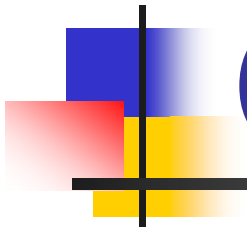


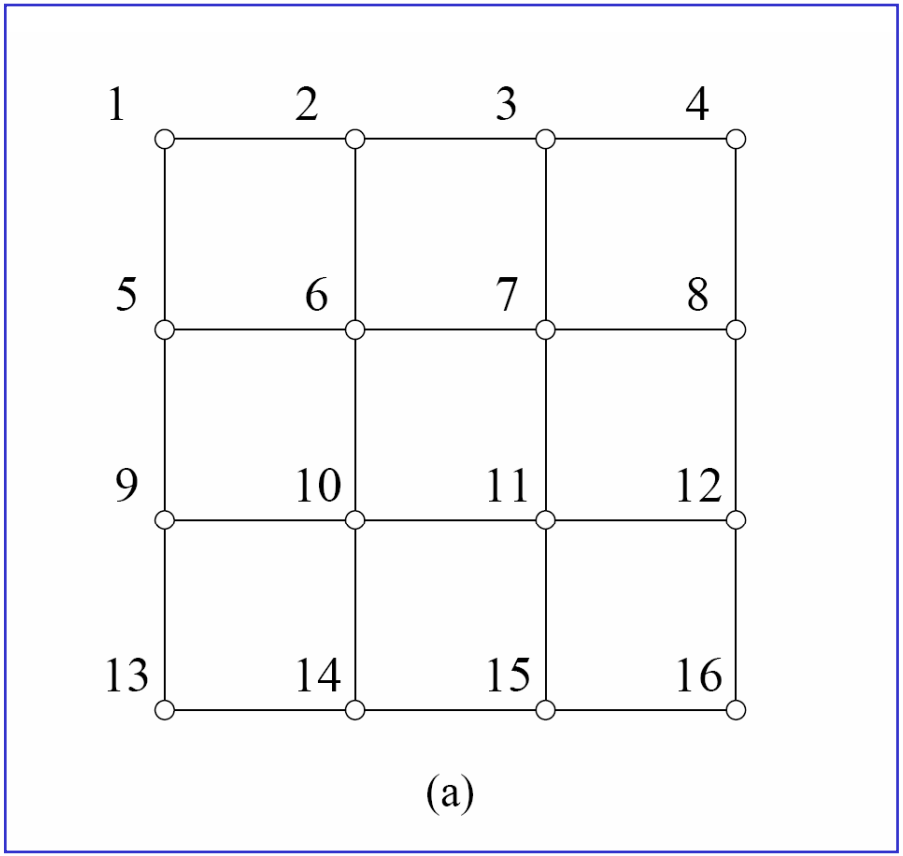
Process-Processor Mapping (2.7)

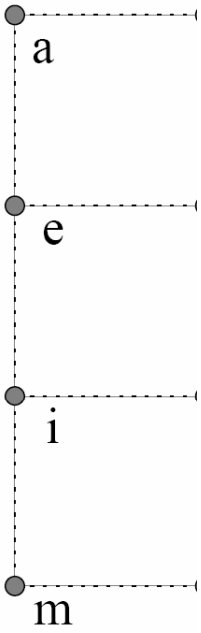
Alexandre David
B2-206

Example

Underlying architecture (physical network).
Processors.

Processes and their interactions.

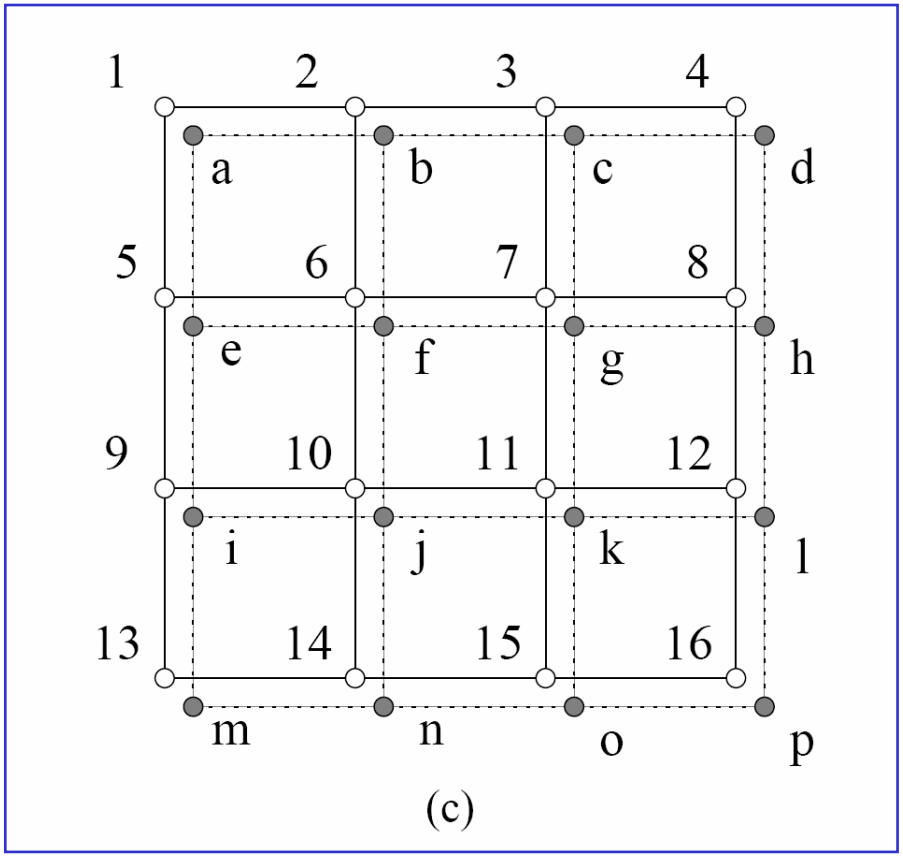


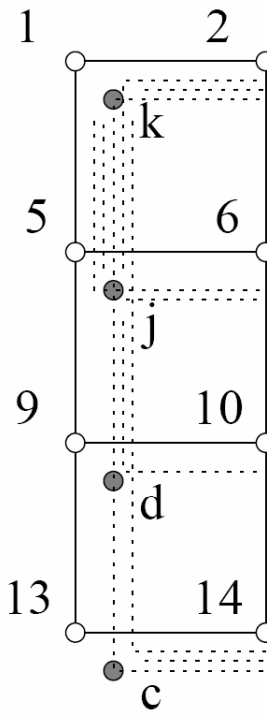


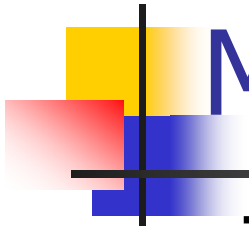
Example

Intuitive mapping.

Random mapping and congestion.

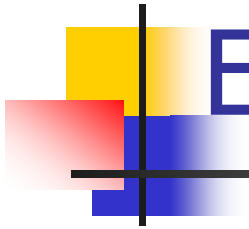






Mapping Techniques For Graphs

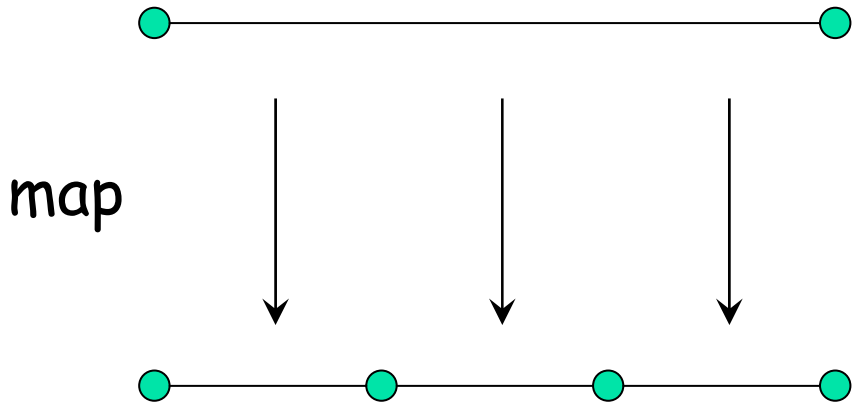
- Topology embedding:
  - Embed a communication pattern into a given interconnection topology.
    Hypercube in a 2-D mesh?
    2-D mesh in a hypercube?
- Why?
  - Cost.
  - Design an algorithm for a topology but you port it to another.



Embedding Metrics

- Map a graph *G(V,E)* into *G'(V',E')*.
  - Dilation: Maximum number of links of E' an edge of E is mapped onto.
  - Expansion: ratio |V'|/|V|.
  - Congestion: Maximum number of edges of E mapped on a single link of E'.

Dilation & Expansion

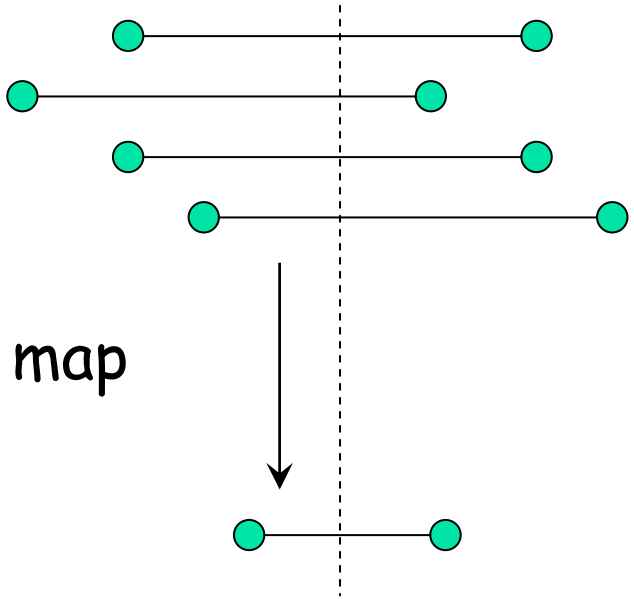


V,E
Source

V',E'
Target

Dilation: 3.
Expansion: 4/2 = 2.

Congestion



V,E
Source

V',E'
Target

Congestion: 4.

Embedding a Linear Array Into a Hypercube

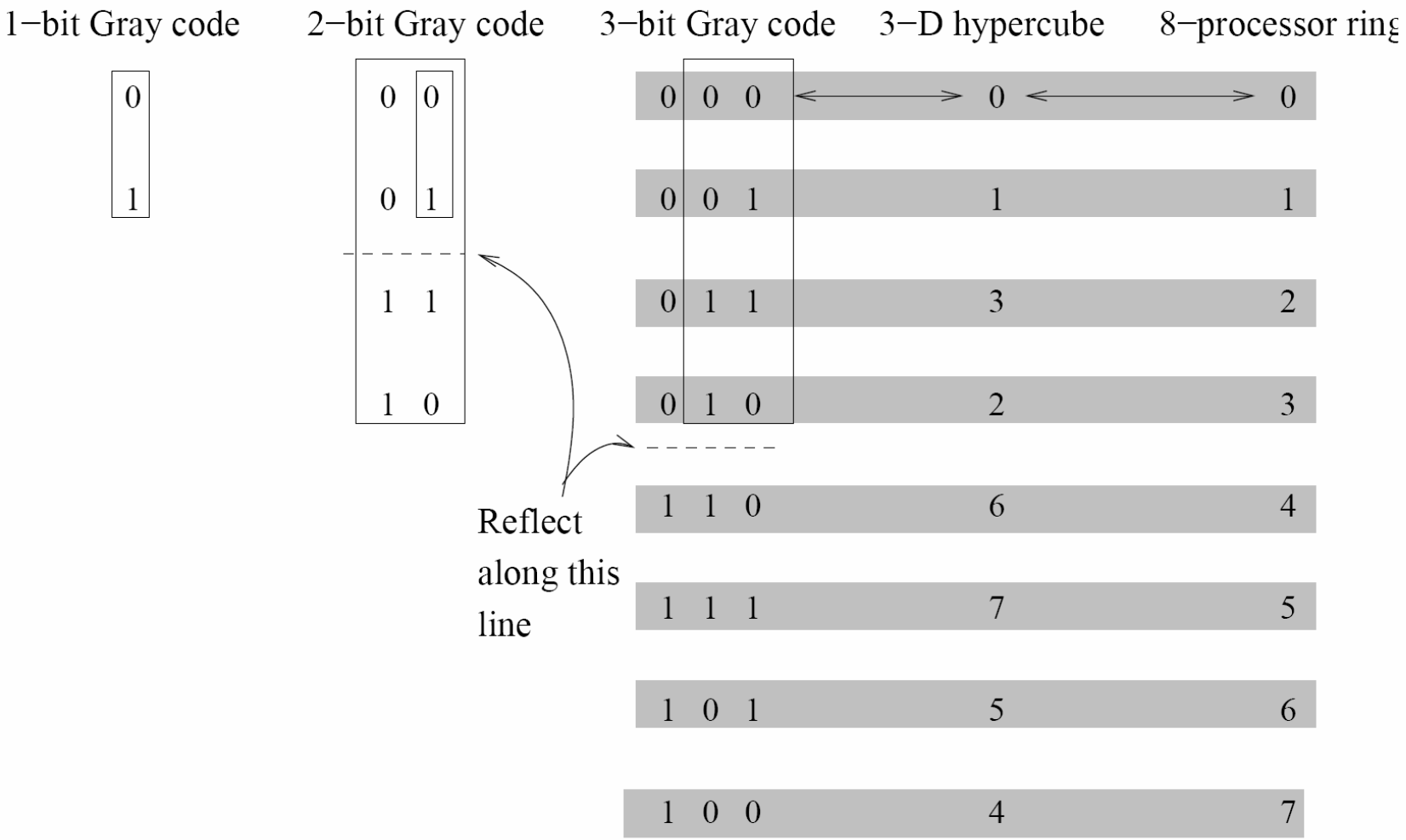
- Map a linear array (or ring) of 2^d nodes into a d-dimensional hypercube.
- How would you do it?
- Gray code function:

$$G(0,1) = 0$$

$$G(1,1) = 1$$

$$G(i,x+1) = \begin{cases} G(i,x), & i < 2^x \\ 2^x + G(2^{x+1} - 1 - i, x), & i \geq 2^x \end{cases}$$

Gray Code


1−bit Gray code
2−bit Gray code
3−bit Gray code
3−D hypercube
8−processor ring
0
1
0 0
0 1
1 1
1 0
0 0 0
0 0 1
0 1 1
0 1 0
1 1 0
1 1 1
1 0 1
1 0 0
0
1
3
2
6
7
5
4
0
1
2
3
4
5
6
7
Reflect
along this
line

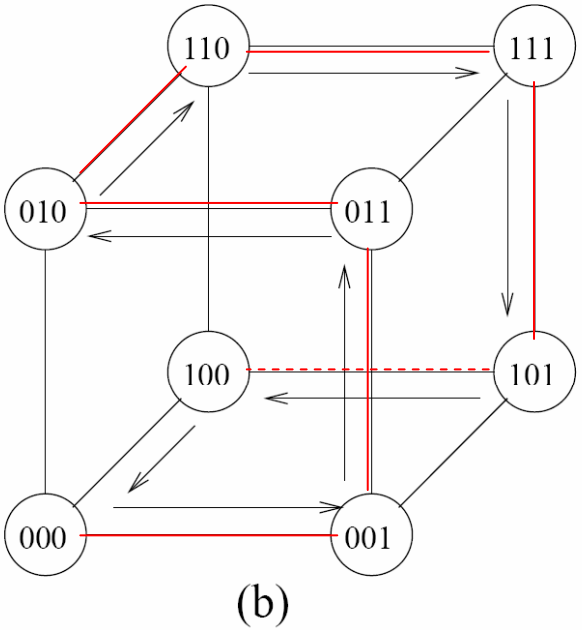


Figure 2.30 (a) A three-bit reflected Gray code ring; and (b) its embedding into a three-dimensional hypercube.

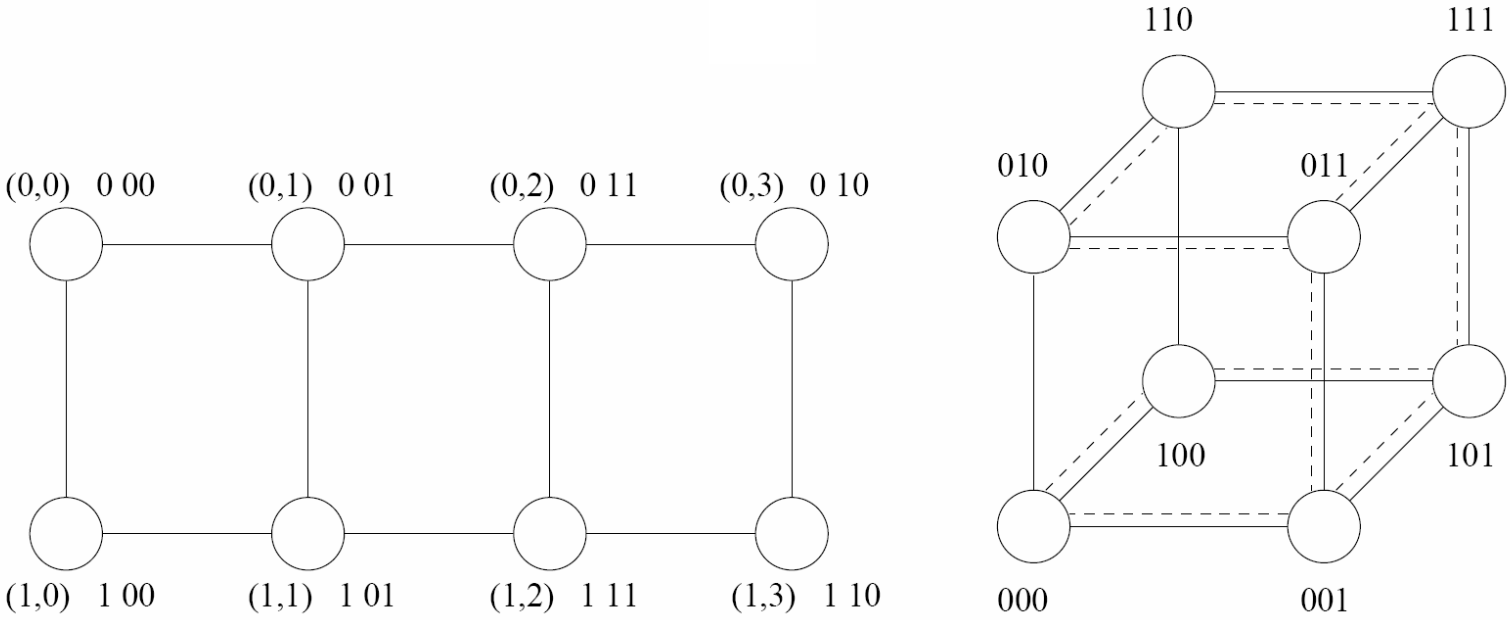
Gray Code Mapping

Gray Code Mapping cont.

- *G(i,d)* : i^{th} entry in sequence of *d* bits.
- Adjoining entries *G(i,d)* and *G(i+1,d)* differ at only one bit.
  - Like hypercubes -> direct link for these nodes.
- Dilation?
- Congestion?

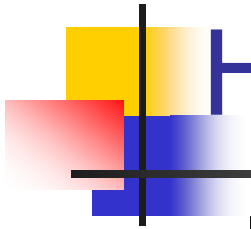
Embedding a Mesh into a Hypercube

- Map a $2^r \times 2^s$ wraparound mesh into a r+s dimension hypercube.
- How?
- Map *(i,j)* to *G(i,r-1)||G(j,s-1)*.
  - Extension of previous coding.



2x4 mesh into a 3-D hypercube

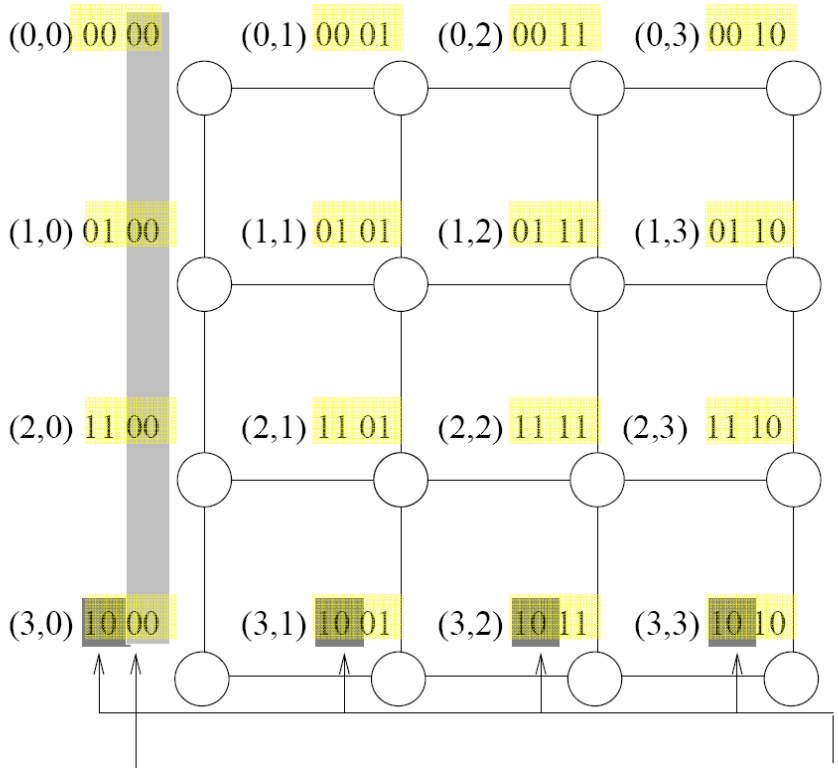
Embedding a Mesh into a Hypercube



- Properties
  - Dilation & congestion 1 as before.
  - All nodes in the **same row** (mesh) are mapped to hypercube nodes with ***r* identical most significant bits**.
  - Similarly for columns: s identical least significant bits.
  - What it means: They are mapped on a sub-cube!

Sub-Cube Property (4x4)

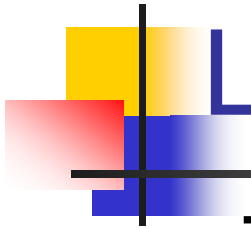
Gray codes



Processors in a column have identical two least–significant bits

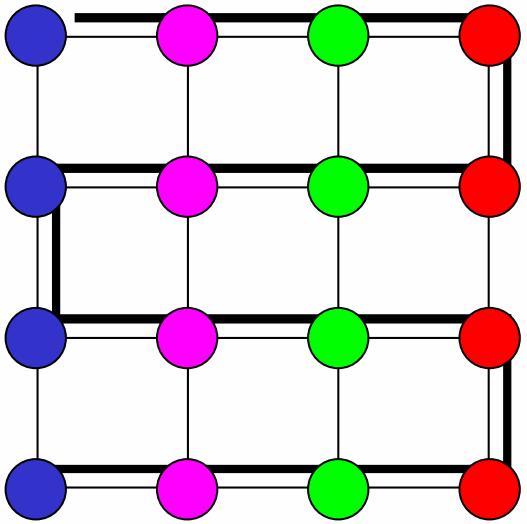
Processors in a row have identical two most–significant bits

Embedding of a Mesh Into a Linear Array



- This time denser into sparser.
- 2-D mesh has *2p* links and an array has *p* links.
  - There must be congestion!
  - Optimal mapping: in terms of congestion.

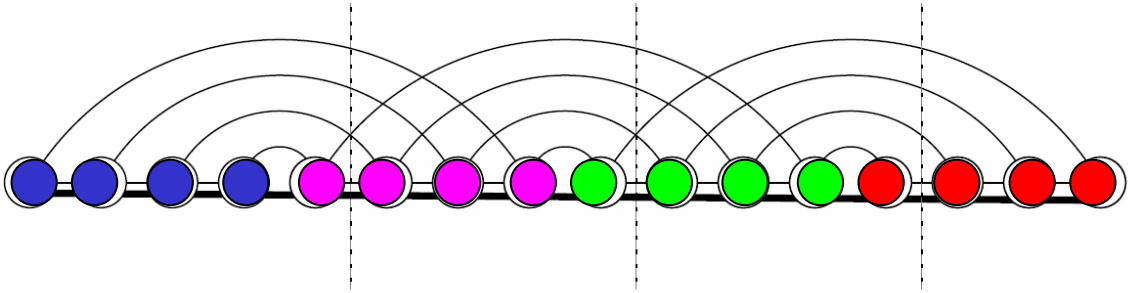
Easy: Linear Array Into Mesh



(a) Mapping a linear array into a 2D mesh (congestion 1).

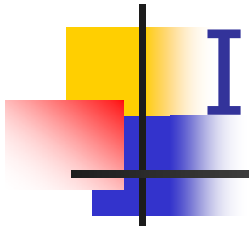
Mesh Into Linear Array

Congestion: 5.



(b) Inverting the mapping − mapping a 2D mesh into a linear array (congestion 5)

Figure 2.32 (a) Embedding a 16 node linear array into a 2-D mesh; and (b) the inverse of the mapping. Solid lines correspond to links in the linear array and normal lines to links in the mesh.

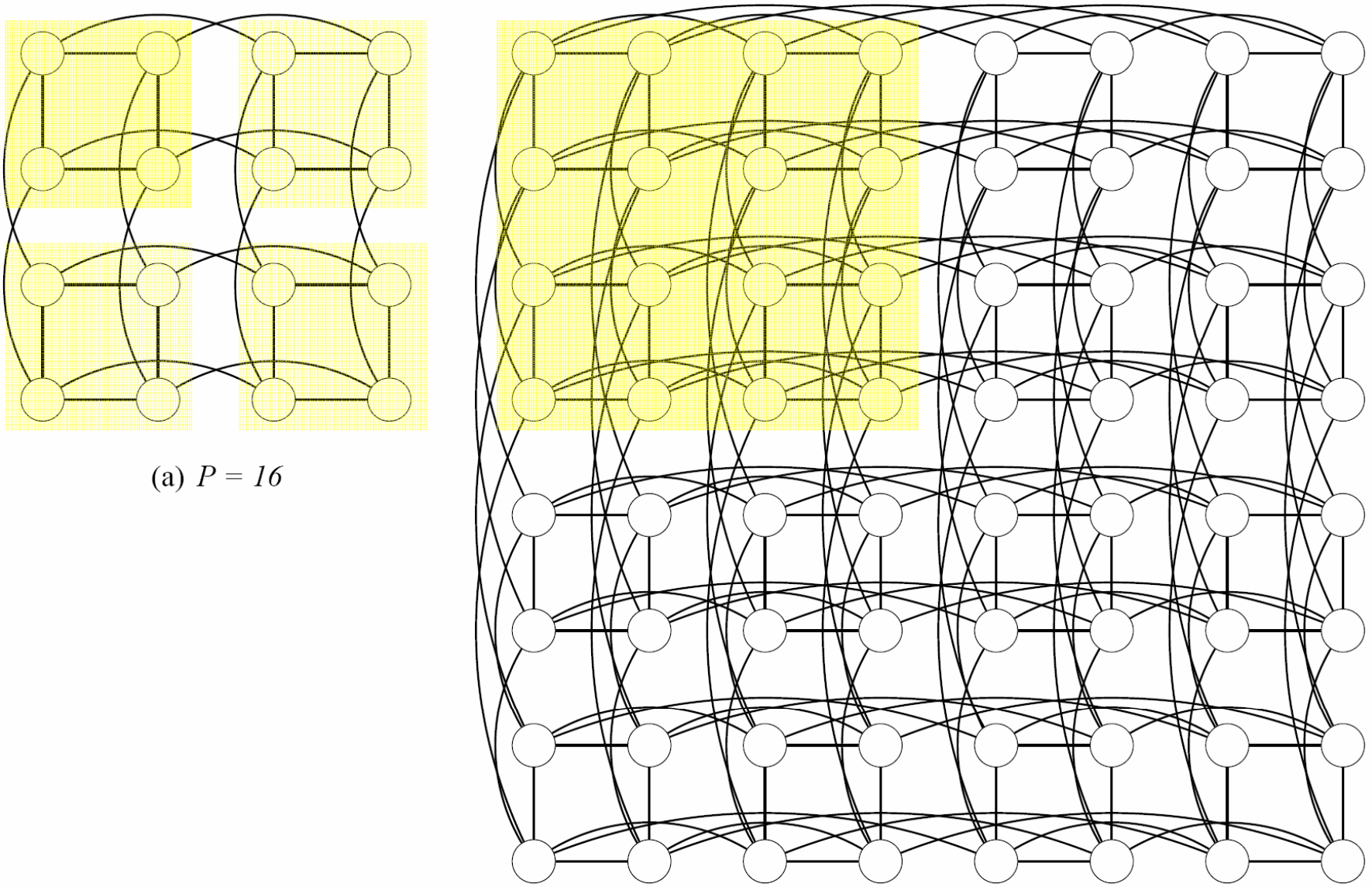


Is It Optimal?

- Bisection of
  - 2-D mesh is sqrt(p).
  - linear array is 1.
- 2-D -> linear array has congestion *r*.
  - Cut in half linear array: cut 1 link, but cut no more than *r* mapped mesh links.
  - Lower bound: $r \geq sqrt(p)$.

Hypercube Into a 2-D Mesh

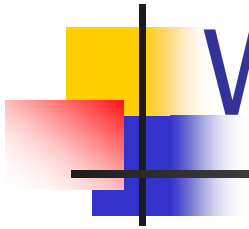
- Denser into sparser again (in terms of links).
- *p* even power of 2.
- *d*= log *p* dimension.
- d/2 least (most) significant bits define sub-cubes of *sqrt(p)* nodes.
- Row/column $\leftrightarrow$ sub-cube, inverse of hybercube to 2-D mesh mapping.



(a) $P = 16$

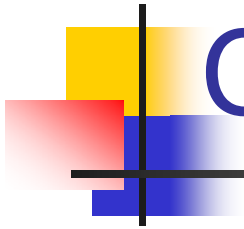
(b) $P = 32$

Figure 2.33 Embedding a hypercube into a 2-D mesh.



What Is The Point?

- Possible to map denser into sparser:
  - Map (expensive) logical topology into (cheaper) physical hardware!
  - Mesh with links faster by *sqrt(p)/2* than hypercube links has same performance!



Cost-Performance

- Read 2.7.2.
- Remember that 2-D mesh is better in terms of performance/cost.