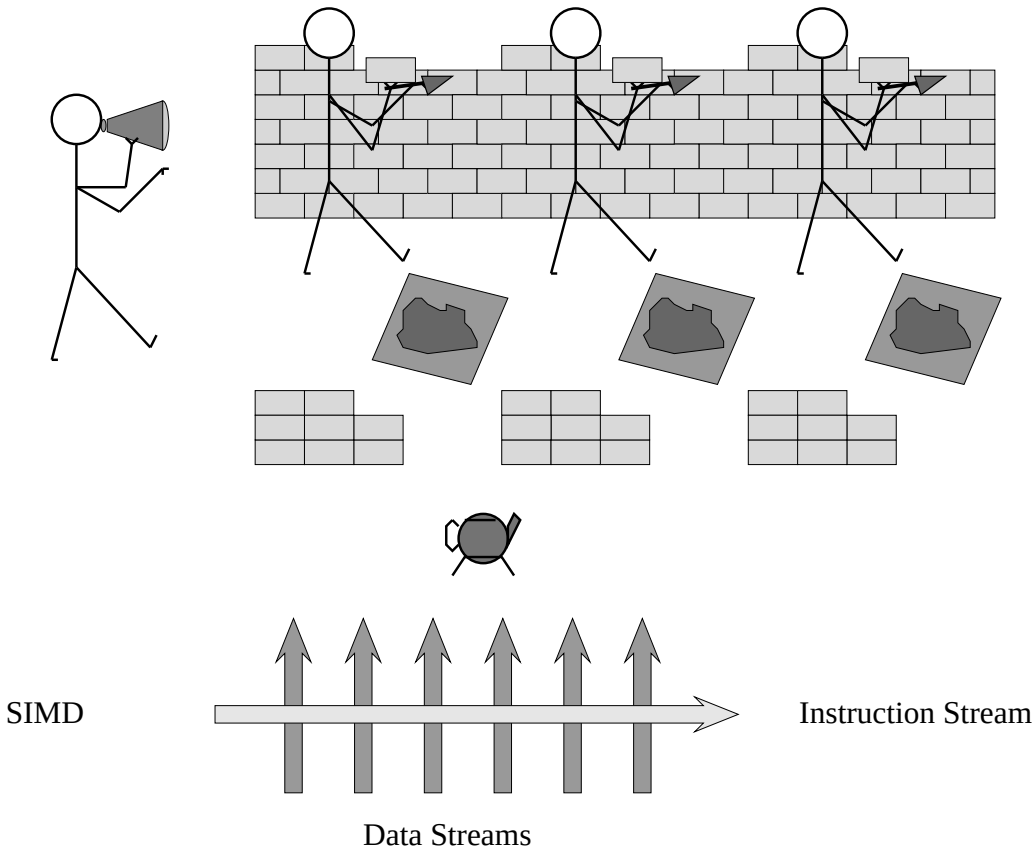
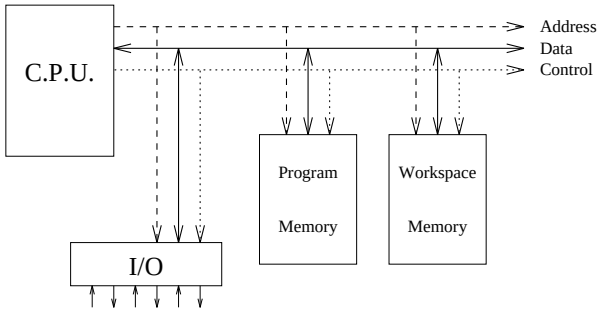
SIMD Computers



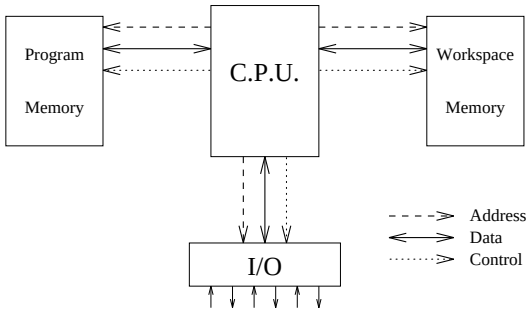
How might you go about designing a SIMD computer?

SIMD - Build your own SIMD Computer

- First take one SISD Computer:
  - Princeton architecture



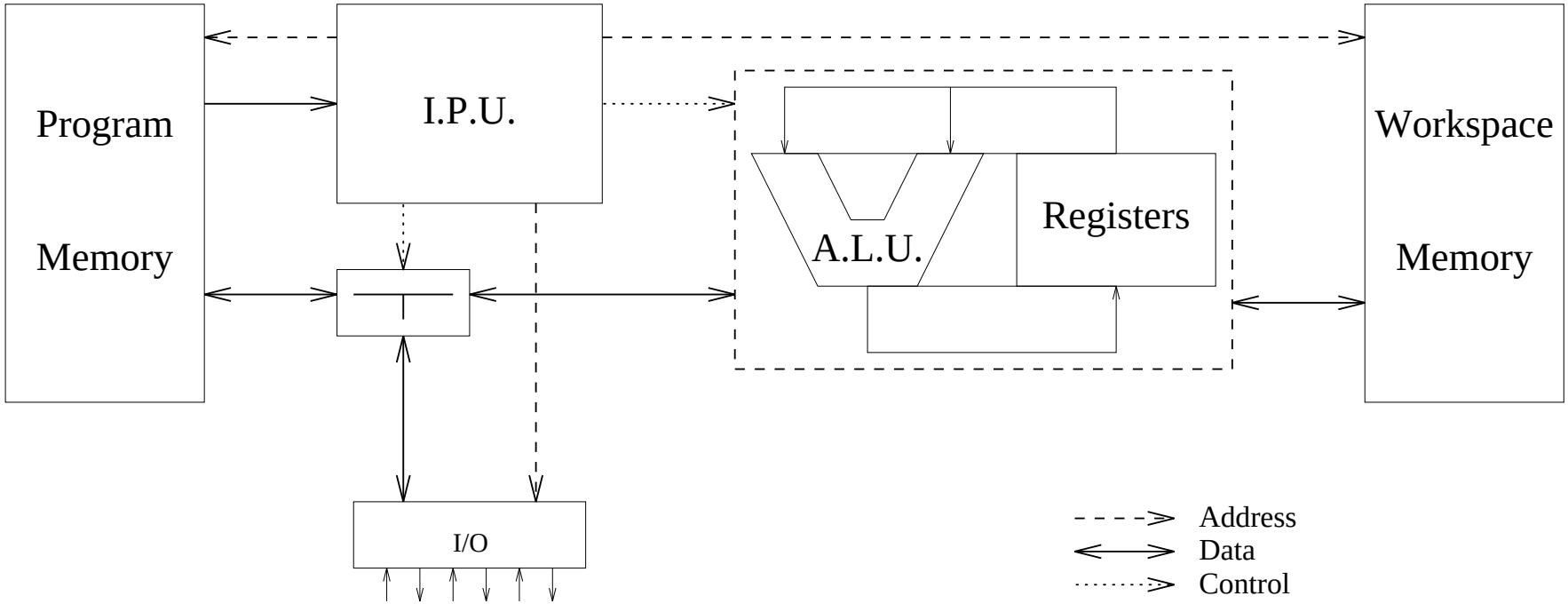
  - Harvard architecture



We will take the Harvard machine.

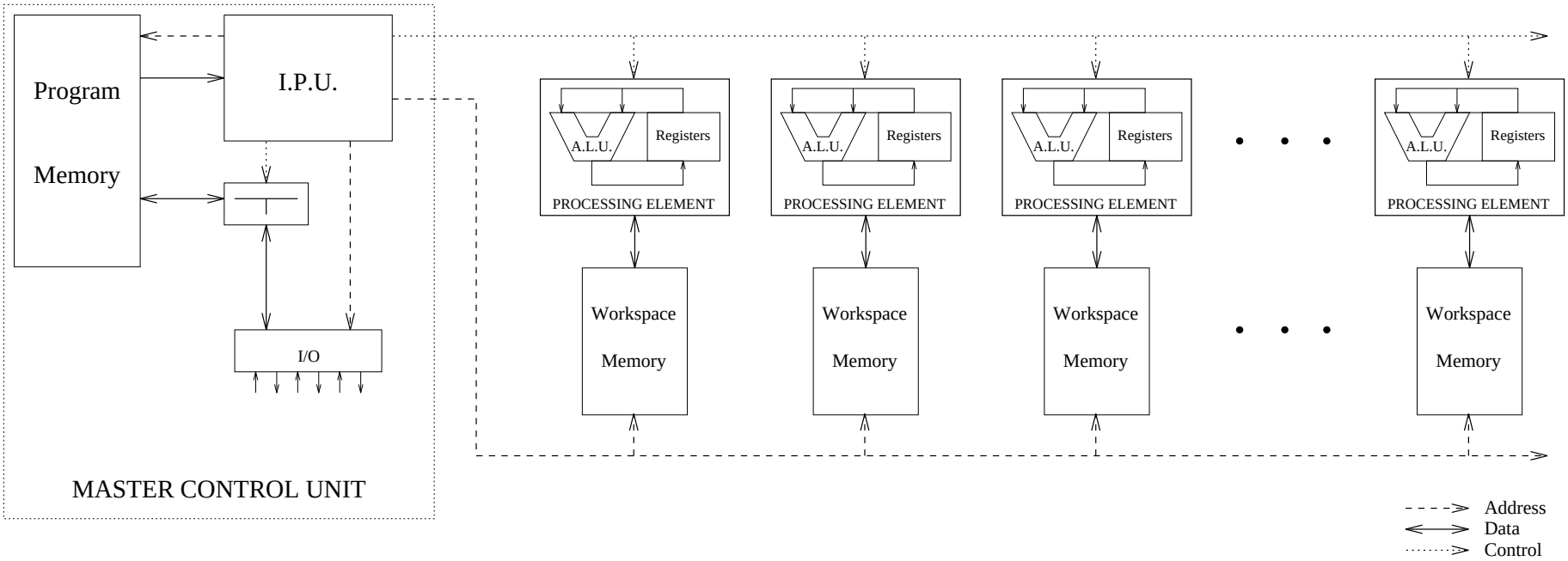
SIMD - Build your own SIMD Computer

- Second split your SISD Computer:



SIMD - Build your own SIMD Computer

- We shall build a *Replicated Parallel Computer*
- **Replicated ALU** and **Distributed Memory**



- Data Communications?

  In splitting and replicating our machine we have lost part of our data path. We must replace lost connectivity and provide new connectivity to cope with multiple PEs:

  – P.E. to P.E. communications

  – M.C.U. to P.E. communications

  – I/O to P.E. communications[1]

[1]This may be included within M.C.U. to P.E. communications or it may be separate.

SIMD - Build your own SIMD Computer

- P.E. to P.E. communications

  We shall optimize our communications for real world modelling, in particular for *Finite Element Analysis*.

  - 1D - Conduction of heat in a bar.
  - 2D - Airflow around a wing - Beam stress - Simple Weather Modeling.
  - 3D - Airflow around a plane - Complex Weather Modeling - Magnetic field problems.

  1D and 2D problems are usually simplifications of 3D problems which would otherwise be too complex to solve.

- We will build an *Array Processor*.

How many dimensions?

Hardware Considerations

We must balance connectivity with wiring cost!

Wiring is very expensive -

On Chip

- 10% - 90% of chip area.
- 10% corresponds to regular 2D structures like memory.
- 90% corresponds to irregular structures - random logic.

Off Chip

- Chip Pin count
- PCB layout - regular 1D and 2D structures are best.

We shall go for a 2D structure

We shall connect our PEs as a 2D Grid (Using Nearest Neighbour Connectivity - Essential for Finite Element Analysis).

Allow for 1D problems

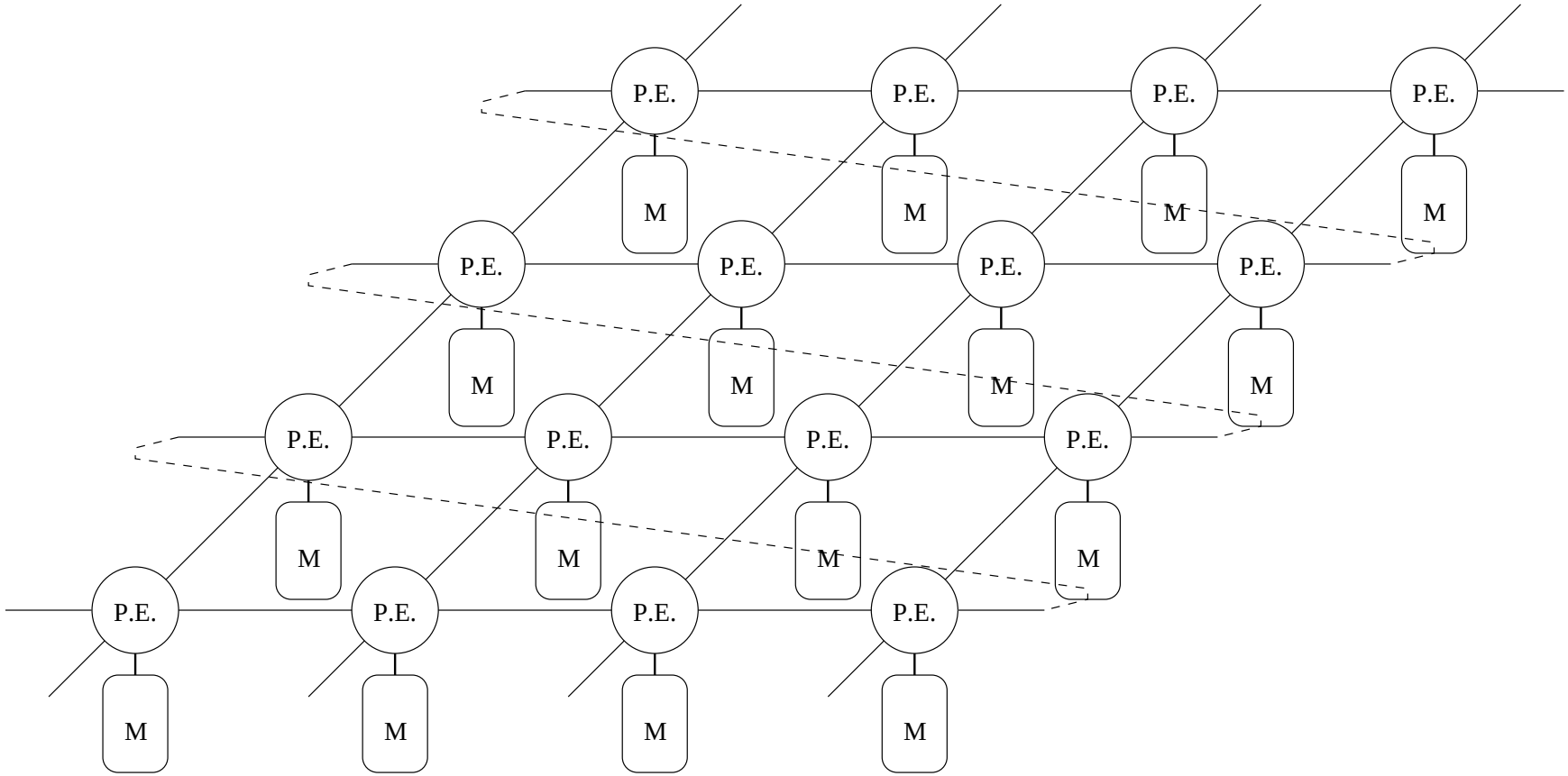
For little extra cost we can allow our grid to be reconfigurable for full 1D connectivity.

Allow for 3D problems

For 3D problems we must use the memory as the third dimension. We are relying on the problems being bigger than the array in order to maintain efficient use of the PEs in a 3D problem.

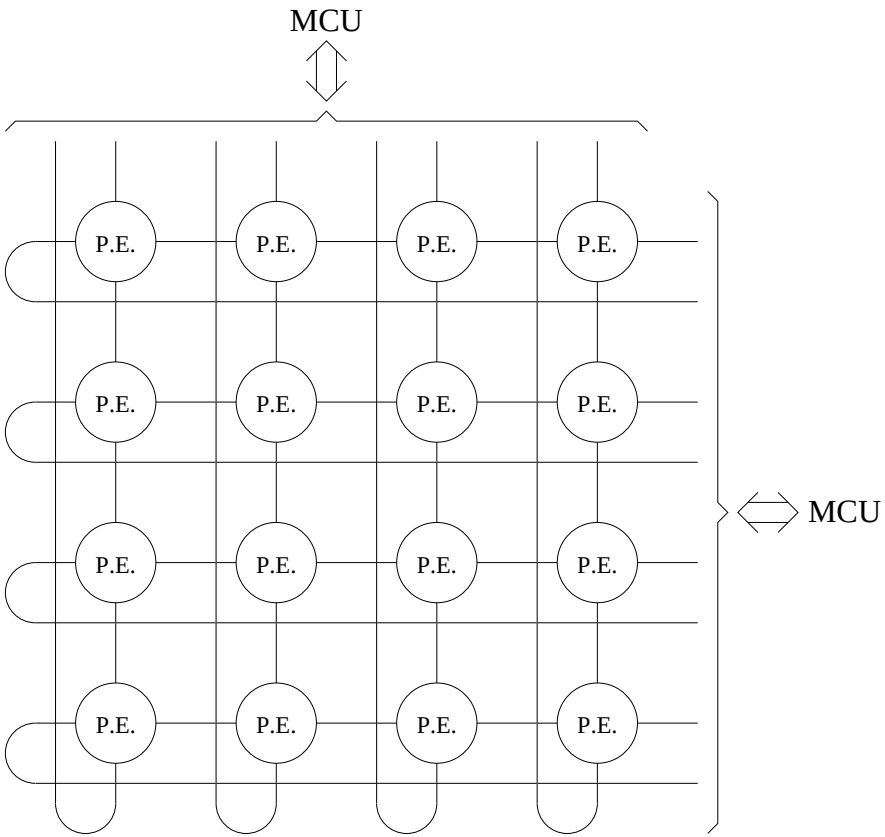
SIMD - Build your own SIMD Computer

Array Connectivity



SIMD - Build your own SIMD Computer

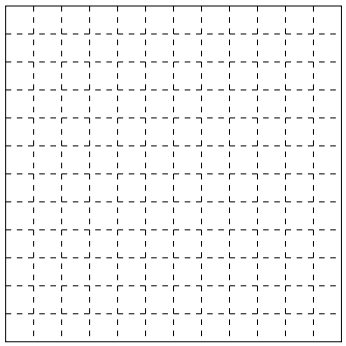
- M.C.U. to P.E. communications



Note that the MCU is now responsible for reconfiguration for 1D operation.

SIMD

Where $PE\ array\ size \neq Data\ array\ size$ we must perform a data mapping.

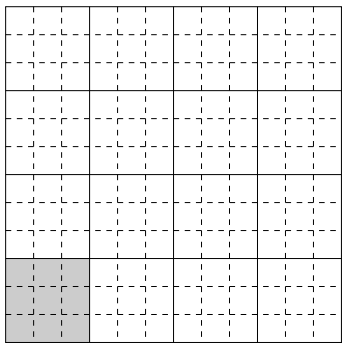


12x12 Element Array

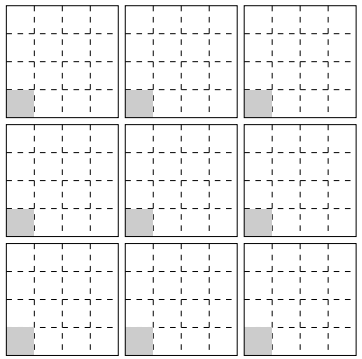
for

4x4 PE Array Processor

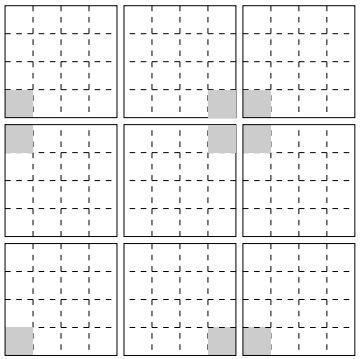
Data allocated to single PE



Crinkle Mapping



Sheet Mapping (A)

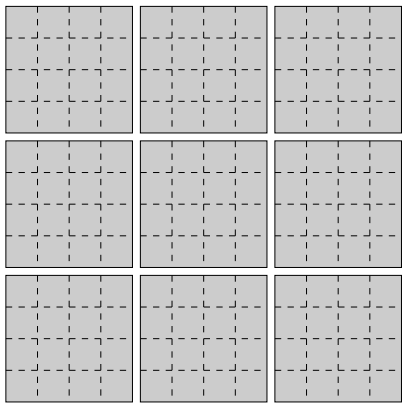


Sheet Mapping (B)

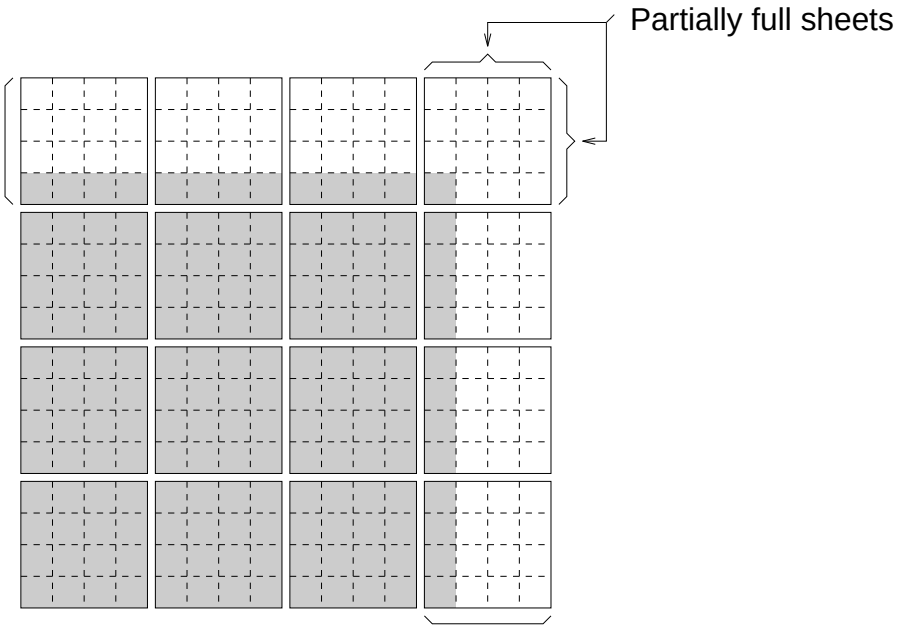
SIMD

Consider mapping efficiency for a 2D array.

Sheet Mapping for
4x4 PE Array Processor



12x12 Element Data Array

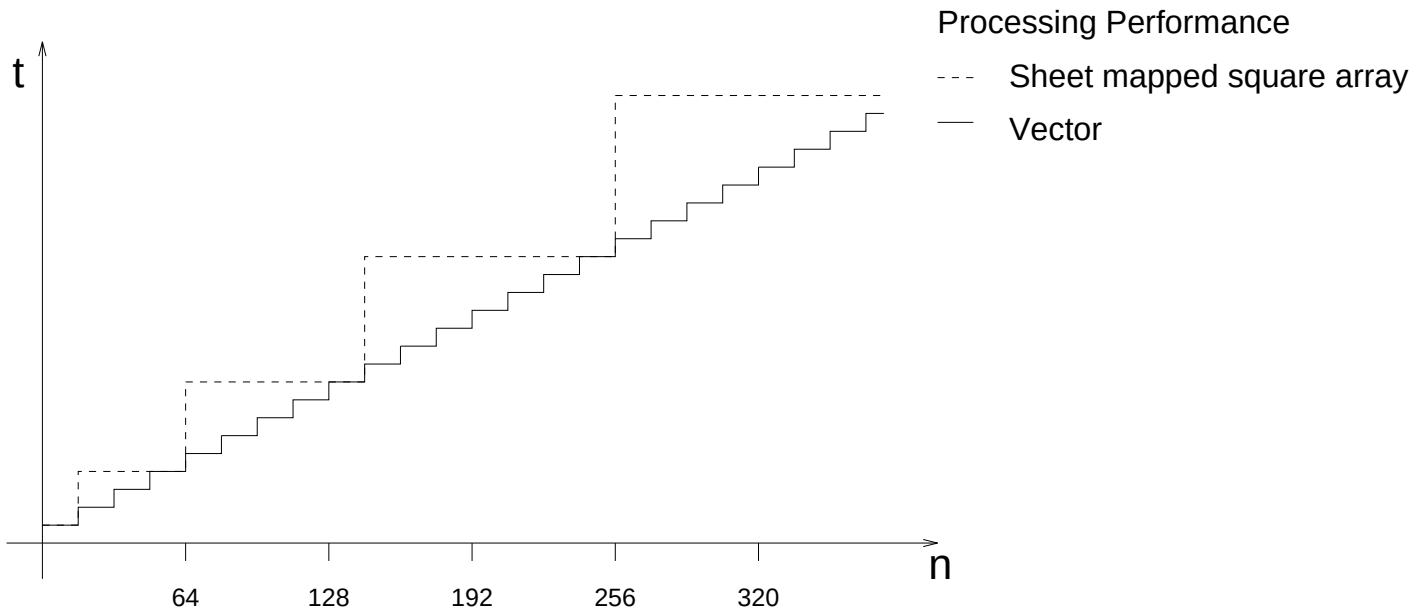


13x13 Element Data Array

Here we have 16 sheets (including 7 partially full) for 169 elements. With a 1D mapping we would need only 11 sheets (including 1 partially full).

SIMD - Array Processor Performance

Consider performance graphs for our array processor:

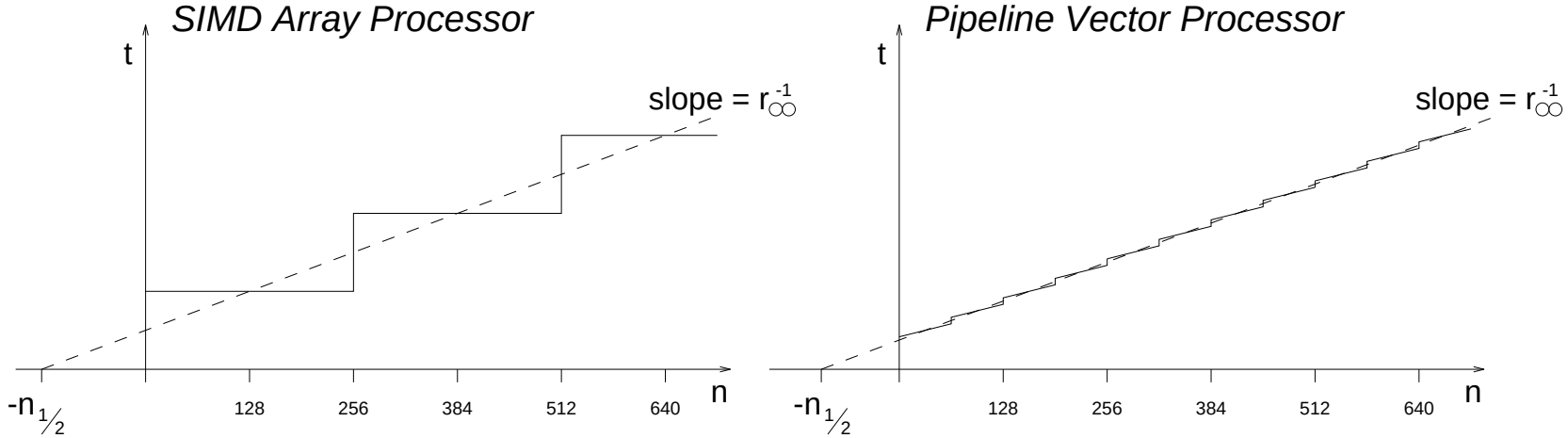


We see that computation time increases only as an increase in vector length causes an increase in the number of sheets required for the mapping.

Note also the graph for a square array which illustrates the potential for wasted resources given a 2D to 2D mapping.

SIMD 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]



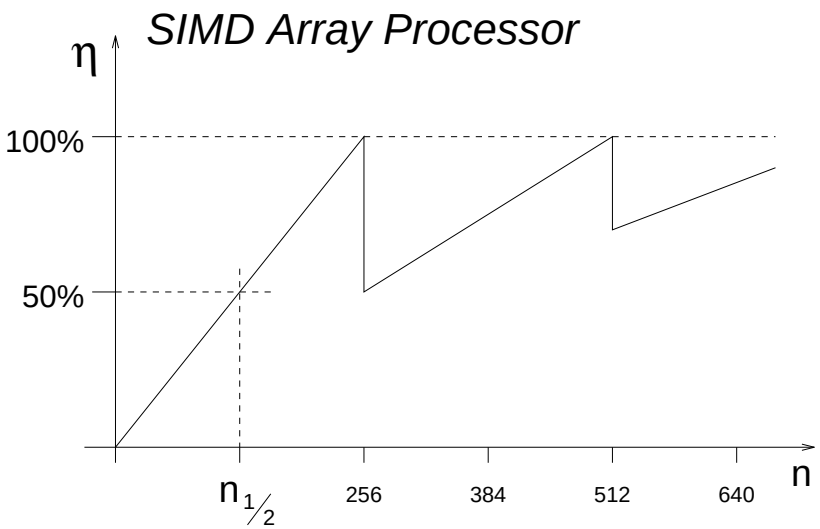
We see that $n_{1/2} = Array_size/2$ for array processor, while it is more determined by pipeline length and startup time for the vector processor.

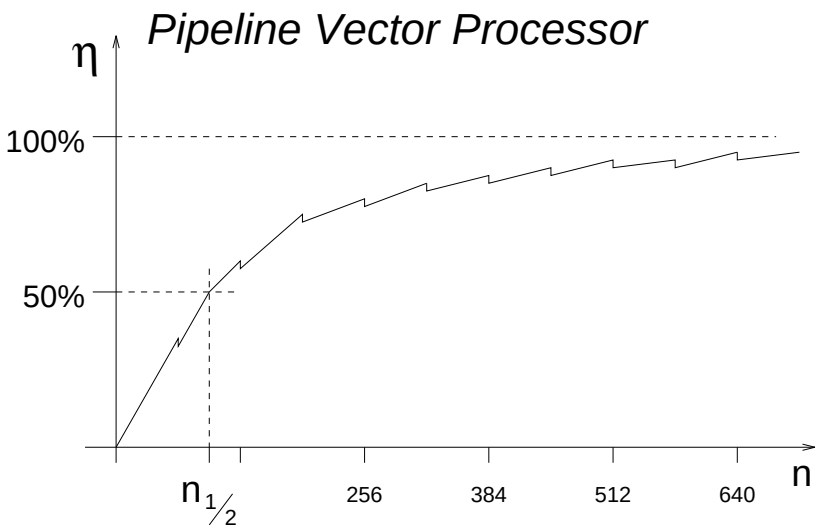
[2]The array processor typically has greater number of less powerful nodes, as shown.

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.