Classification of Parallel Computers

Parallel computers can be classified by their datapath architecture and their control unit architecture.

Datapath Classification

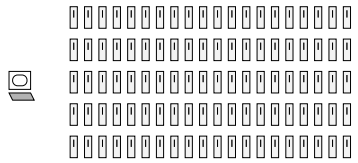
Parallel computers have traditionally been divided into

- *Pipelined Vector Computers*
- *Replicated Parallel Computers*

Although this distinction is still clear, there are an increasing number of machines that are

Replicated Parallel Pipelined Vector Computers.

Replicated Parallel Computers.



Replicated parallel machines will have a number of identical processing elements. These processing elements must "communicate and co-operate" in order to solve problems.

This description leads us to a number of questions:

- What constitutes a processing element?
- How many processing elements of what complexity?
- How do they communicate?
- How do they co-operate?

Replicated Parallel Computers – Summary of Issues

What constitutes a processing element?

What parts of our sequential computer do we replicate?

- Do we replicate control structures or keep central control?
- Do we replicate memory or keep common memory?

We have already suggested that not replicating screen and keyboard is likely to be a good move. We will see that decisions made here can greatly effect the overall architecture of a parallel machine.

Replicated Parallel Computers – Summary of Issues

How many processing elements of what complexity?

Given that we have a limited budget, how should we spend our money?

- We could have *many simple* processing elements.
- We could have *fewer complex* processing elements.

In the spectrum of replicated parallel machines we have machines with very large numbers of tightly coupled small processors, and machines with relatively few loosely coupled large processors.

There are as many opinions on what constitutes a good parallel machine as there are research groups working in the field.

Replicated Parallel Computers – Summary of Issues

How do they communicate?

What systems do we provide to transfer data around our parallel machine?

- What network topology do we use?
- What Message Passing system do we use?

Data transfer is often the weakest part of a parallel computer, we must consider it carefully.

Replicated Parallel Computers – Summary of Issues

How do they co-operate?

This last question deals with the difficult task of getting a number of machines to solve a single problem.

- Programming - How much can the compiler do for us?
- Should there be some central co-ordinator?
  - How would it communicate with the replicated processing elements?

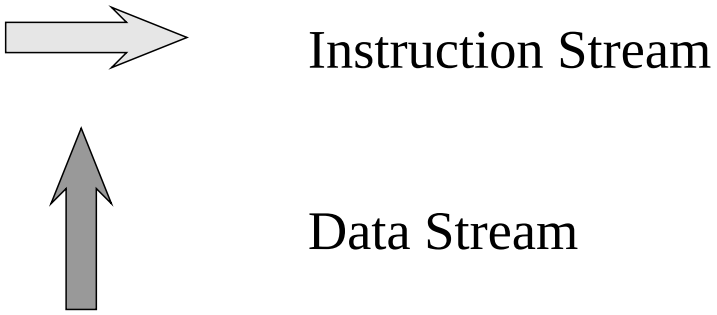
Parallel machines are notoriously difficult to program

Flynn's Taxonomy

No introduction to parallel architecture would be complete without a look at Flynn's Taxonomy.

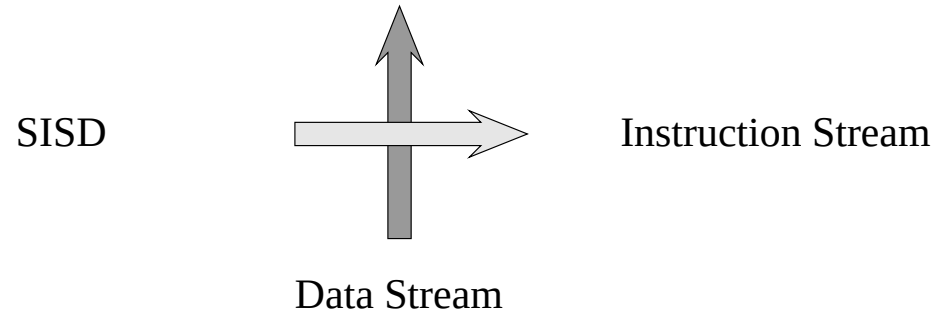
Flynn classifies computers according to how the machine relates its instructions to the data being processed.

A stream is defined as a sequence of items (instructions or data) as executed or operated on by a processor.



Flynn's Taxonomy

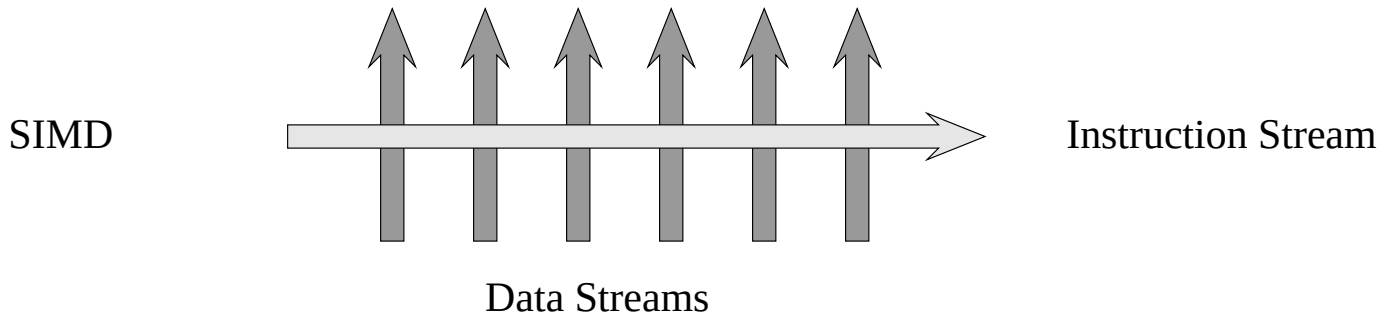
SISD - Single Instruction Stream / Single Data Stream



- This is a conventional sequential machine.
- There is a single instruction stream and thus in practice a single instruction processing unit.
- Each arithmetic instruction results in one arithmetic operation.

(Example: Sinclair ZX81)

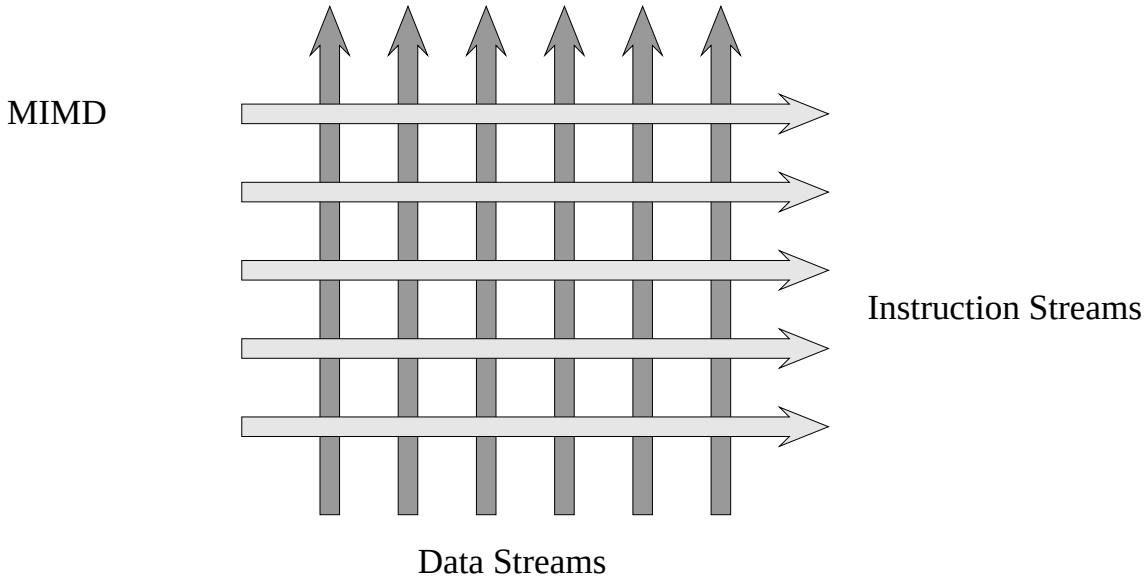
SIMD - Single Instruction Stream / Multiple Data Stream



- This machine retains the single instruction stream of SISD machine, but supports instructions which initiate many operations on conceptually different streams of data.

(Examples: CRAY 1 Pipelined Vector Computer, AMT DAP Array Processor)

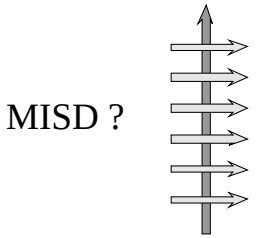
MIMD - Multiple Instruction Stream / Multiple Data Stream



- The multiple instruction streams lead to a number of instruction processing units each with one or more data streams.

(Examples: Intel Hypercube, Transputer Array)

MISD - Multiple Instruction Stream / Single Data Stream



- There are no examples of machines which apply multiple instruction streams to the same stream of data, although Flynn tells us that they might exist.

The main distinction that will concern us is that between SIMD and MIMD.[1] This is the *Control Unit Architecture Classification* that was mention earlier.

- SIMD machines have a single instruction processing unit.
- MIMD machines have a number of different instruction processing units.

[1]Note that a Pipelined Vector Computer is SIMD, but a Replicated Parallel Computer or a Replicated Parallel Pipelined Vector Computer may be SIMD or MIMD.

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

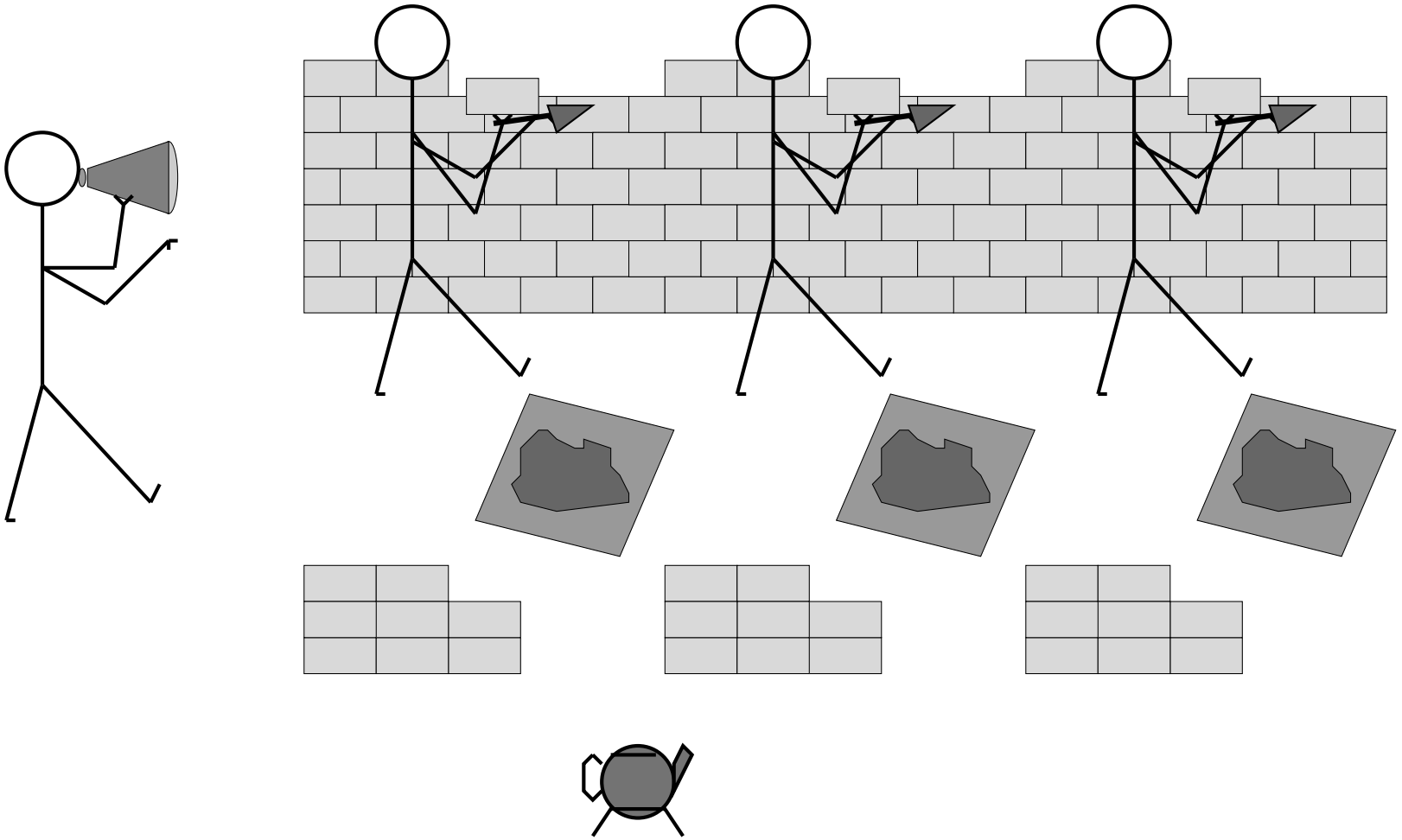
- DAP (Distributed Array of Processors) from Active Memory Technology
- CM-1 and CM-2 Connection Machines from Thinking Machines

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*



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.[2]

- ## Specialist Scalar Processing

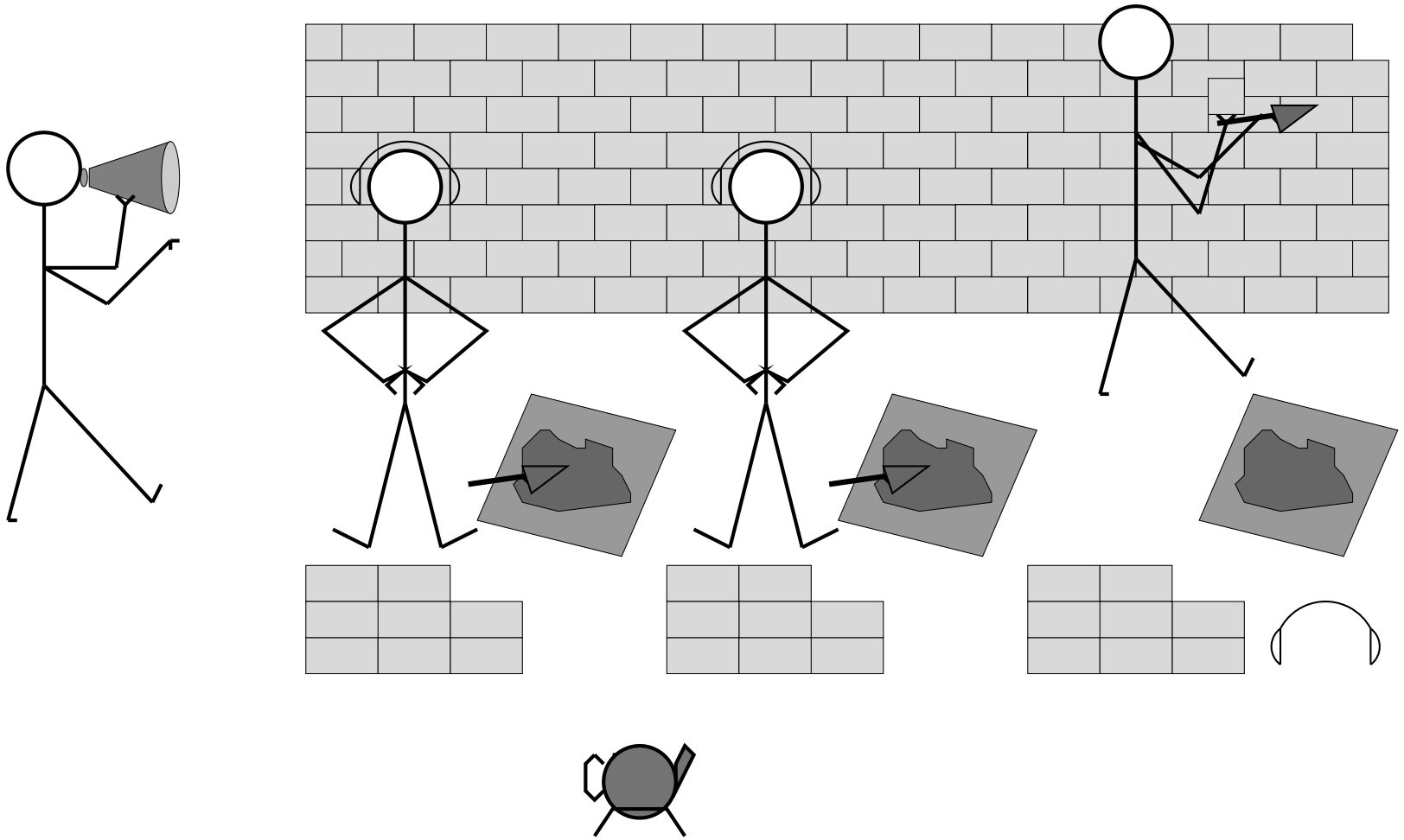
  We might arrange for certain sequential tasks to be done by the master processor.

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

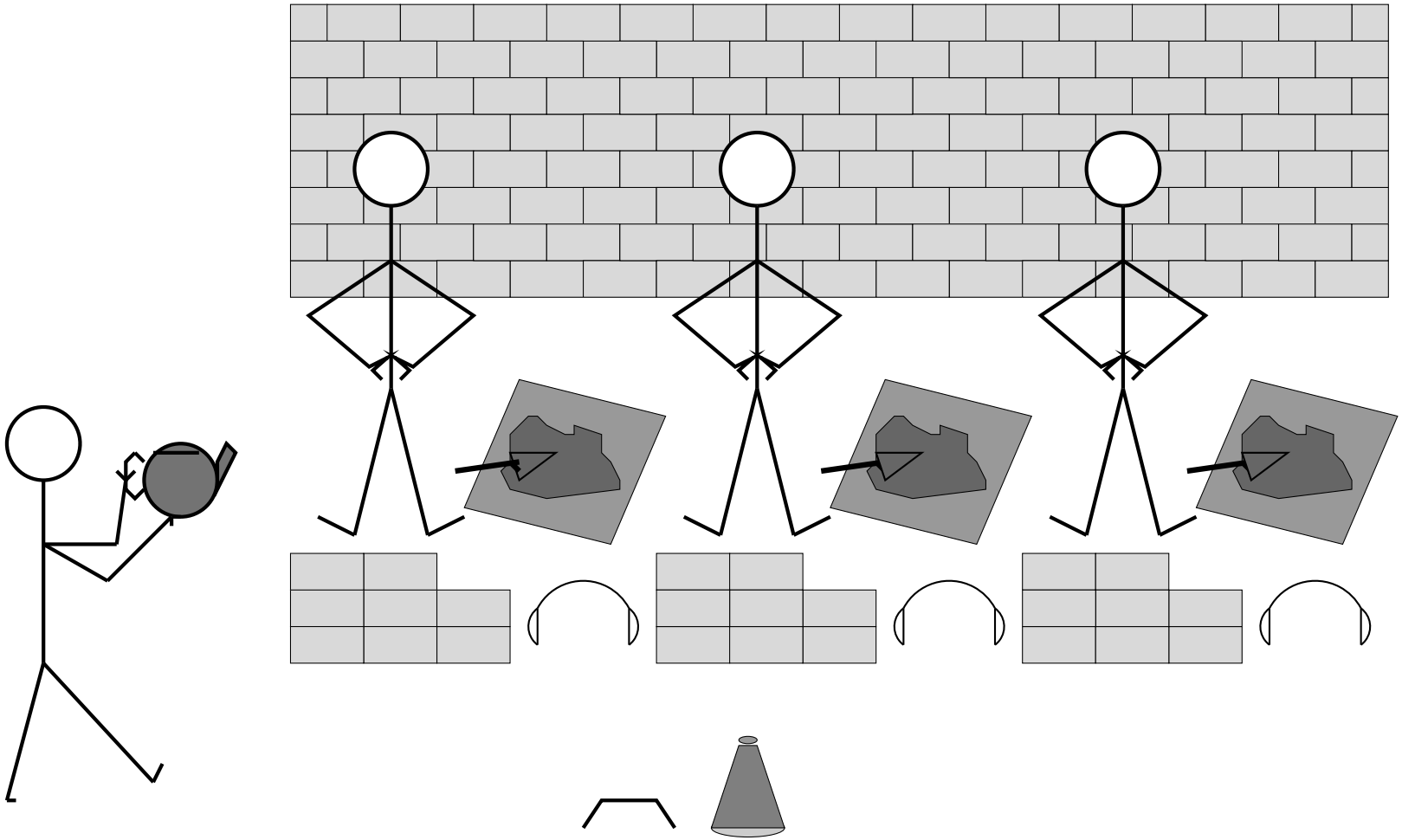
  Note that our slaves will still be idle during sequential tasks.

[2] 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

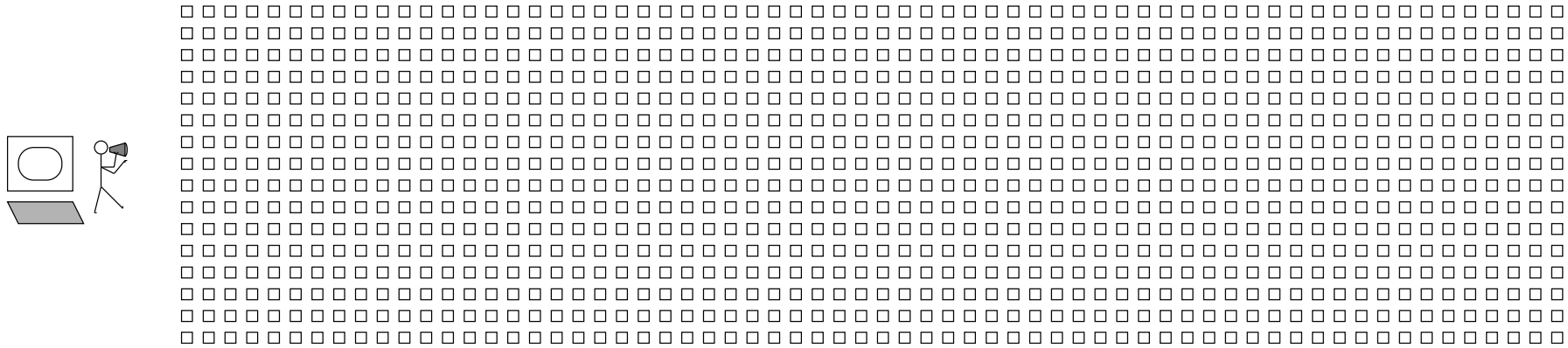


Sequential Processes on SIMD machines



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.

MIMD Parallel Computers

There are are a very large number of parallel machines which bear the label MIMD.

We shall choose a few significant machines for particular attention in this course:

- The Intel Hypercube
  - because it is typical of a large number of machines and we've got one at Chilworth.

- Machines based on the Inmos Transputer
  - because the Transputer is the first microprocessor designed as a building block for parallel machines.

- The CM-5 Connection Machine from Thinking Machines
  - because it represents an interesting view of the future.

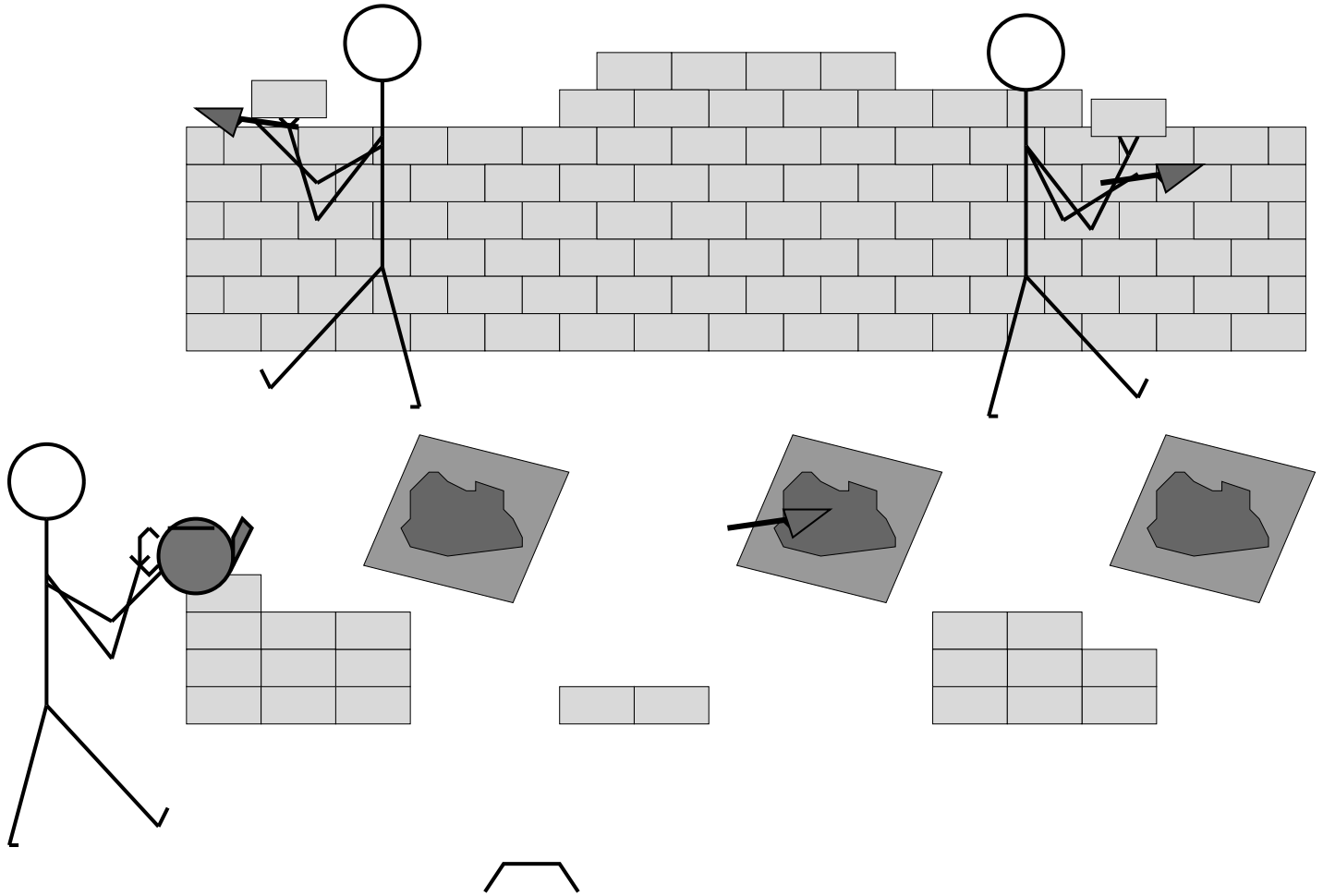
MIMD Parallel Computers

Features of MIMD machines

- MIMD machines typically consist of a number of PEs each of which is a computer in its own right.
- Each PE has a separate instruction stream which it follows independent of the actions of the others.
- In order to communicate and co-operate they may use shared memory locations or have other dedicated hardware.
- One PE may act as master in the co-ordination of tasks, or this co-ordinating role might be distributed amongst a number or all of the PEs.

Now we can look at wall building by 'MIMD Construction Ltd.'.

MIMD Construction Ltd. - *Wall builders Co-operative*



Benefits of MIMD Architectures

- MIMD machines are more flexible.
  - Boundary Effects
    - - End workers behave differently without stopping the central workers.
  - Insufficient Work
    - - 1000 Slaves building a wall 500 bricks wide – 500 carry bricks 500 carry trowels.
  - Sequential Tasks
    - - Making the Tea – done by a single worker while the others build.

  Due to this flexibility, MIMD machines are seen by many as the general purpose parallel computers of the future.
- MIMD machines are often built using standard sequential computer components (e.g. standard microprocessors).

  As the sequential market is much larger than the parallel market these components are well understood and cheap.

Problems with MIMD Architectures

- We must pay for increased flexibility. Each processing element must have an IPU although there will seldom be as many distinct strands of control as there are processors.

- Due to the independence of their PEs, MIMD machines are difficult to co-ordinate. Synchronization and distribution of control information are more complex than in SIMD machines.

- Inter-processor communication is made more difficult because the PEs do not operate in lockstep.

  Frequently we may find that one PE will sit idle waiting for another to send data.

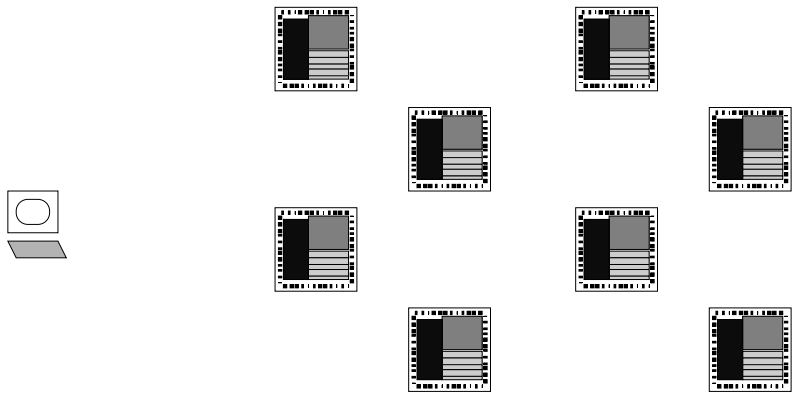
- MIMD machines are very difficult to program.

  Along with multiple instruction streams come new programming concepts. The interaction of instruction streams can lead to deadlock and other such fatal software ailments.

MIMD - Processor size and number

Due to the overheads of an IPU and complex communications hardware, MIMD processing elements are generally of comparable size to SISD processors.

A typical machine will have relatively few, higher complexity processing elements.



The actual numbers of processors will then be limited by the budget and the efficiency of the communications network.

- Transputer machines containing as few as 8 processors with 32 bit floating point capability are common.
- For those with more money the CM-5 machine is potentially configurable to 16384 vector parallel nodes giving a total power of 2 TerraFlops.

MIMD - Exploiting Parallelism

- ## Process Parallelism

  MIMD machines can exploit *Process Parallelism* since the task performed by one PE need not be related to that perfored by another.

- ## Data Parallelism

  MIMD machines can exploit *Data Parallelism* since it is a sub-set of *Process Parallelism*. Nevertheless, many MIMD machines include elements of SIMD architecture to improve performance on *Data Parallel* tasks.