.[2]



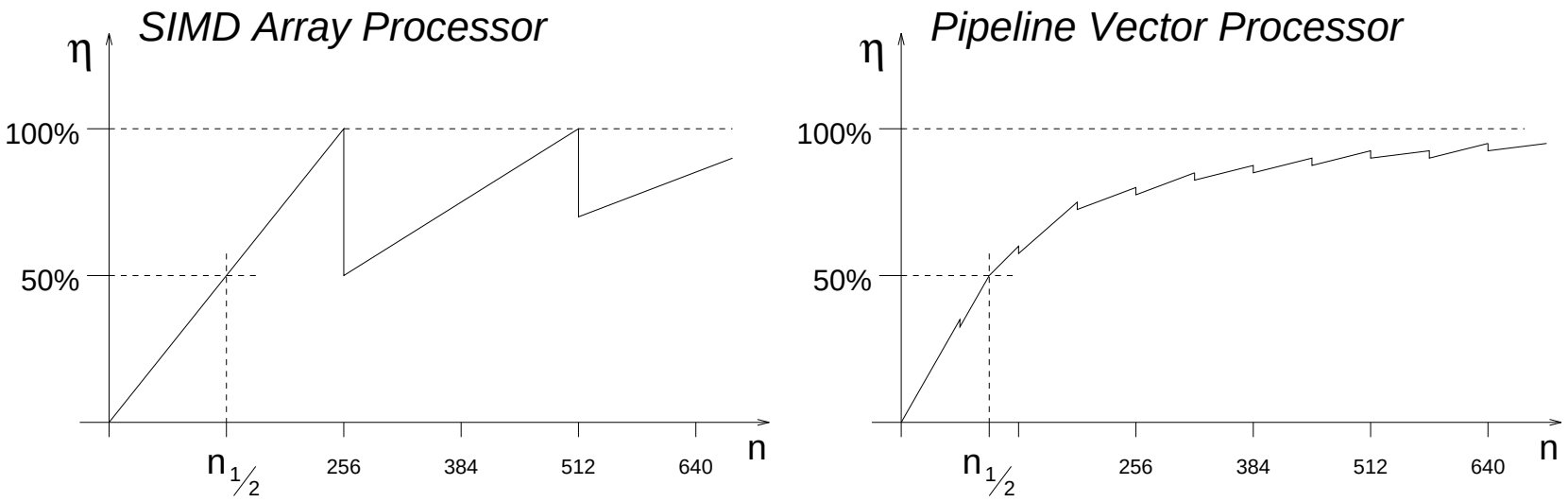
We see that $n_{1/2} = Array\_size/2$ for array processor, while it is more determined by pipeline length and startup time for the vector processor.

[2]The array processor typically has greater number of less powerful nodes, as shown.

Further Performance Measurement

- Vector Efficiency, η.

  This measure is defined as the actual processing rate divided by the asymptotic rate.



Here we can see that the array processor actually reaches its "asymptotic" performance, $\eta = 100\%$, for values of n divisible by the array size.

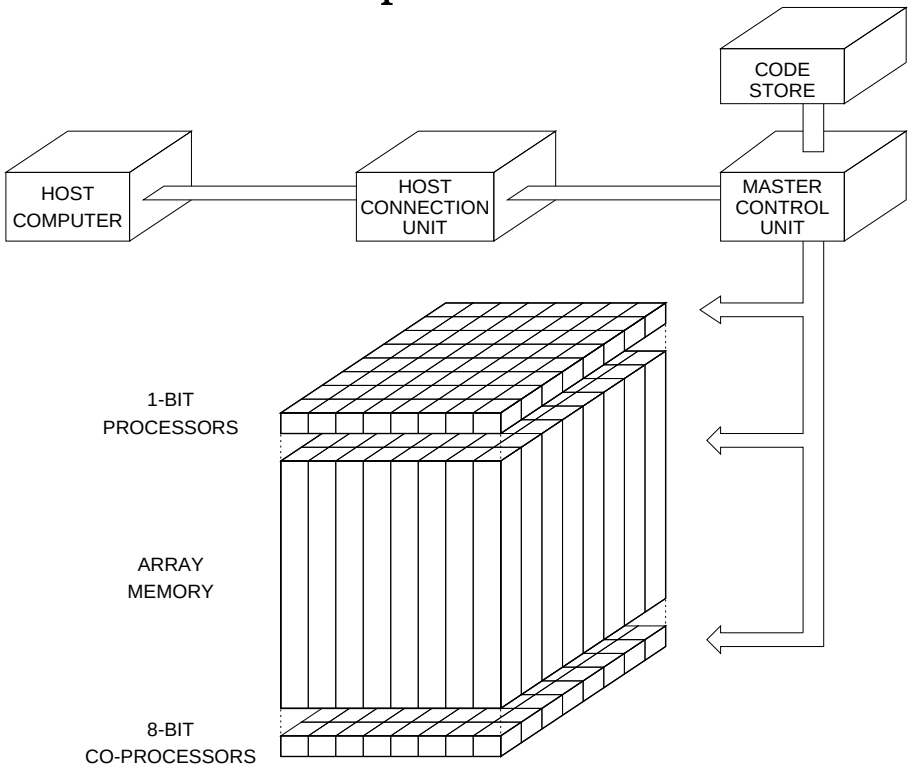
Using Single Bit PEs

- Boolean calculations
  - Very Efficient - Very Powerful.

- Numeric calculations
  - Numbers are normally mapped 'vertically' within the memory of a single PE.
  - The architecture permits 'horizontal' mapping of short vectors for increased efficiency.

- Variable Precision Arithmetic
  - Calculation with reduced precision is carried out at greater speed.
  - Greater efficiency than fixed width processors.

For improved multi-bit integer and floating point performance each PE may optionally be provided with an 8-bit co-processor.



N.B. A larger co-processor would cause an imbalance between processor power and memory bandwidth (just 1 bit per cycle).