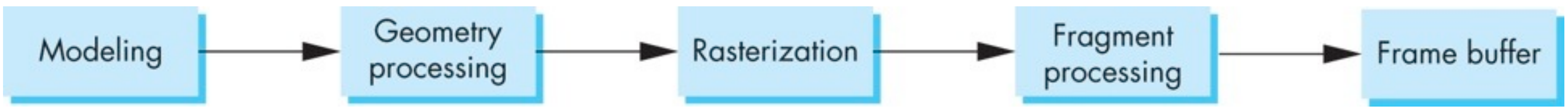
CSSE 351
Computer Graphics

Clipping

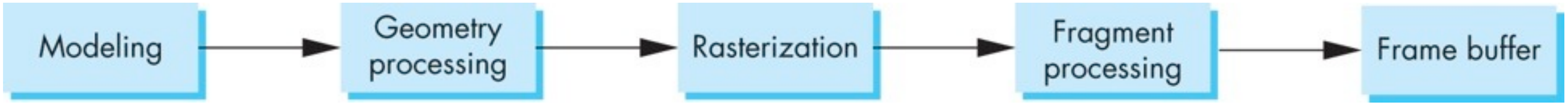
Session schedule

- Rendering pipeline
- 2D clipping
- Triangle clipping
- 3D clipping

Render pipeline

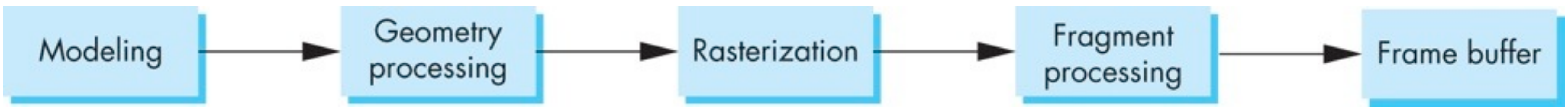


Render pipeline



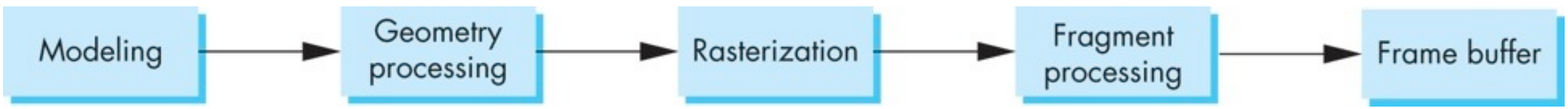
- Modeling
  - Creation of objects in 3D
  - Send to GPU
  - Can be done in shaders also! (geometry/tesselation shader)

Render pipeline



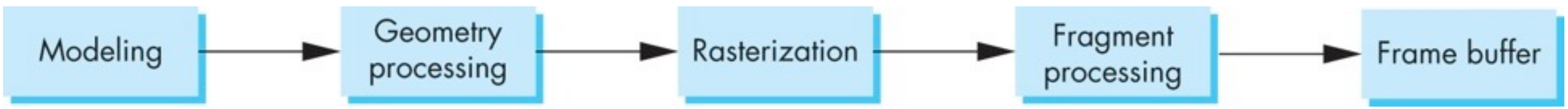
- Geometry processing
  - Apply transforms (view, projection)
    - Vertex shader
  - Clip against view volume
  - Homogenize coordinates

Render pipeline



- Rasterization
  - Interpolate over objects
    - Take discrete samples
  - Called scan conversion
  - Convert to window coordinates

Render pipeline



- Fragments
  - Compute color
  - Compose fragments
  - Final depth sorting
  - Output to framebuffer!

OpenGL Clipping

- Start in 2D, extend to 3D
- Can clip in any coordinate frame
  - OpenGL clips in ‘Clip space’
  - Just before homogenization and NDC

OpenGL Clipping

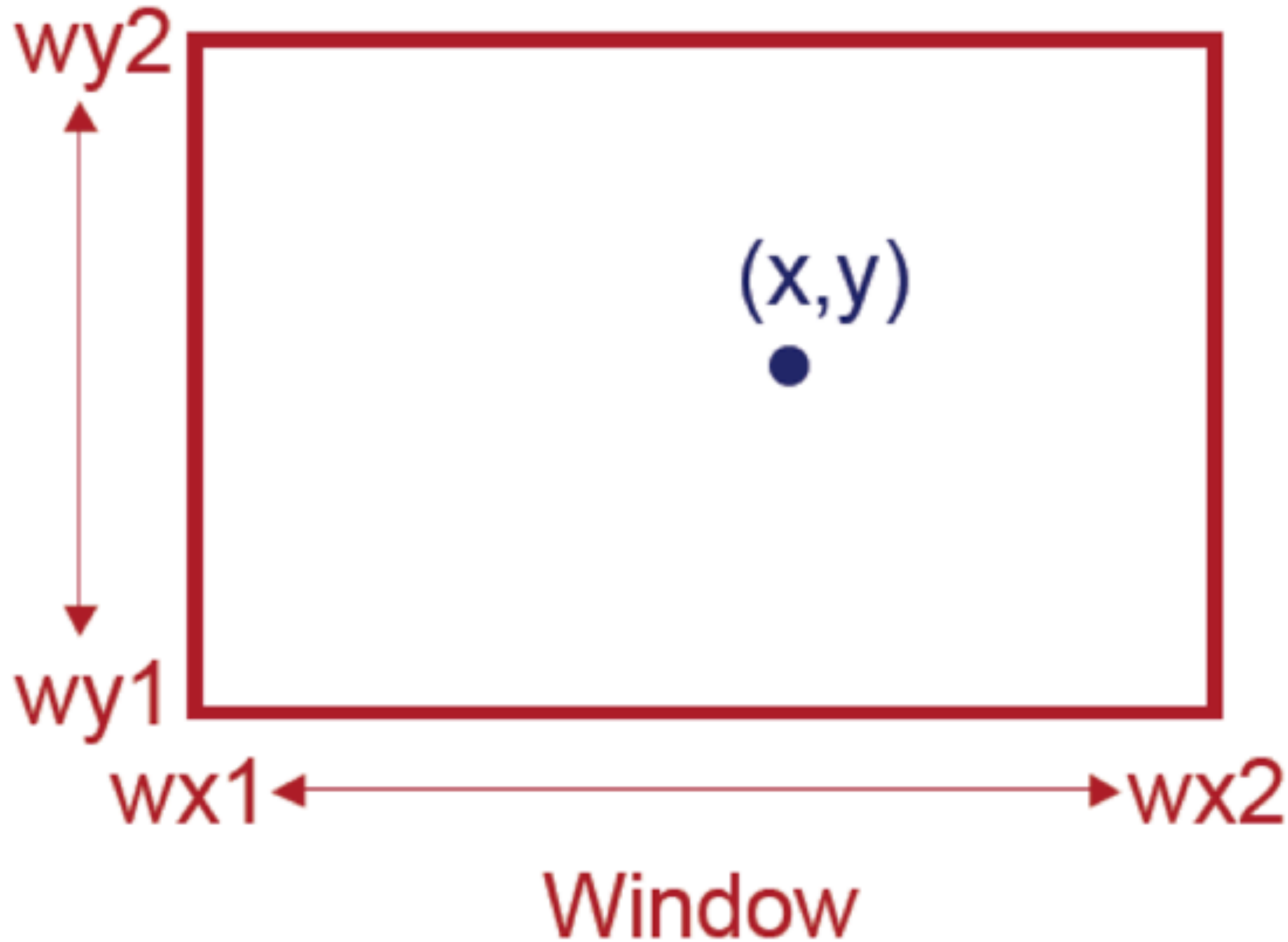
- Camera takes vertices to view/camera space
- Projection takes vertices to clip space
- In clip space
  - Primitives are clipped
  - w is homogenized
- Result is Normalized Device Coords.

2D clipping

- Clip against viewport or view window
- Viewport defined by
  - x min, x max
  - y min, y max

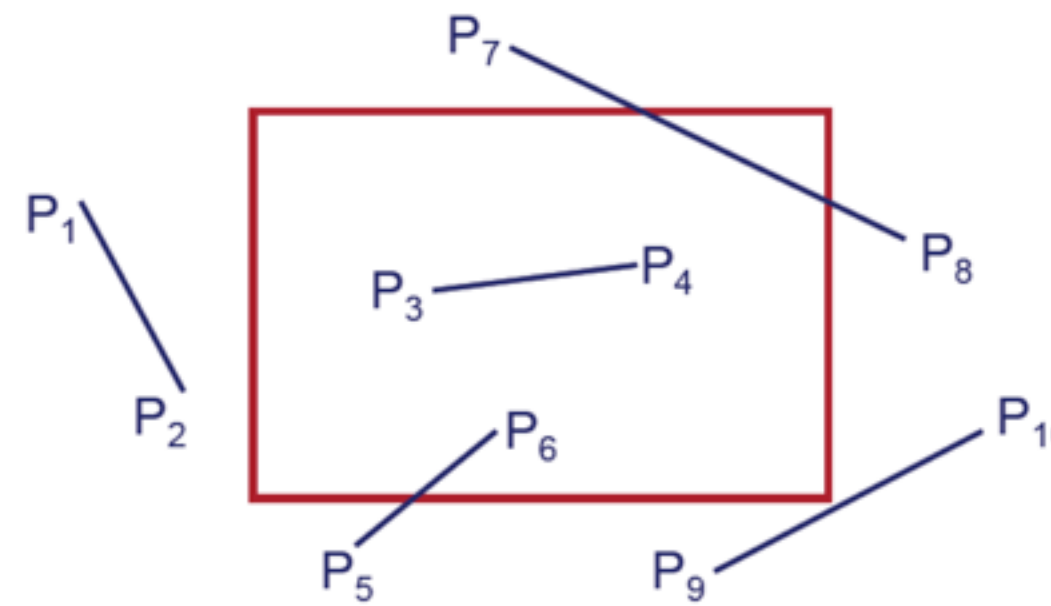
2D clipping

- Clip vertex (x,y) against view window

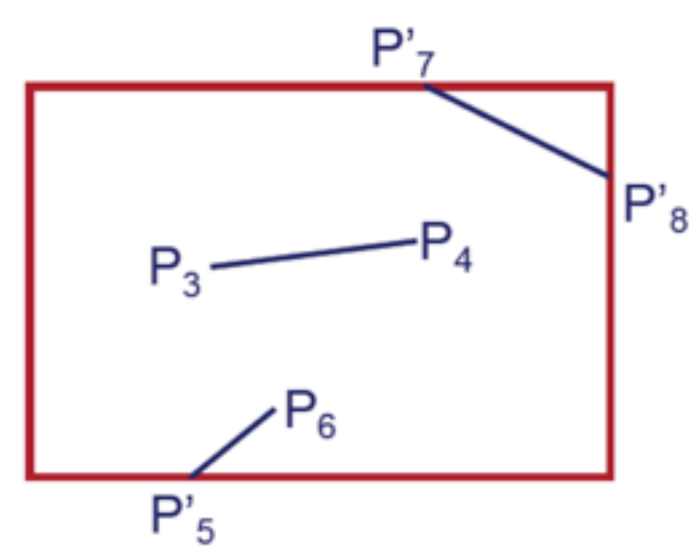


2D clipping

- Clipping line (x1, y1) (x2, y2)
  - More complex
  - 2 points to check



- Can result in new points
  - New line segments

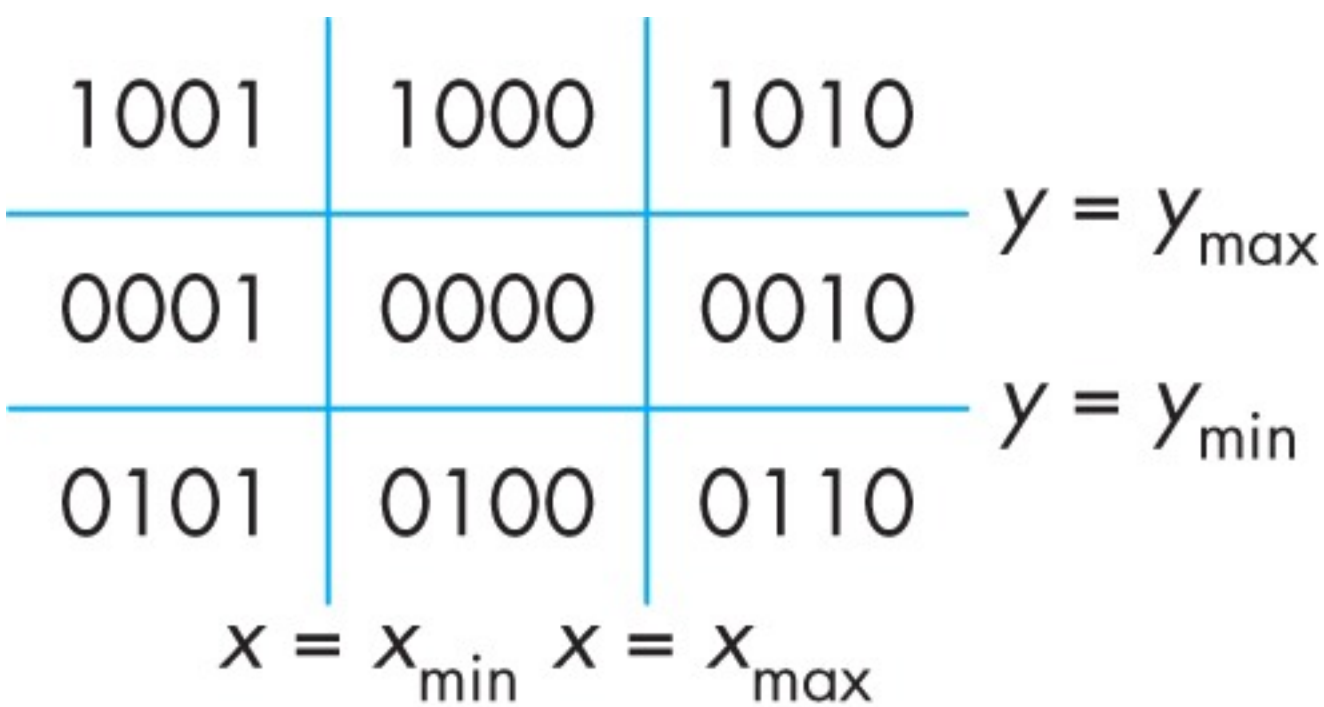


Cohen-Sutherland clipping

- Divide 2D space into 9 regions
- Assign each region ID
- Compute each point's region ID (outcode)
- Compare outcode to determine clipping

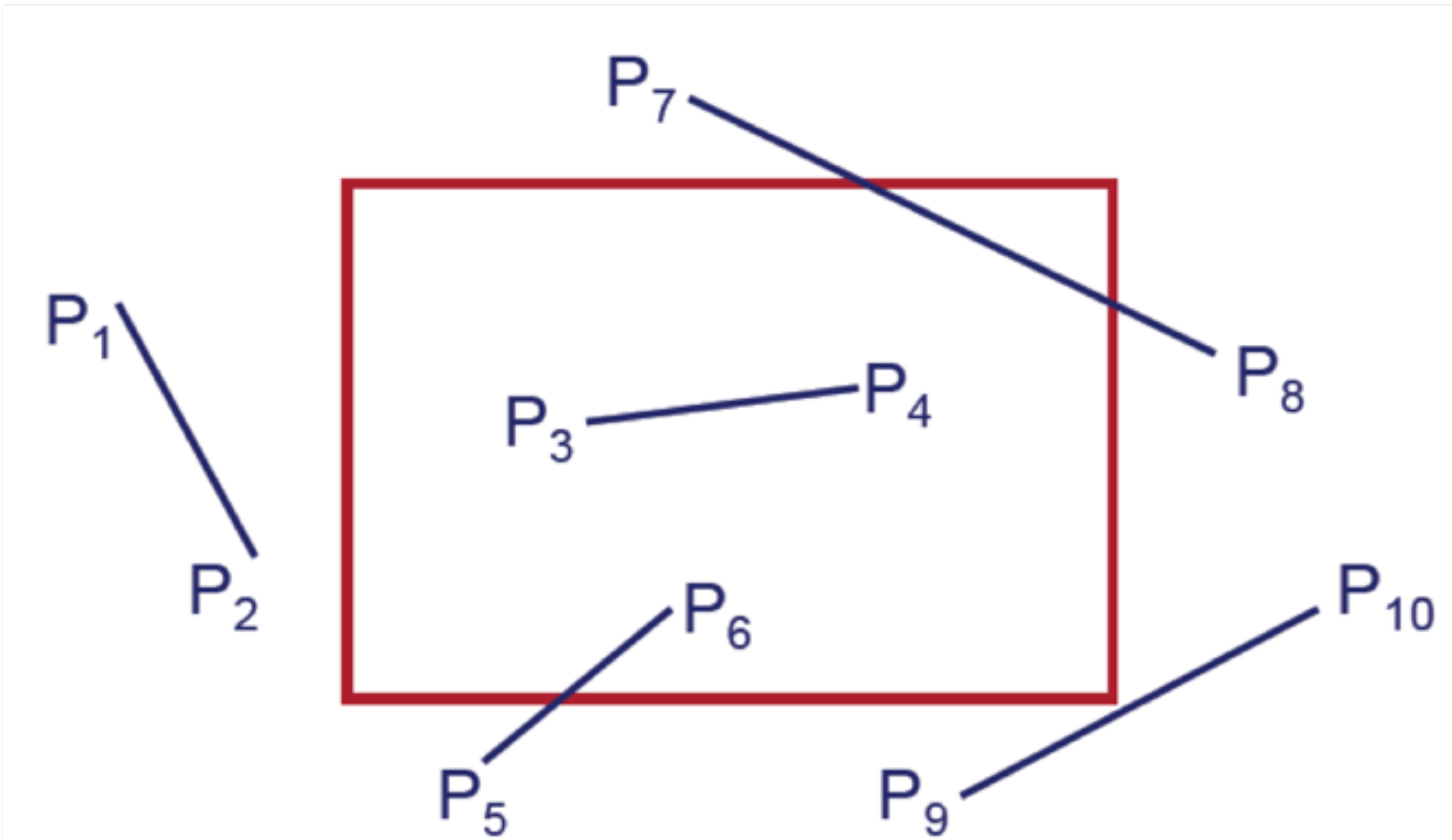
Cohen-Sutherland clipping

- Space outcodes
  - 1bit per half plane
  - Outcodes o1, o2 from line points

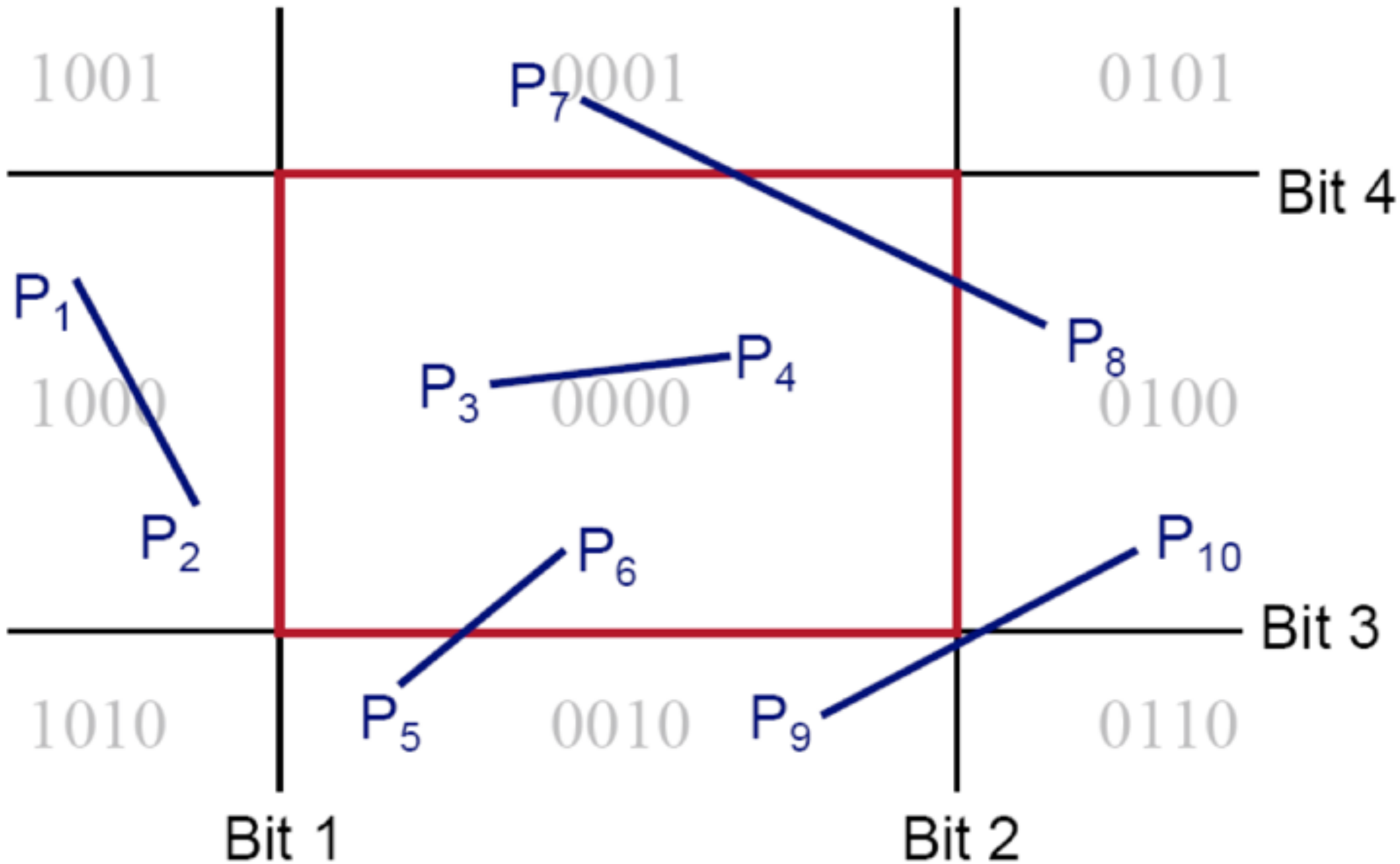


- **Tests**:
  - o1 = o2 = 0 : inside view
  - o1 != 0, o2 = 0 : must clip
  - o1 & o2 != 0 : outside view
  - o1 & o2 = 0 : maybe clip

Example

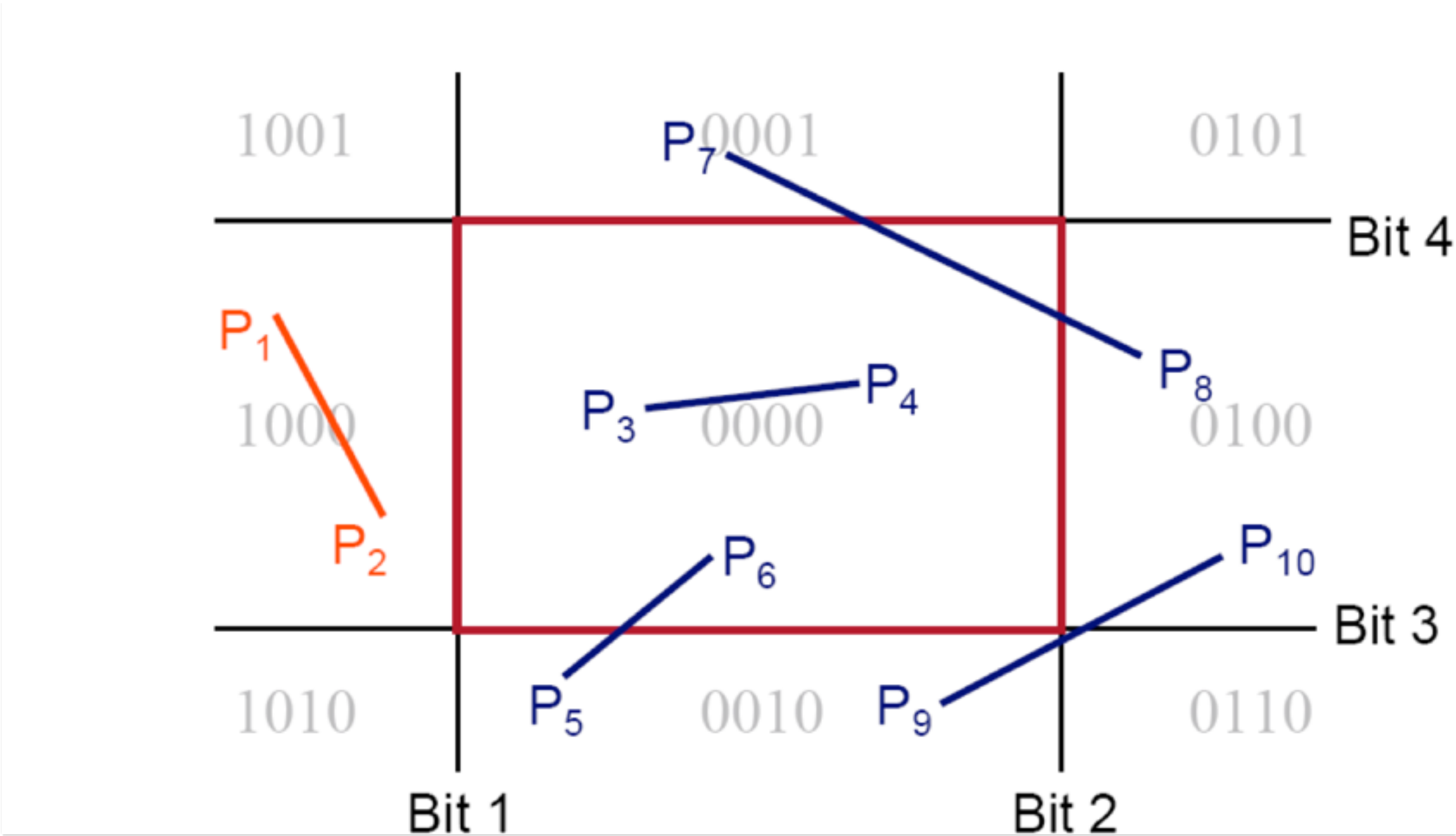


Example



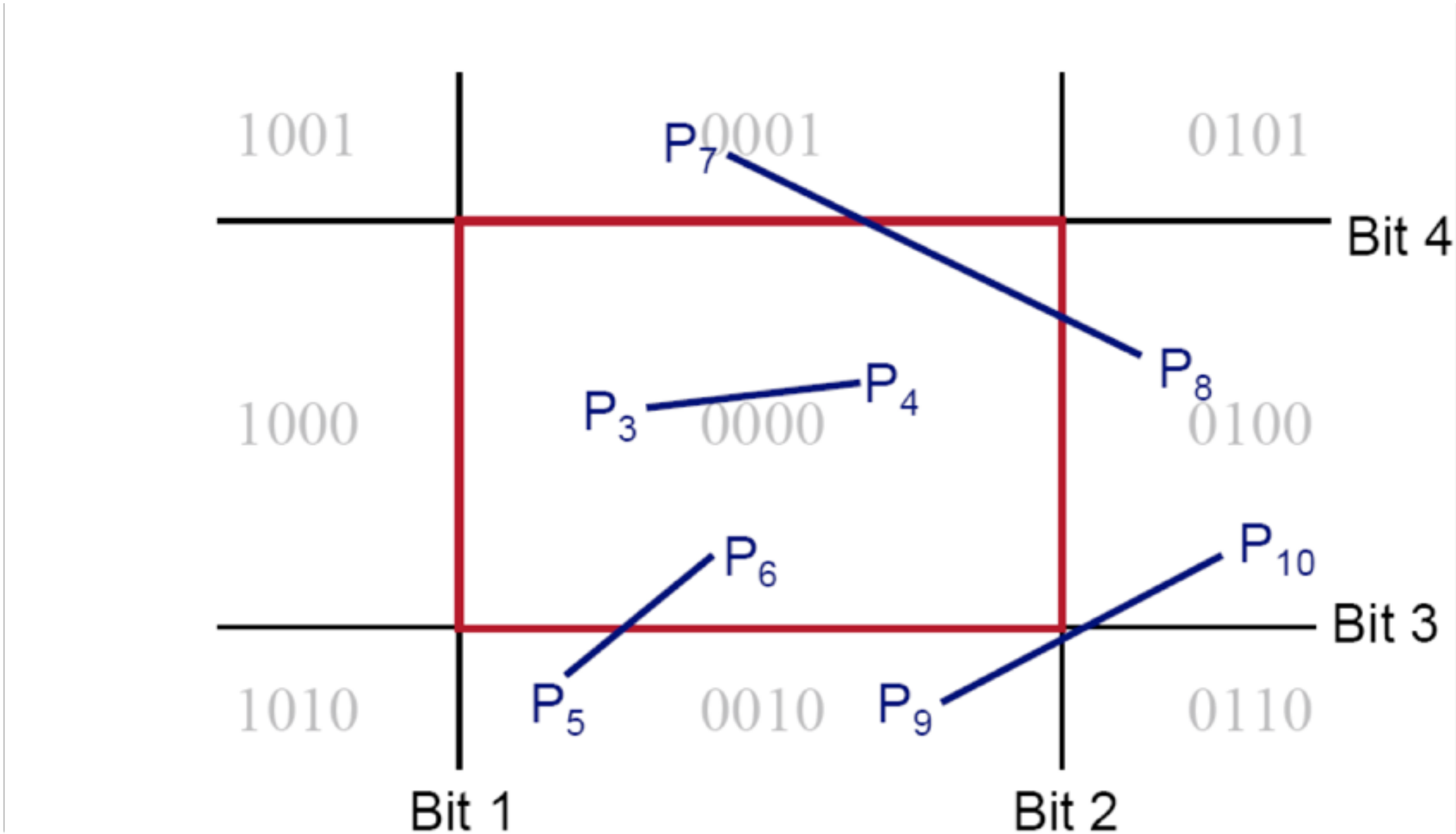
1001
P7
0001
0101
Bit 4
P1
P3
P4
P8
1000
0000
0100
P2
P6
P10
Bit 3
1010
P5
0010
P9
0110
Bit 1
Bit 2

Example



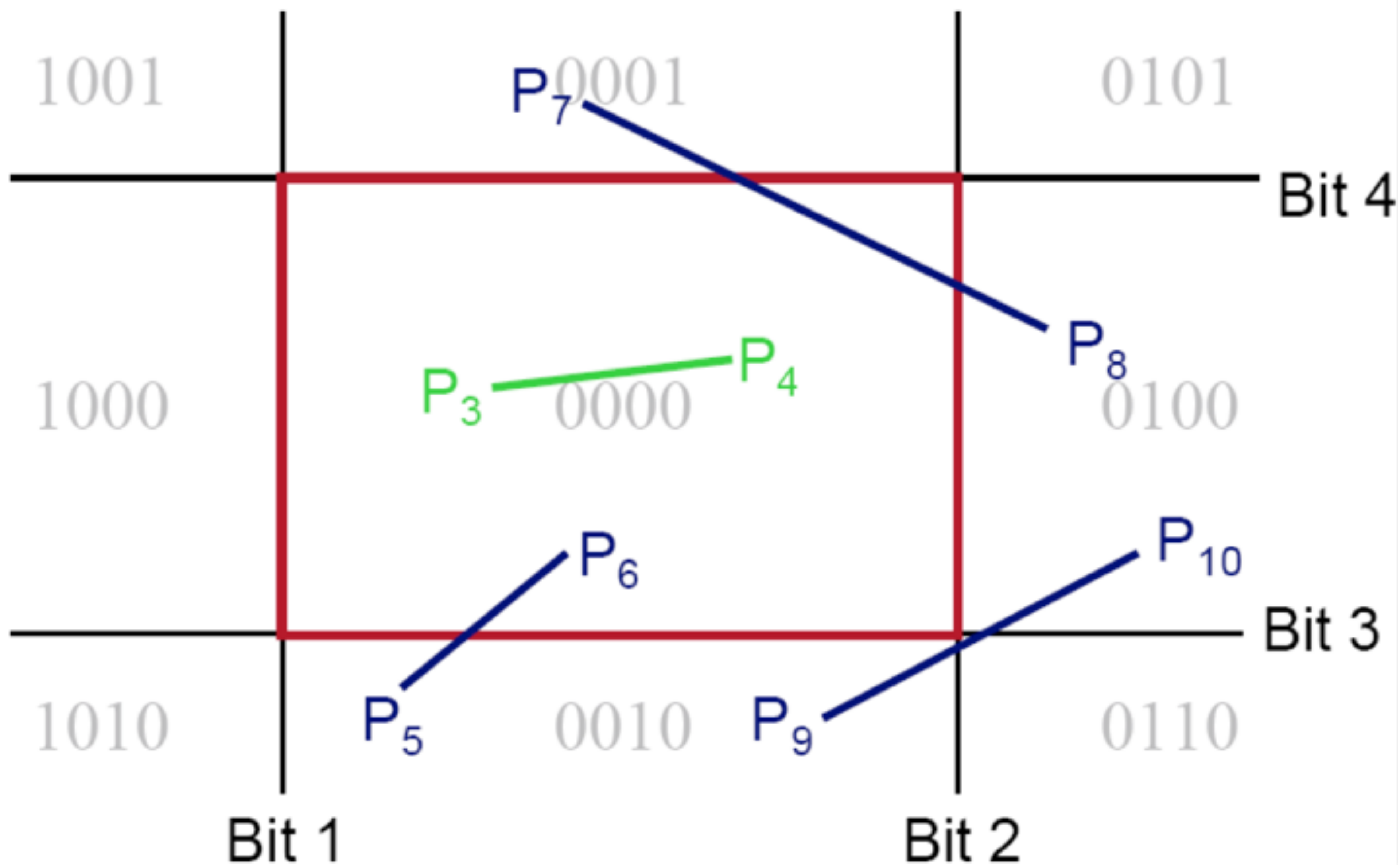
1001
P7
0001
0101
Bit 4
P1
P8
P3
P4
1000
0000
0100
P2
P6
P10
Bit 3
P5
P9
1010
0010
0110
Bit 1
Bit 2

Example



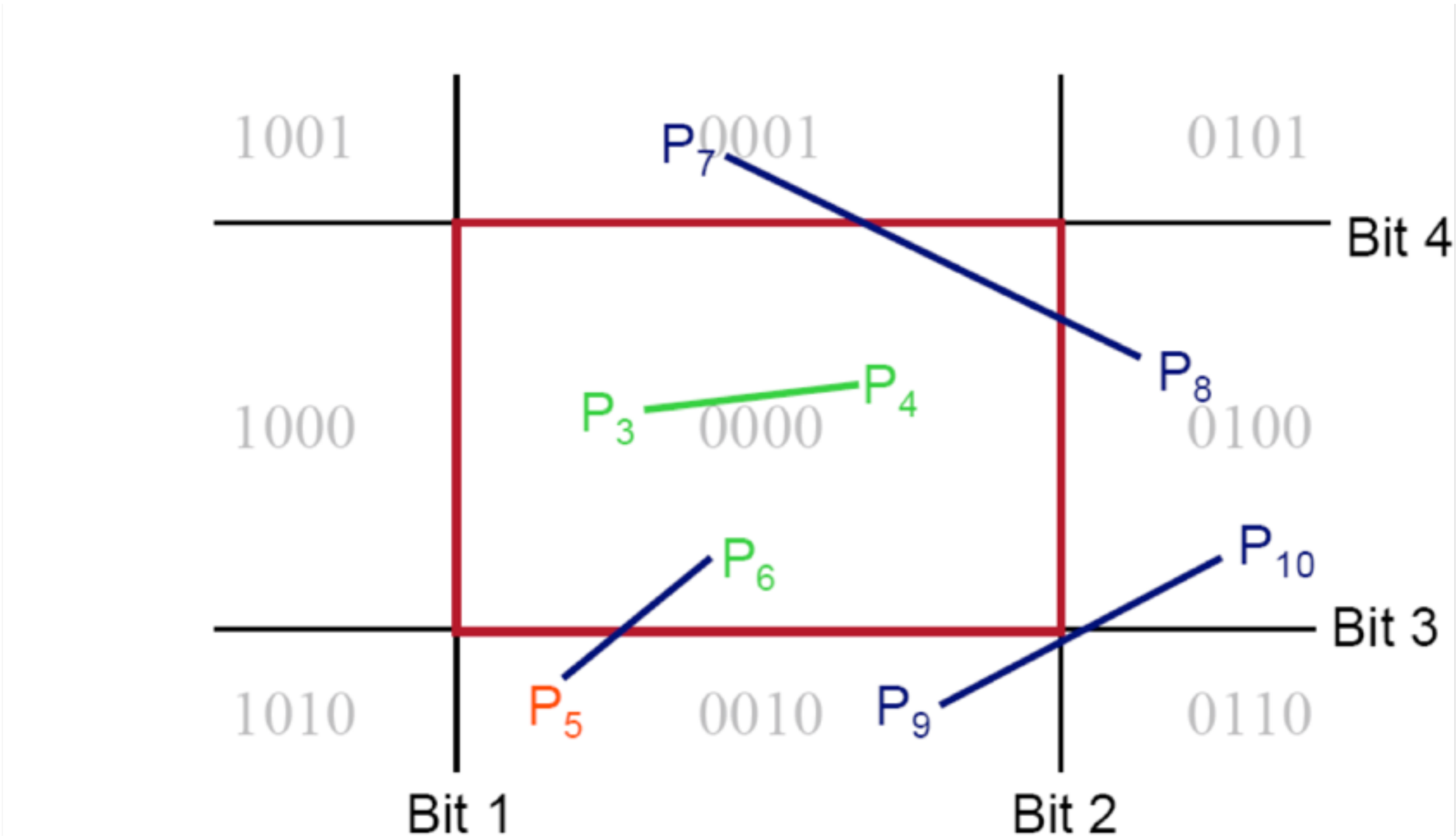
1001
0001
0101
P7
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P5
P9
1010
0010
0110
Bit 1
Bit 2

Example



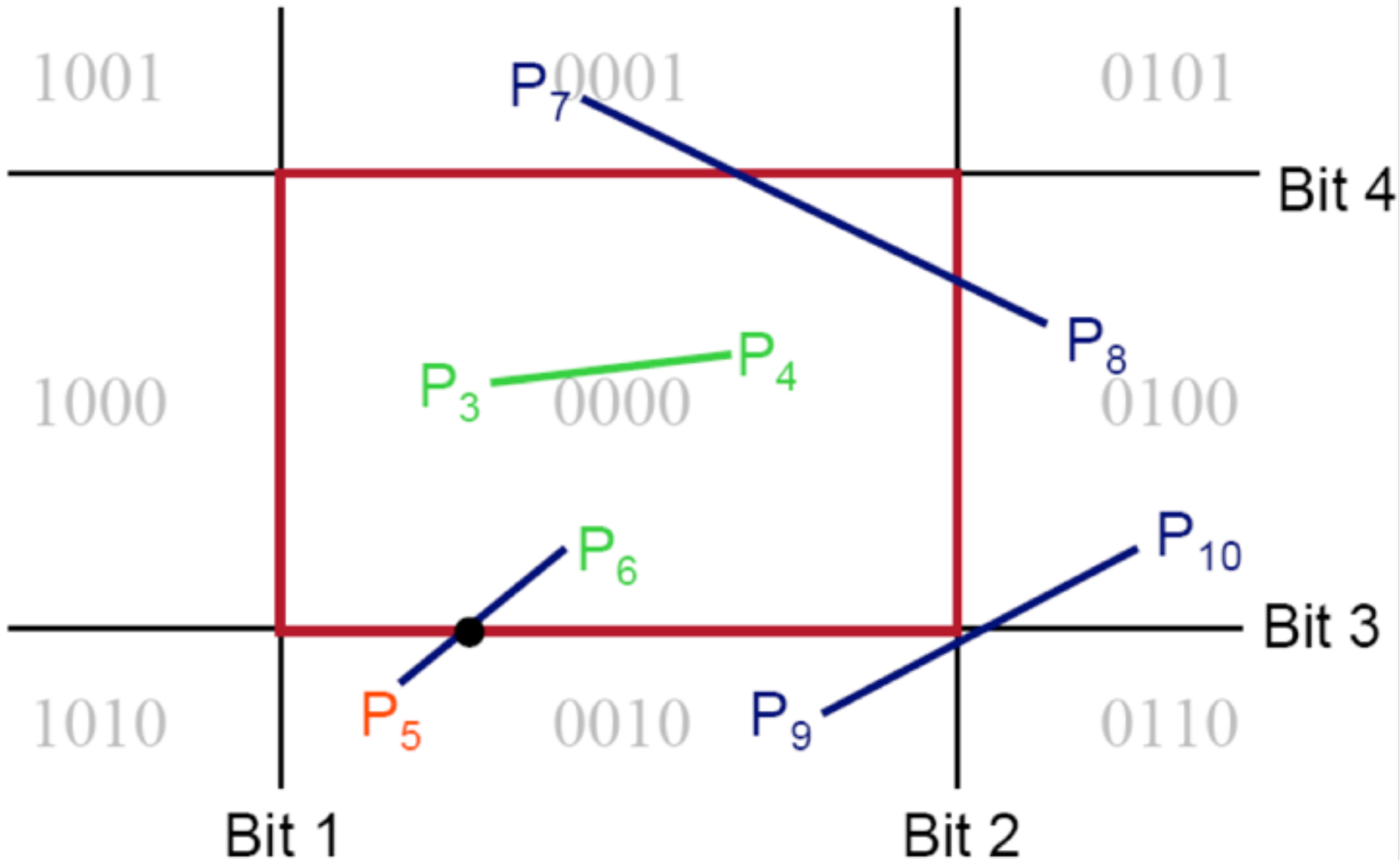
1001
P7
0001
0101
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P5
P9
1010
0010
0110
Bit 1
Bit 2

Example



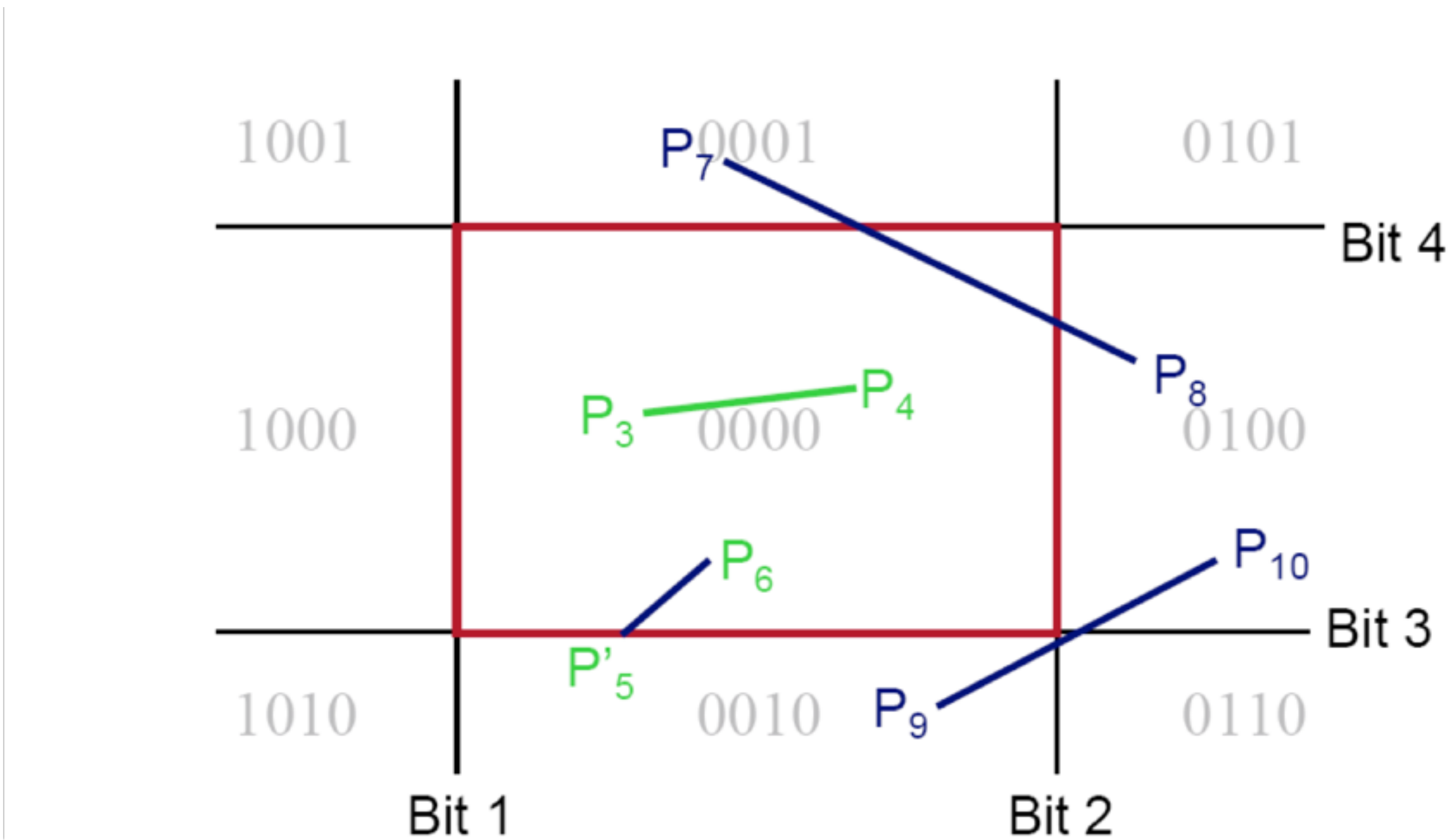
1001
0001
0101
P7
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
1010
P5
0010
P9
0110
Bit 1
Bit 2

Example



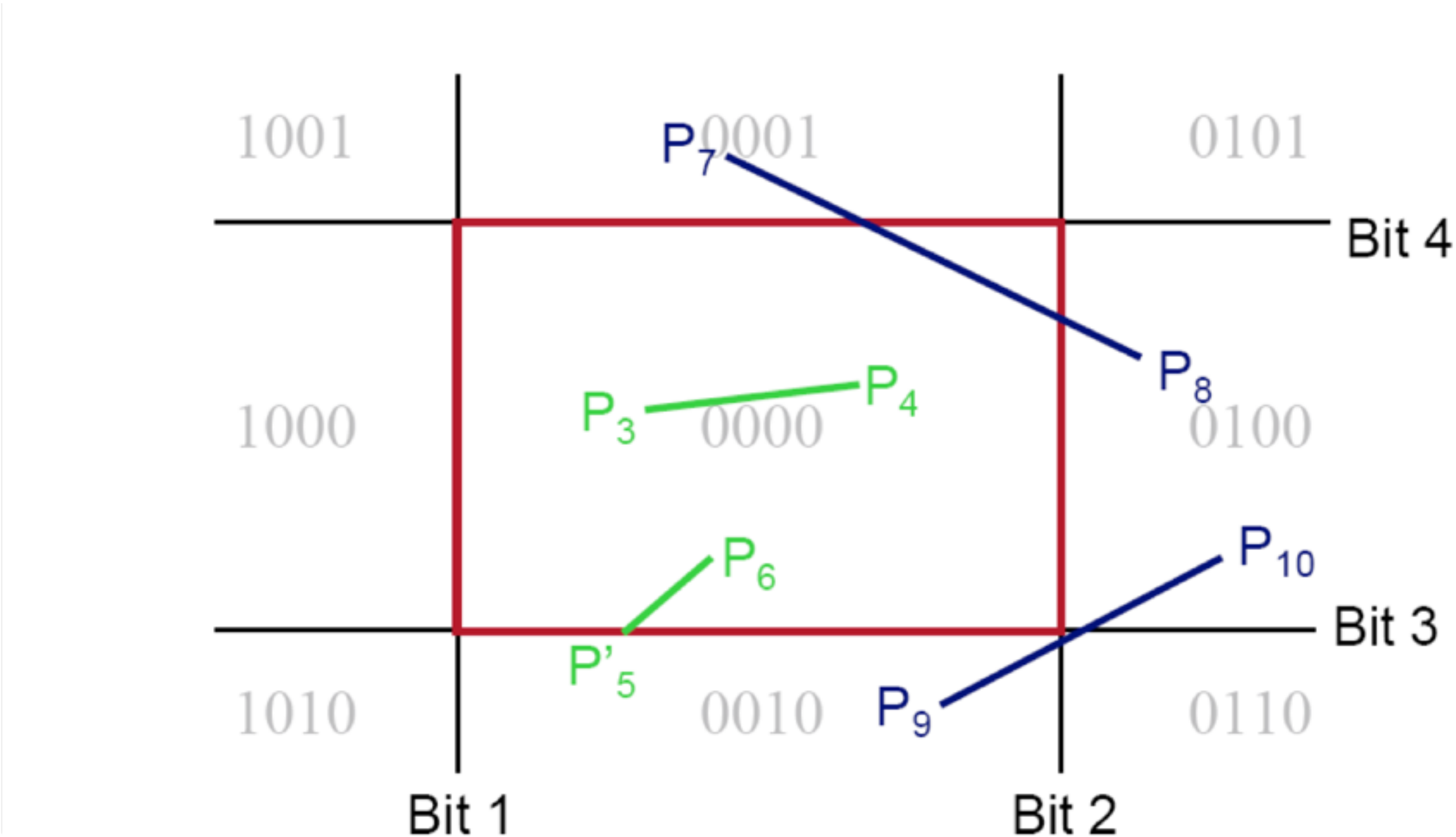
1001
P7
0001
0101
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P5
P9
1010
0010
0110
Bit 1
Bit 2

Example



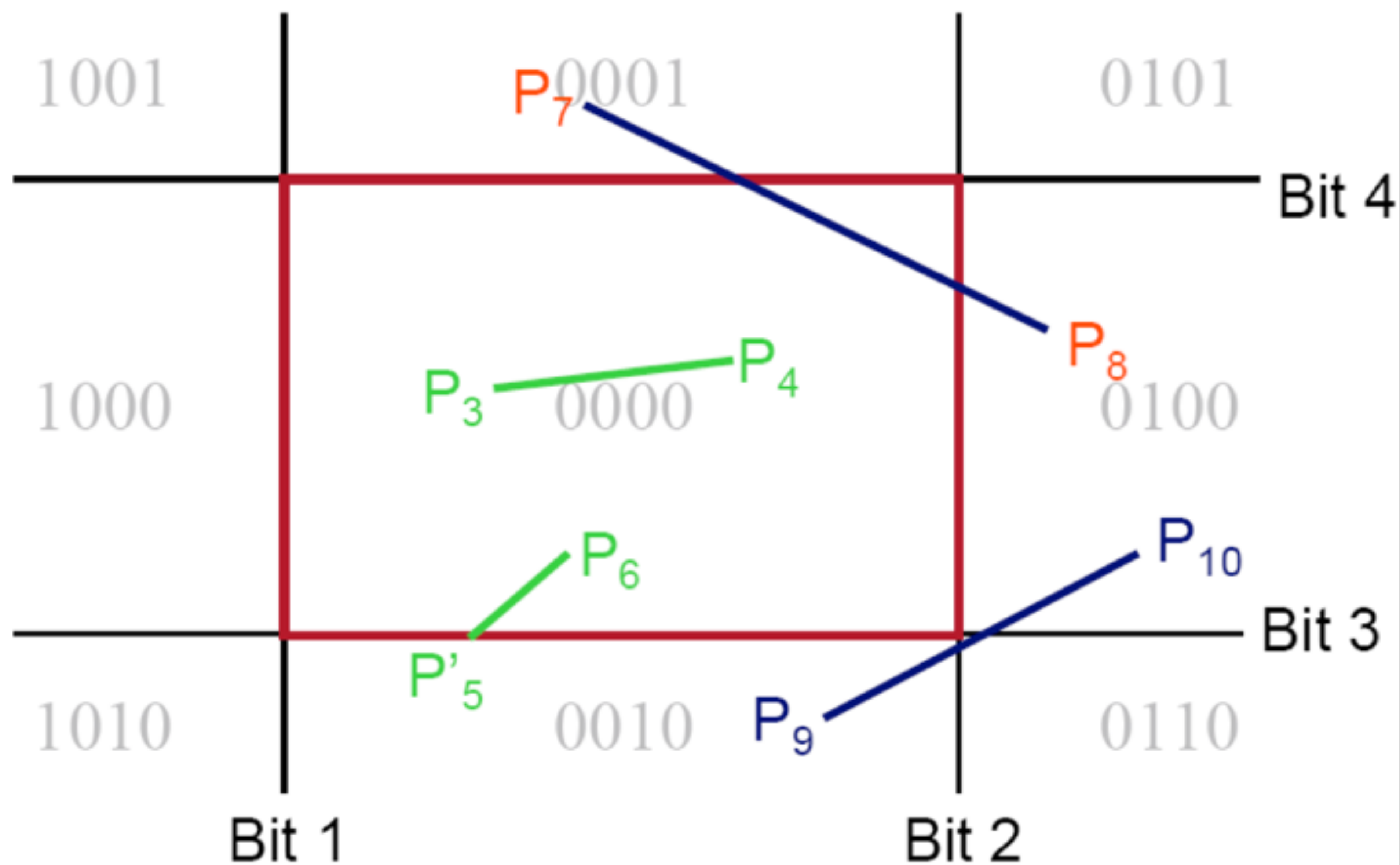
1001
P7
0001
0101
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example



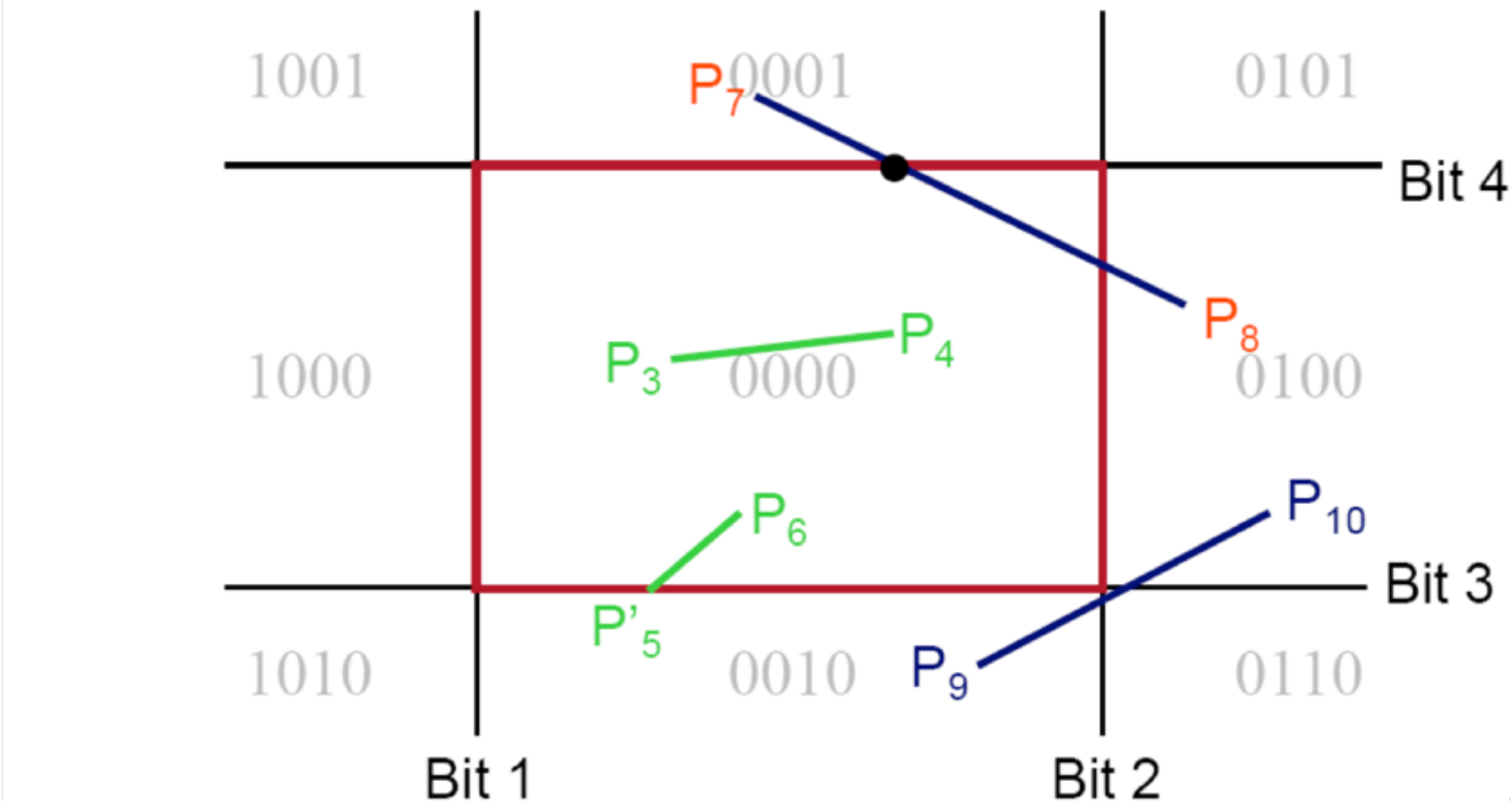
1001
0001
0101
P7
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example



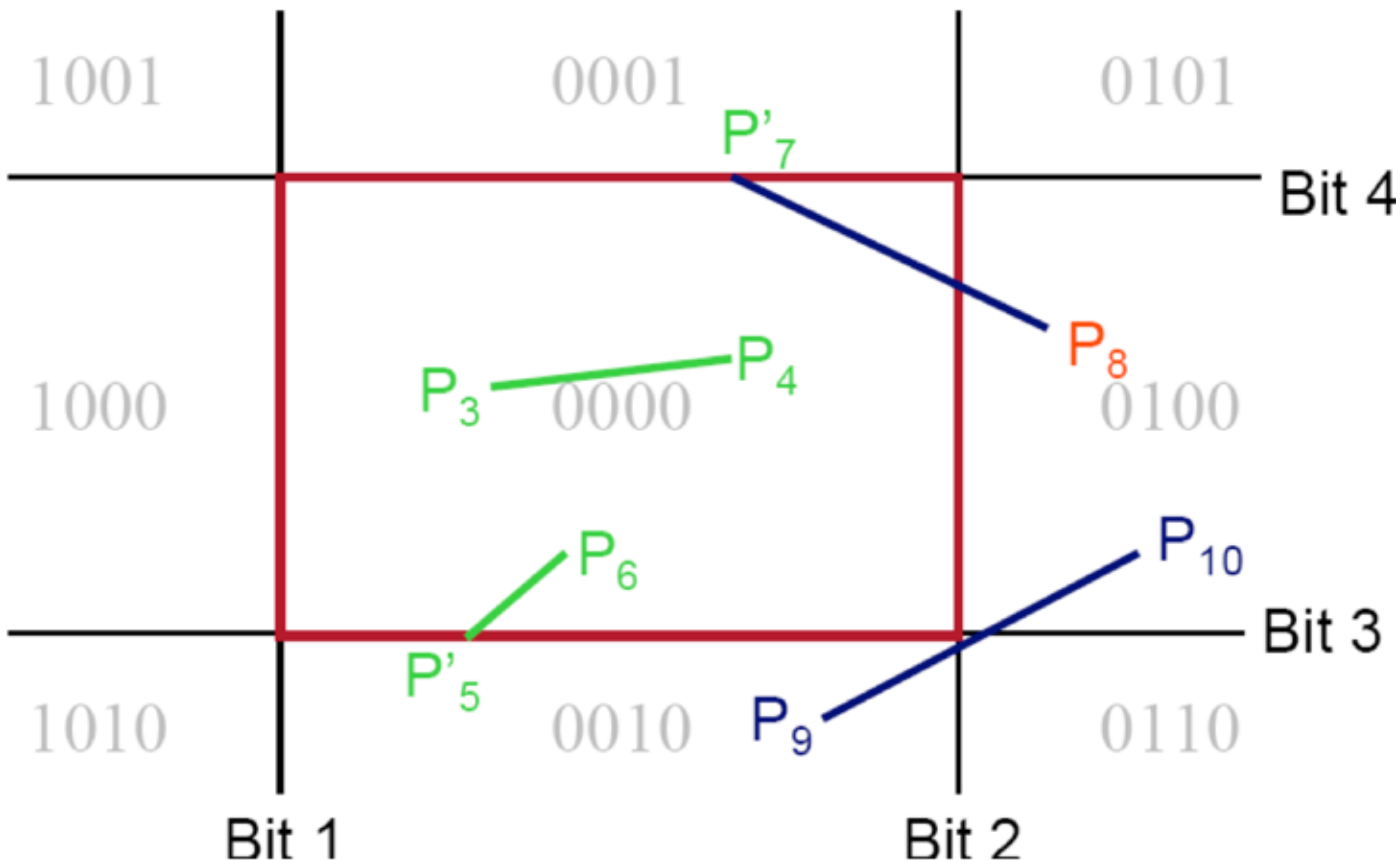
1001
0001
0101
P7
Bit 4
P8
P3
P4
1000
0000
0100
P10
P6
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example

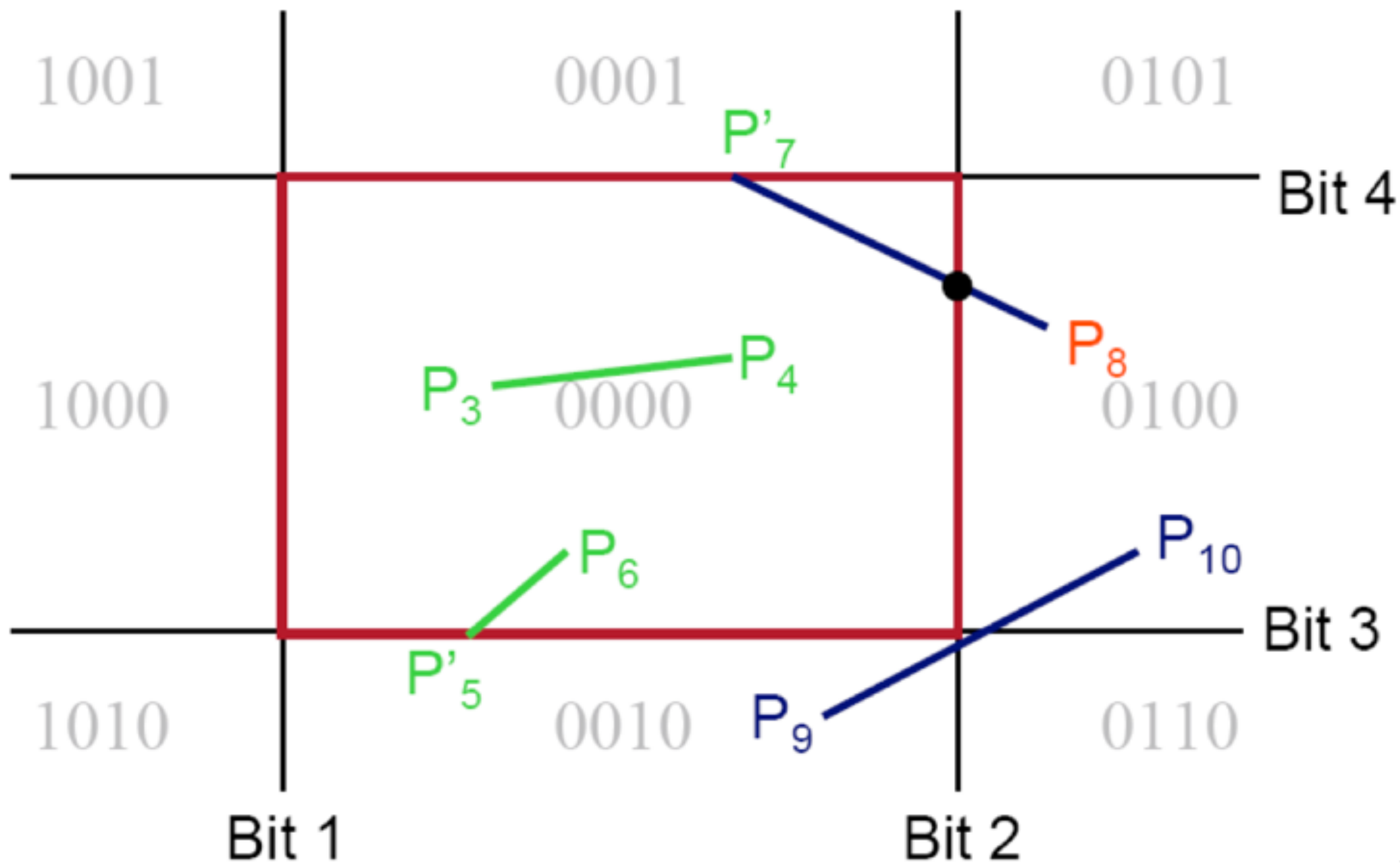


1001
P7
0001
0101
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example

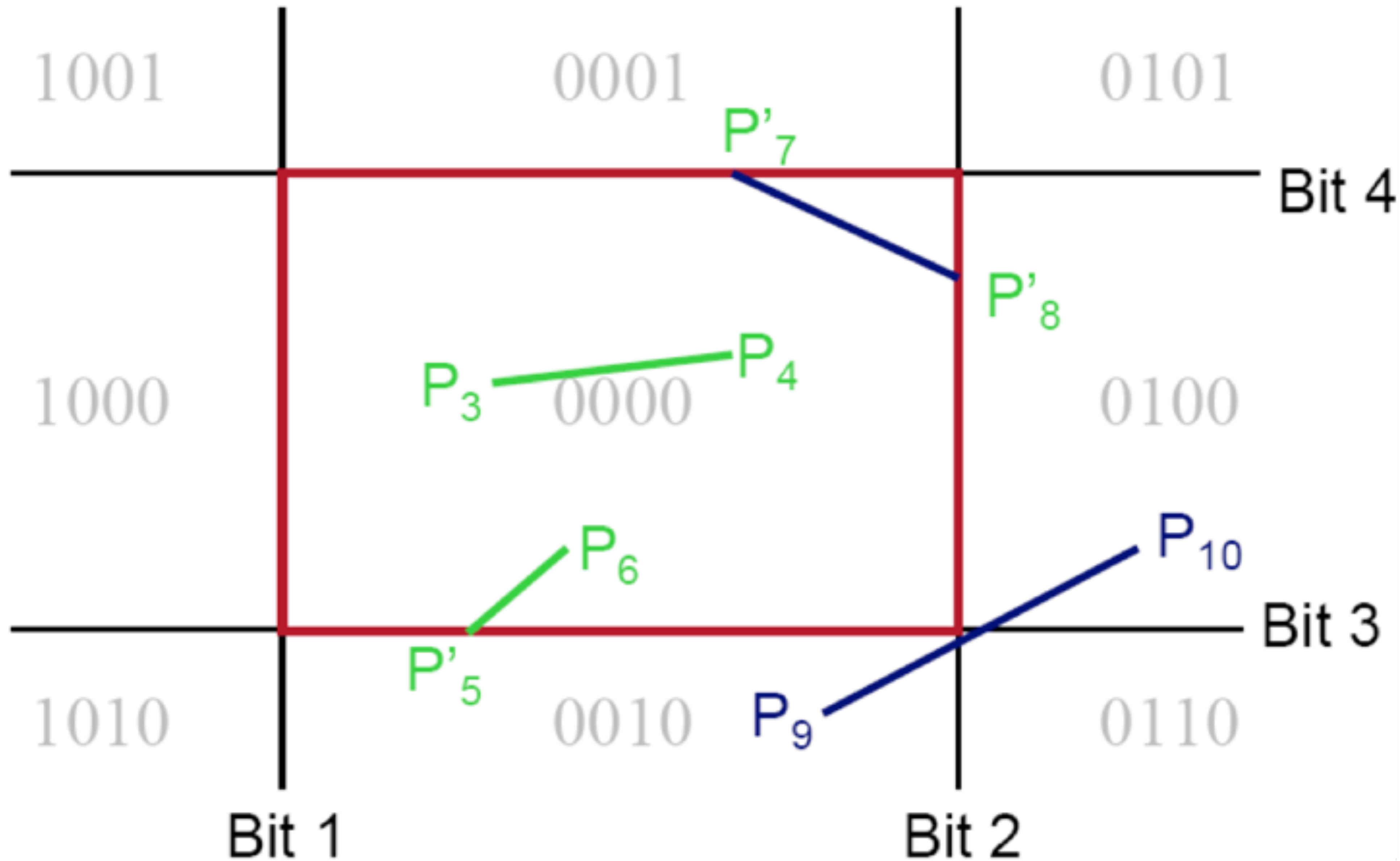

1001
0001
0101
P'7
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example

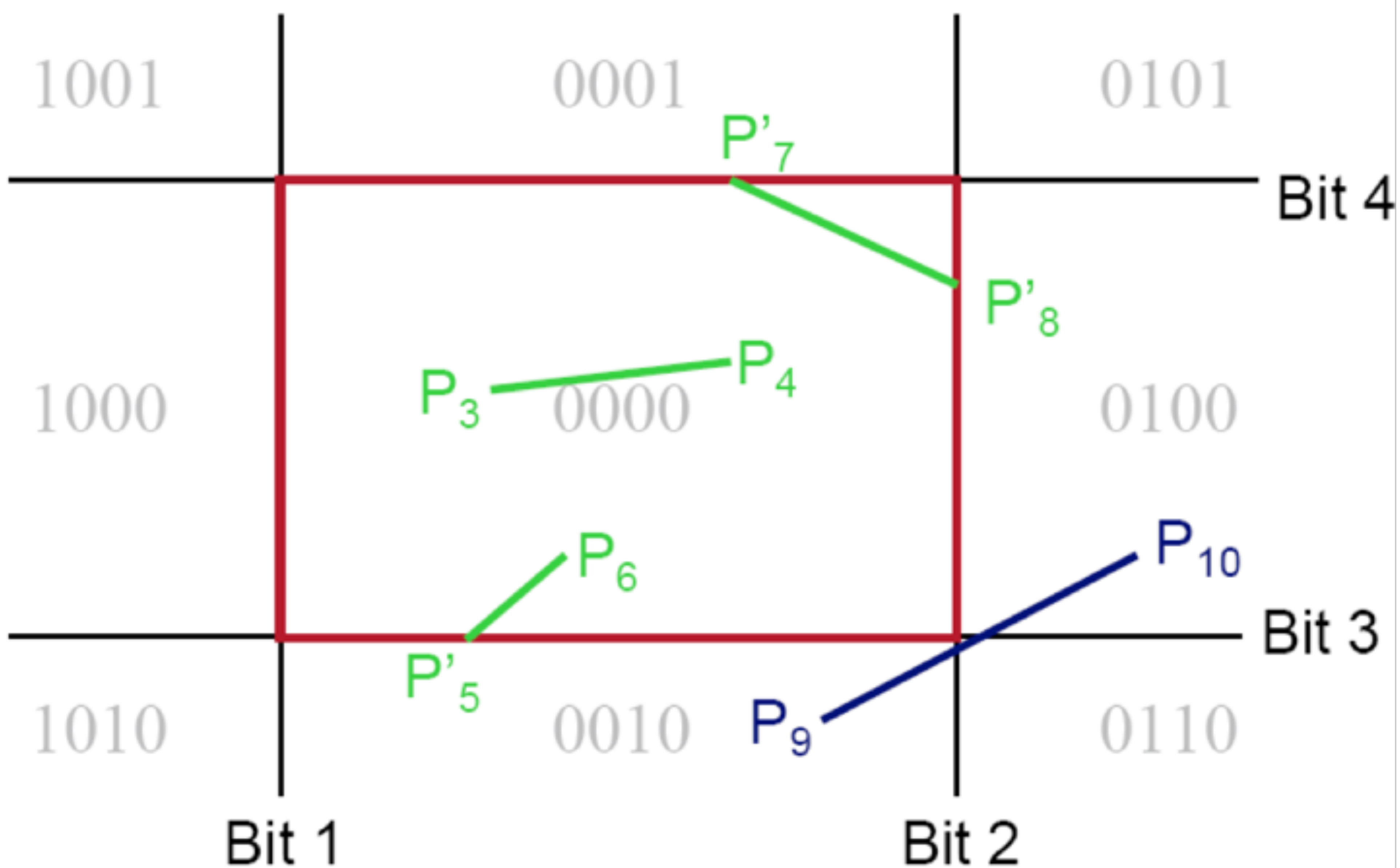


1001
0001
0101
P'7
Bit 4
P8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example

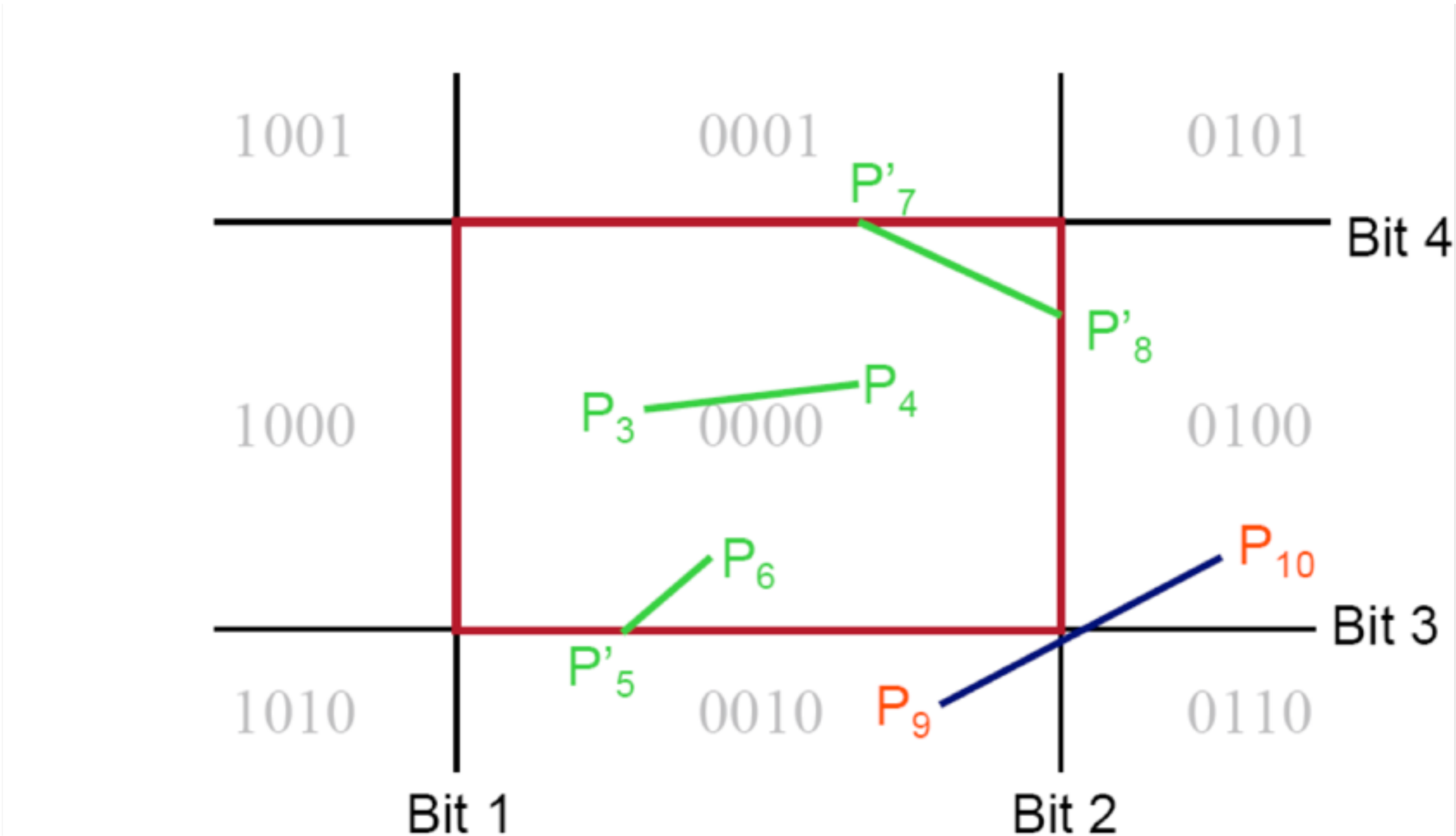

1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example



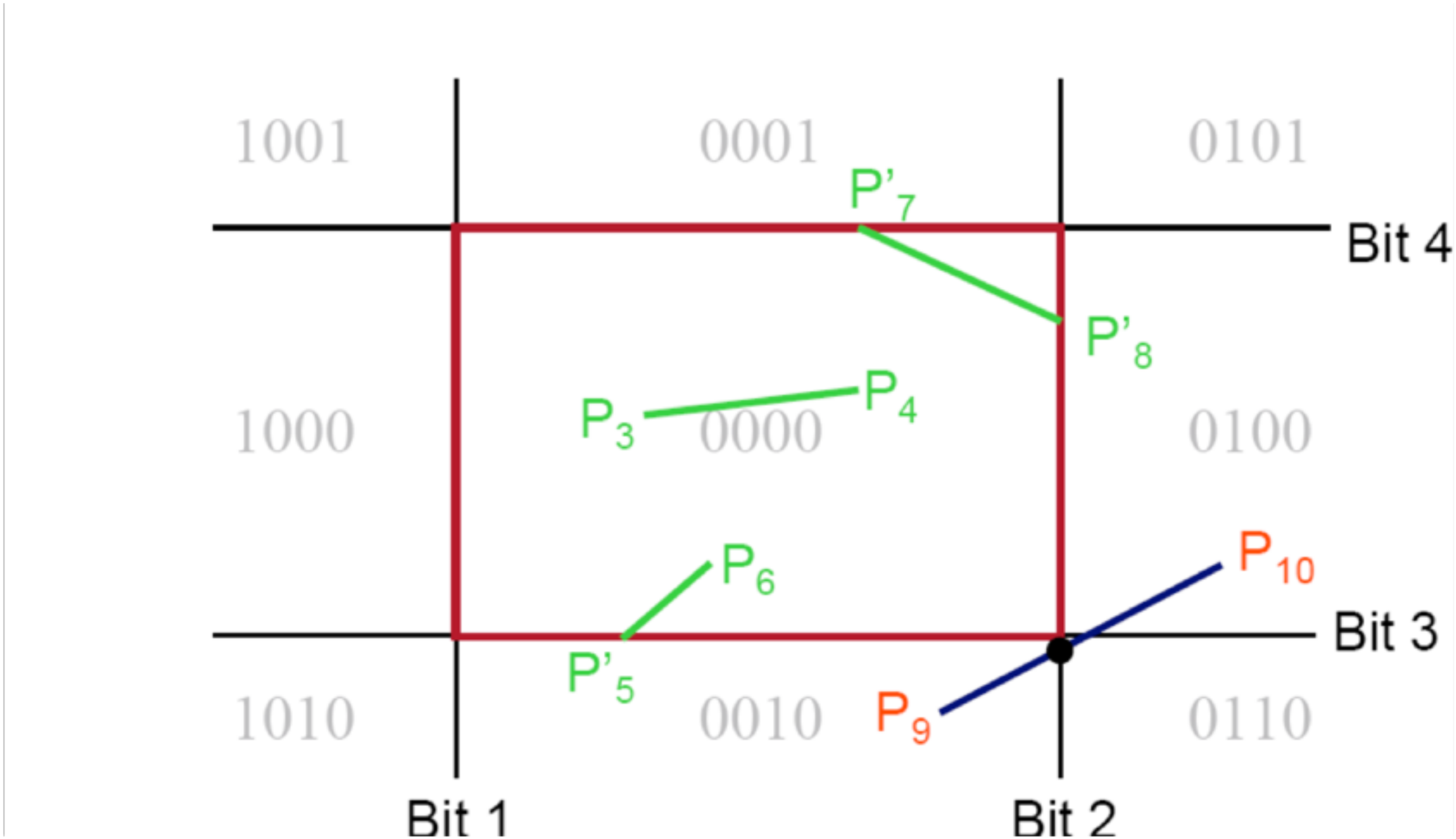
1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example



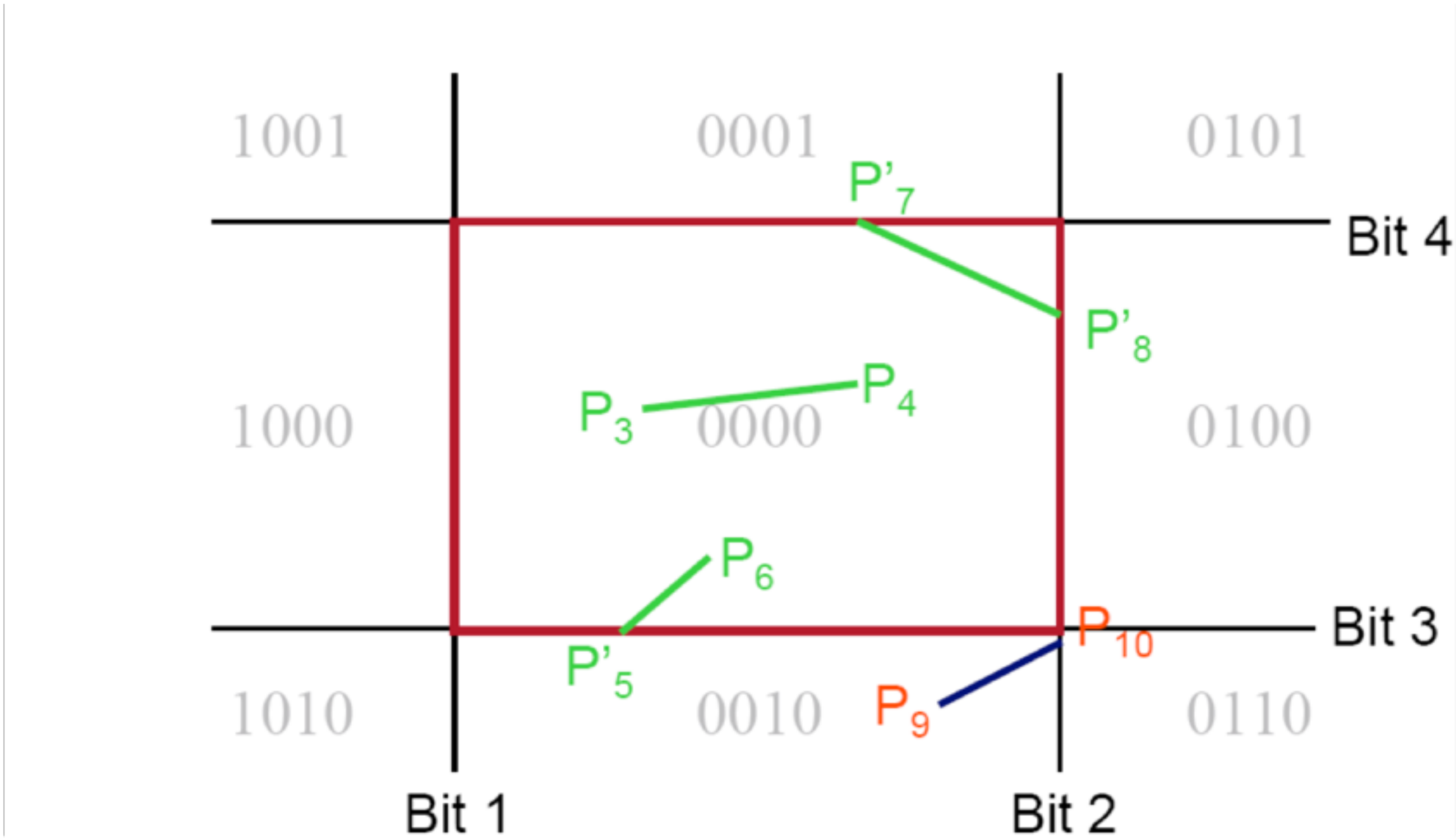
1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
P9
1010
0010
0110
Bit 1
Bit 2

Example

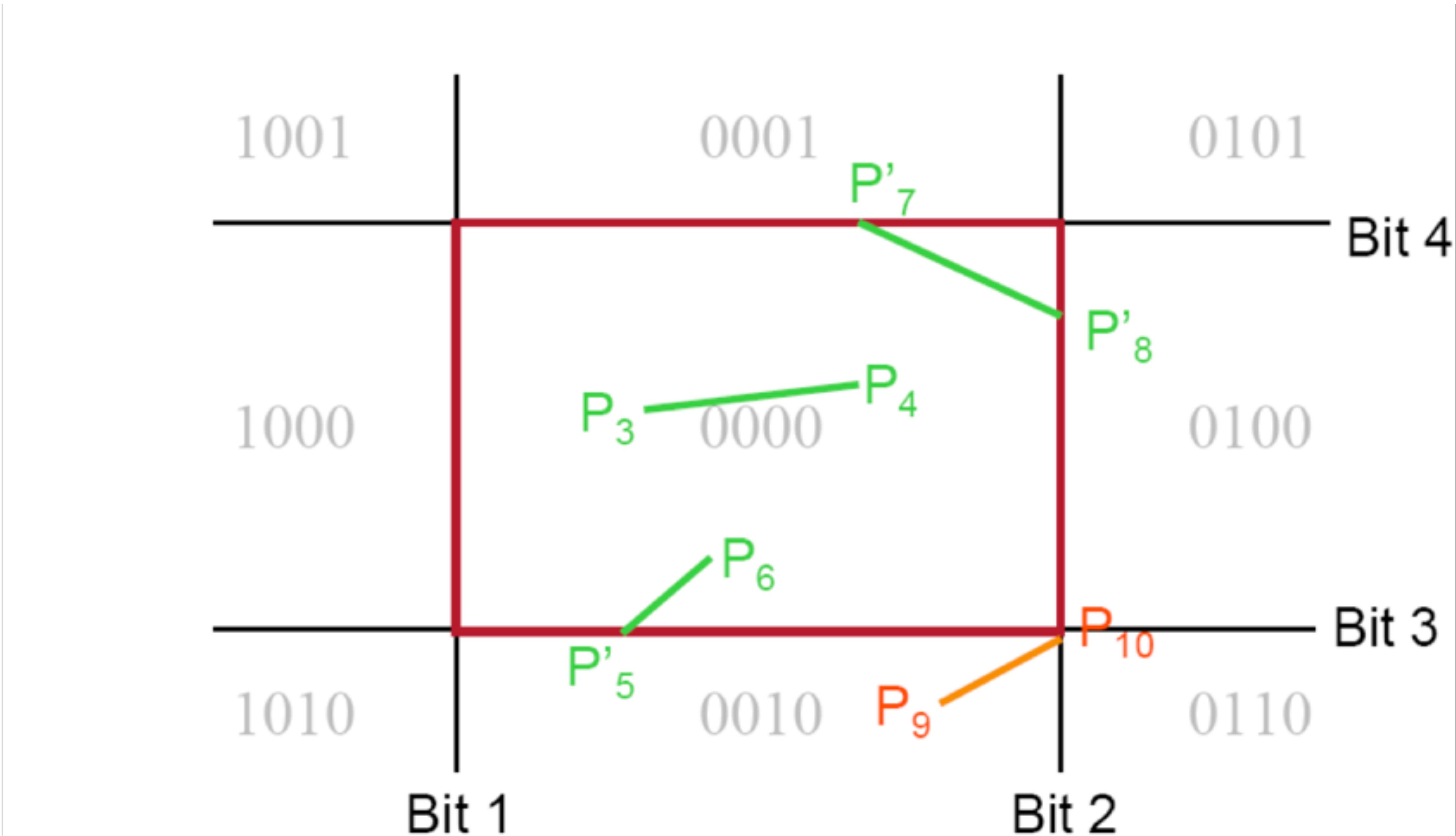


1001
0001
0101
P'7
Bit 4
P'8
P4
P3
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example

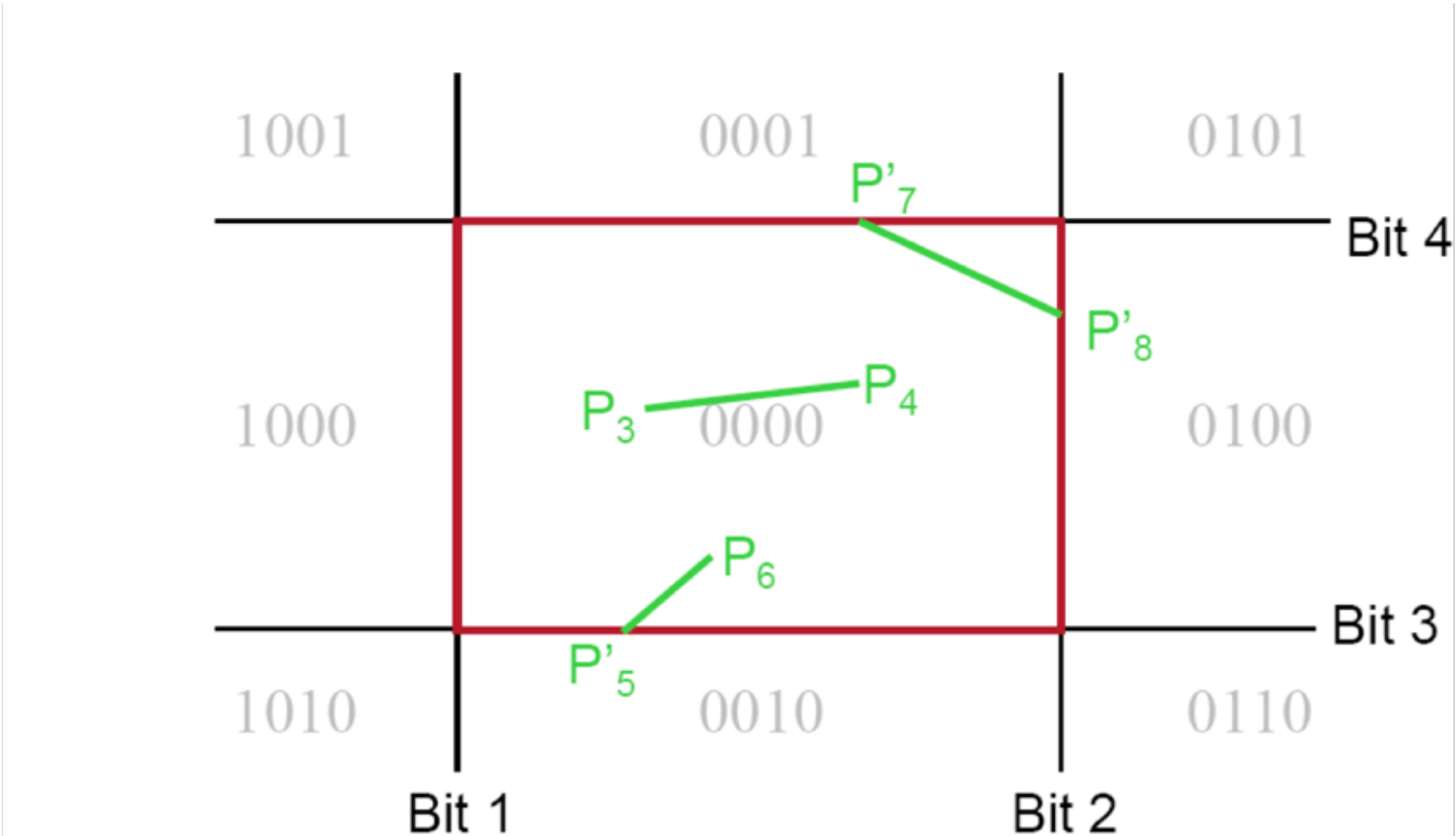

1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
P10
Bit 3
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example



1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
Bit 3
P10
P'5
1010
0010
P9
0110
Bit 1
Bit 2

Example


1001
0001
0101
P'7
Bit 4
P'8
P3
P4
1000
0000
0100
P6
Bit 3
P'5
1010
0010
0110
Bit 1
Bit 2

Intersection

- Can use explicit line equation
  - $y = mx+b$
  - Find for m, b
  - Solve for intersection values
- Handle vertical lines as special case

Polygon clipping

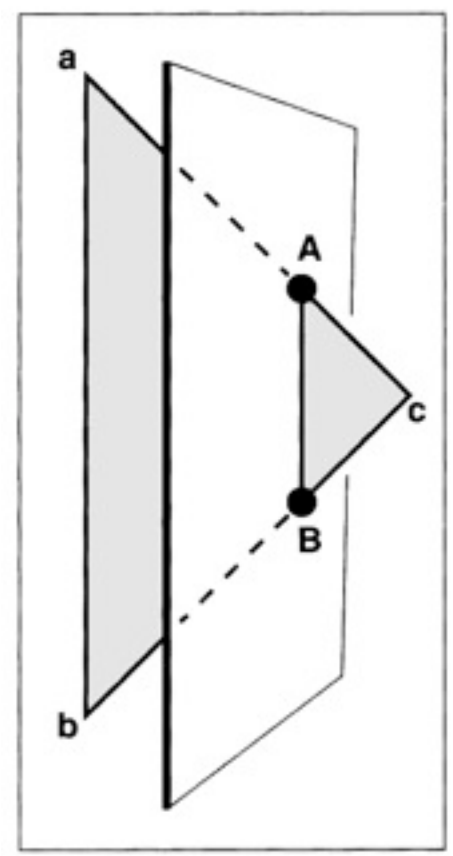
- General polygon clipping
  - Intersect clip line against polygon
  - Insert new vertices
  - Create new polygons

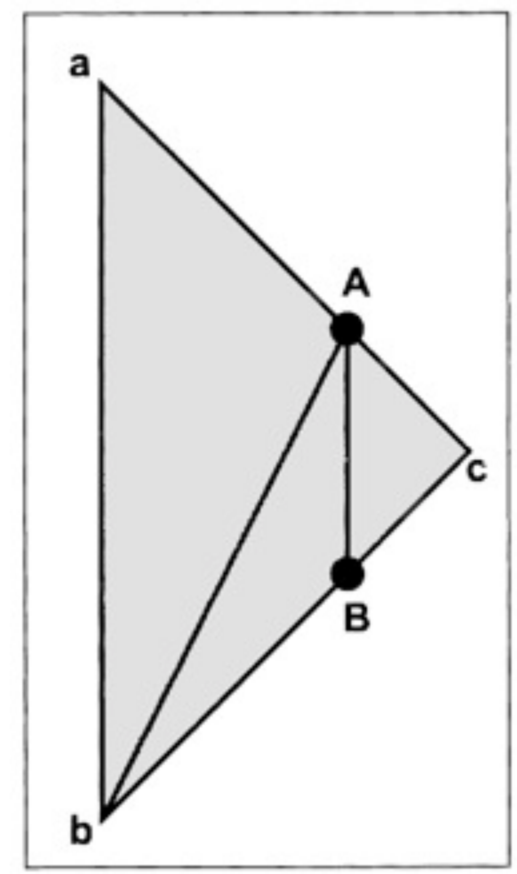
Polygon clipping

- More complicated if
  - Topology restrictions (triangles only!)
  - Surface properties (vertex attributes)

Triangle clipping

- Triangles must appear as single objects
- Tesselate triangle during clipping
- Compute vertex attributes if needed





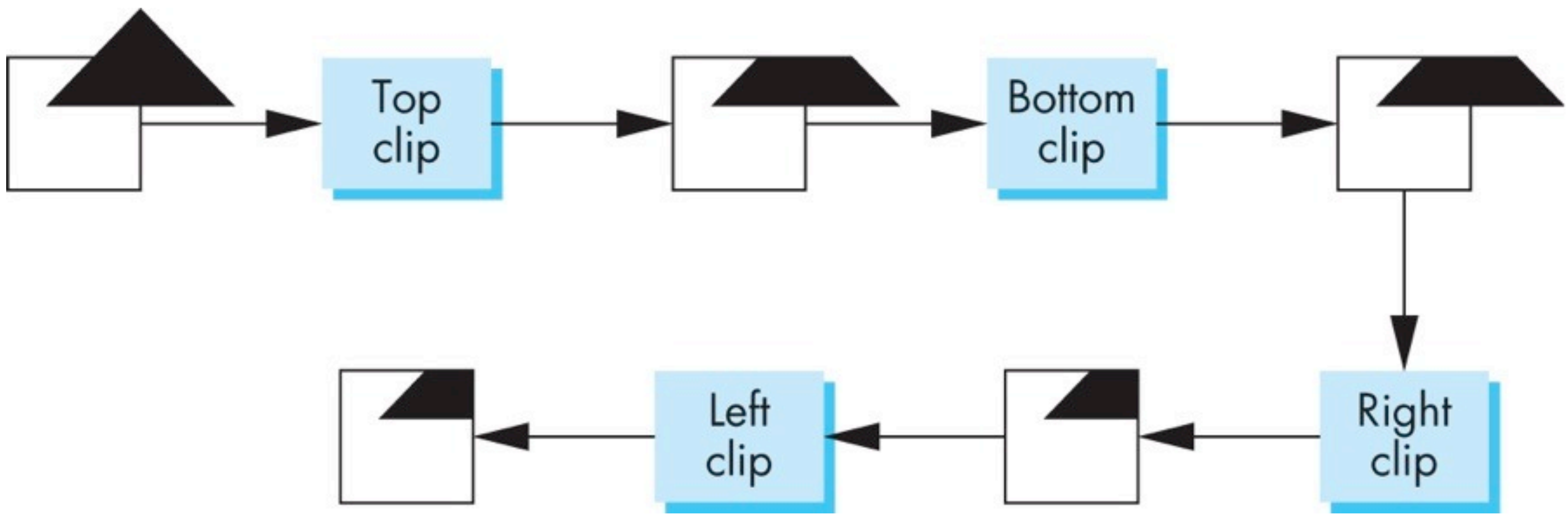
Clipping pipeline

- Clipping tests are independent
- Can be performed in serial or parallel

- Pipeline line clipping against axes bounds
  - Test each axis independently

Clipping pipeline

- Final pipeline result is fully clipped polygon

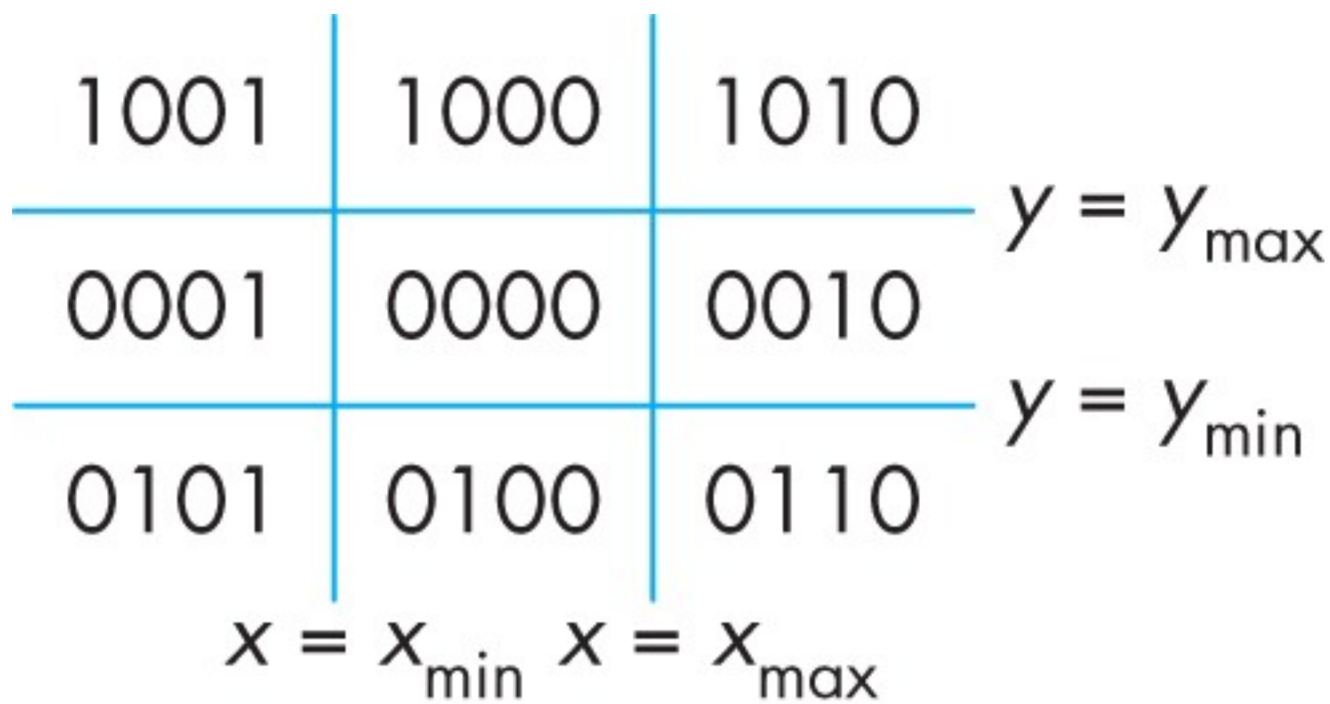


3D clipping

- Extend Cohen-Sutherland to 3D?

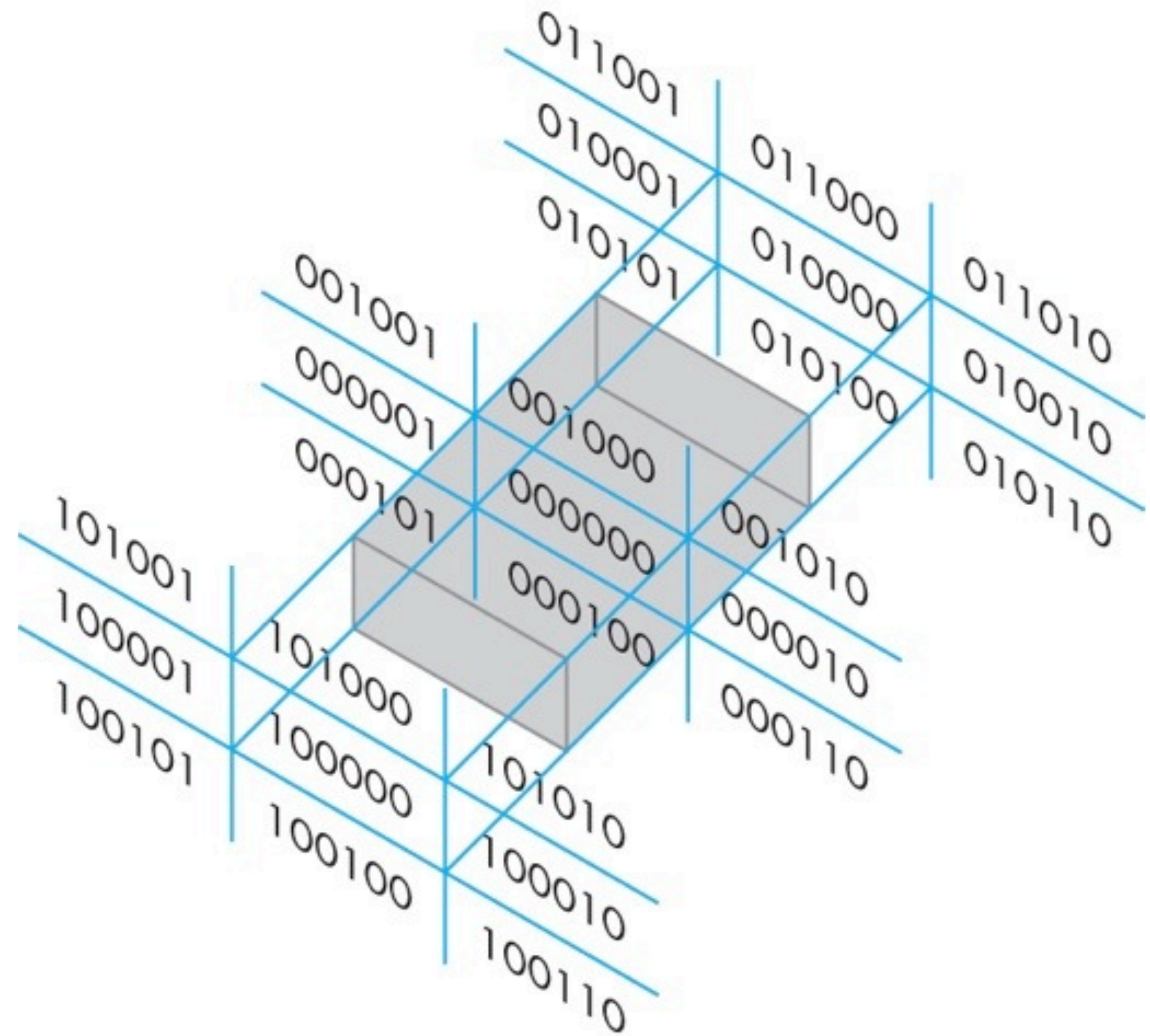
3D Cohen-Sutherland

- 2D case had 4 bounds & 4 bit opcodes



3D Cohen-Sutherland

- 3D case has 6 bounds & 6 bit opcodes



3D Clipping

- Can operate in clip space if we assume NDC is (-1, -1, -1) : (1, 1, 1)
- Remember, need to homogenize by w

$-w \leq x \leq w$

$-w \leq y \leq w$

$-w \leq z \leq w$

OpenGL Clipping

- Start in 2D, extend to 3D
- Can clip in any coordinate frame
  - OpenGL clips in 'Clip space'
  - Just before homogenization and NDC

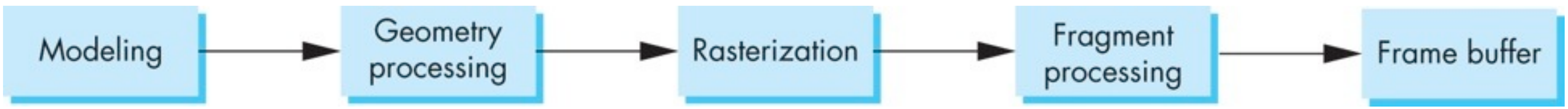
OpenGL Clipping

- Camera takes vertices to view/camera space
- Projection takes vertices to clip space
- In clip space
  - Primitives are clipped
  - w is homogenized
- Result is Normalized Device Coords.

Render pipeline

- After homogenization
  - All geometry is in NDC
  - No geometry out of view volume (NDC)

Render pipeline



- Render pipeline changes coordinate/vector spaces
- Ready for
    - Fragment conversion
    - Interpolation
    - Depth sorting

Rasterization

- Compute fragment locations in window coordinates
- Interpolate vertex attributes
- Compute fragment color

- Sort fragments by depth

Other methods

- Many other ways to clip

Liang-Barsky clipping

- Form parametric equation of line
- Compute entrance and exit from clipping region
- Check if order is valid, clip if needed

Parametric lines

- Forming parametric line equation
  - Given points p1 and p2
  - Vector parallel to line is p2-p1
  - 'Start' of line is p1
  - All valid points on line are in range $p = p1 + a(p2-p1)$, where $0 \leq a \leq 1$

Parametric lines

- Forming parametric line equation
  - Given points p1 and p2
  - All valid points in line are between p1 & p2
  - Linearly interpolate between p1 and p2
    $p = (1-a)p1 + a(p2)$, where $0 \leq a \leq 1$

Liang-Barsky clipping

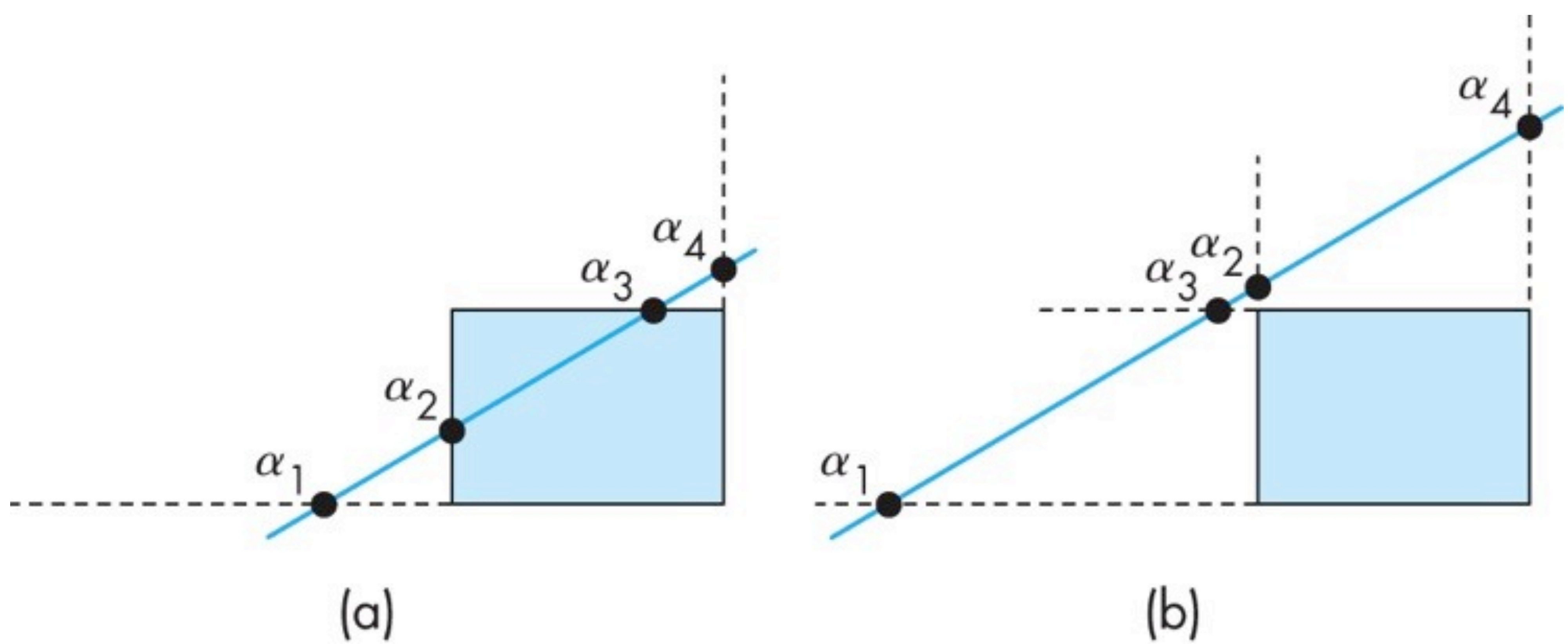
- Form parametric equation of line
- Compute entrance and exit from clipping region
- Check if order is valid, clip if needed

Compute intersect

- Clip region bounded by
  x min, x max
  y min, y max
- Split line equation into x and y forms:
  $x = (1-a)x1 + a(x2)$
  $y = (1-a)y1 + a(y2)$
- Solve for intersects

Compute intersect

- Clip region bounded by
- Solve for intersects



Compute intersect

- Set equal to intersect point
  y max = (1-a)y1 + a(y2)
- Check if a is bounded by 0 and 1
- Compute a intersects for all clip bounds

Compute intersect

- Check if entrance and exit intersects are in correct order
  - Must enter x or y bound
  - Must enter other axis bound
  - Then may exit either axes bounds

Liang-Barsky clipping

- Form parametric equation of line
- Compute entrance and exit from clipping region
- Check if order is valid, clip if needed

Clip line segment

- If entrance and exit are valid
  - Already have intersect points
  - Line is between:
    last entrance point and first exit point