Computer Graphics

Lecture-07
Two –Dimensional Viewing and Clipping

Md Imtiaz Ahmed
Lecturer,
DIIT

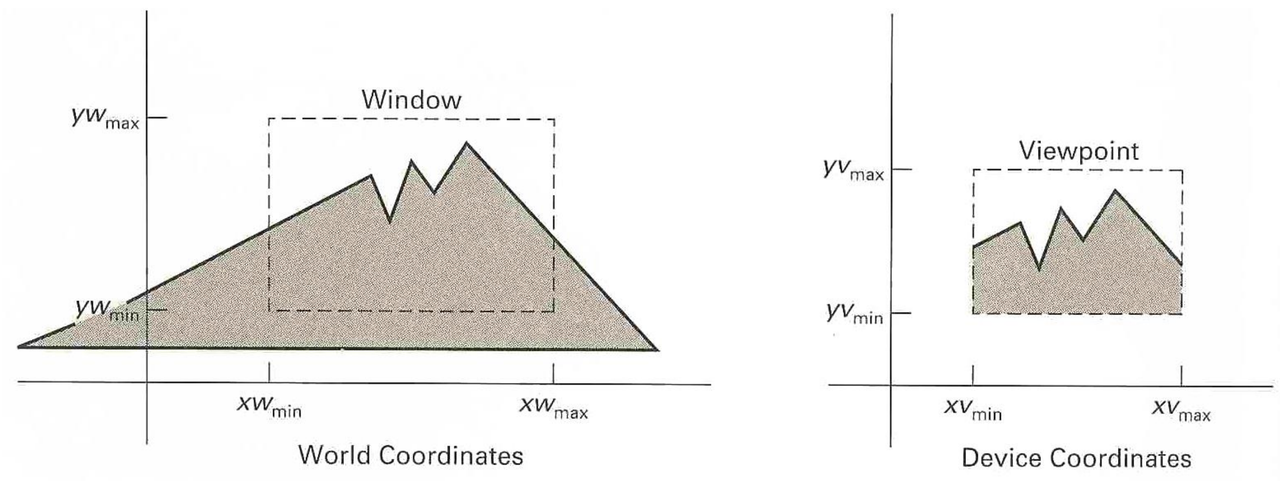
11/15/2021

Introduction

- **window**
  - a world-coordinate area selected for display
  - define what is to be viewed
- **view port**
  - an area on a display device to which a window is mapped
  - define where it is to be displayed
  - define within the unit square
  - the unit square is mapped to the display area for the particular output device in use at that time
- **windows & viewport**
  - be rectangles in standard position, with the rectangle edges parallel to the coordinate axes

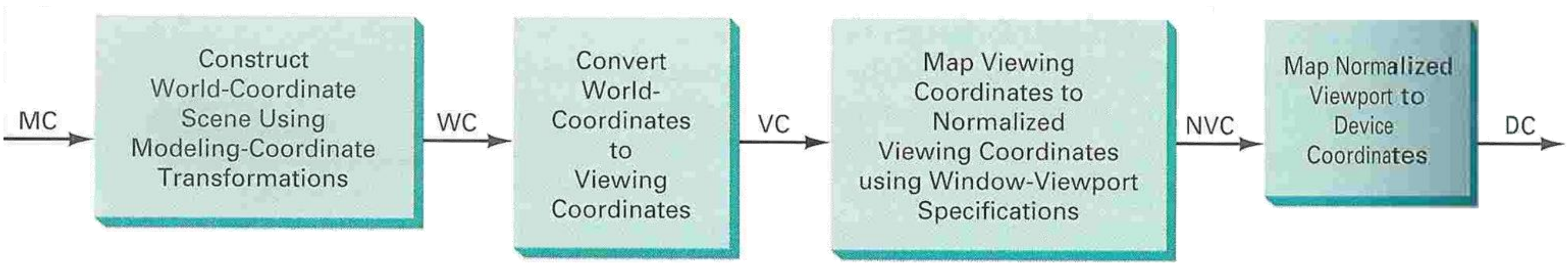
Introduction

- **viewing transformation**
  - the mapping of a part of a world-coordinate scene to device coordinates
  - 2D viewing transformation = window-to-viewport, windowing transformation



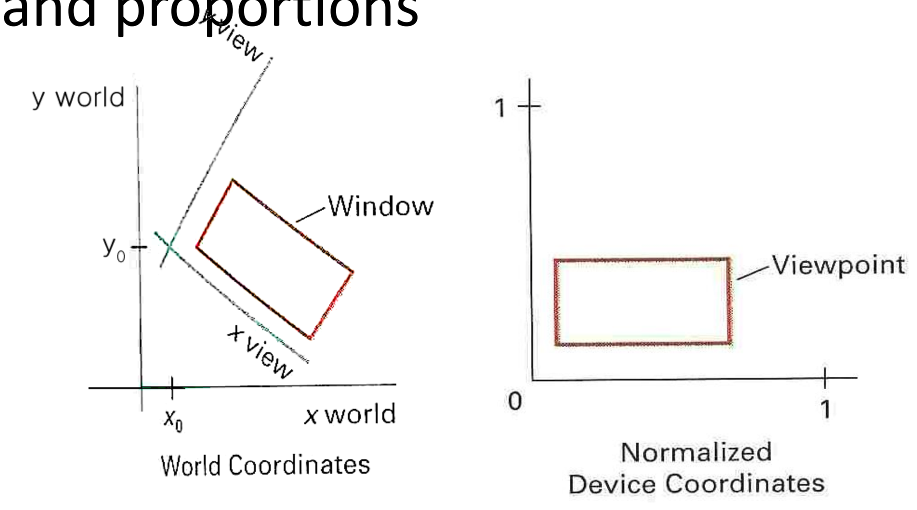
Introduction

- **viewing-transformation in several steps**
  - construct the world-coordinate scene
  - transform descriptions in world coordinates to viewing coordinates
  - map the viewing-coordinate description of the scene to normalized coordinates
  - transfer to device coordinates



Introduction

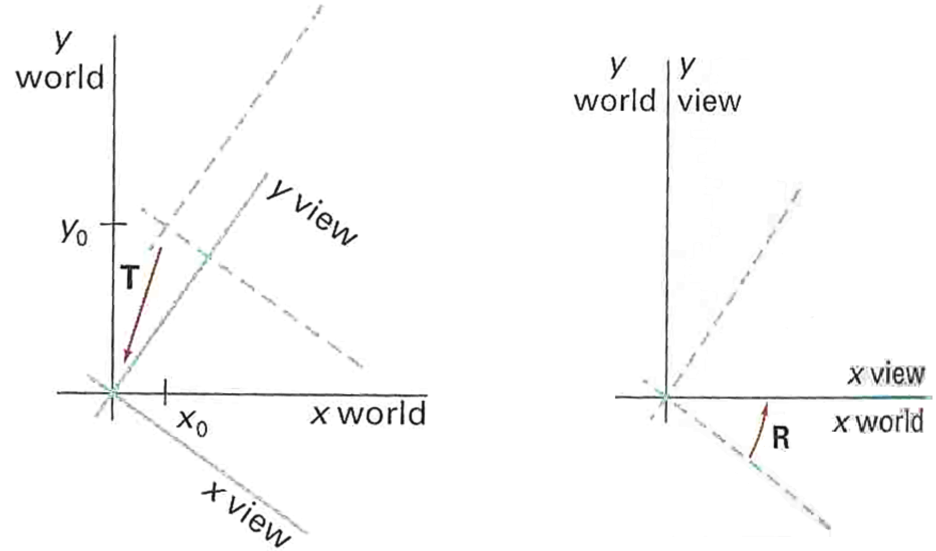
- **viewing-transformation**
  - by changing the position of the viewport
    - ✓can view objects at different positions on the display area of an output device
  - by varying the size of viewports
    - ✓can change the size and proportions of displayed objects
    - ✓zooming effects



Viewing coordinate reference frame

- **The composite 2D transformation to convert world coordinates to viewing coordinates**

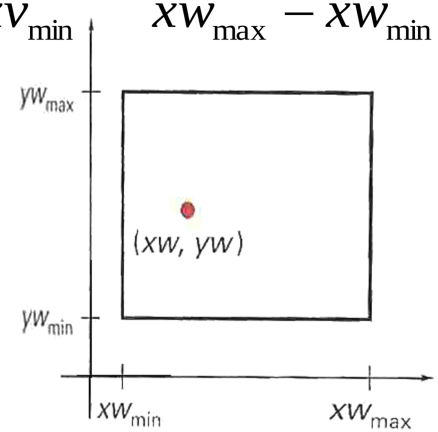
$$\mathbf{M}_{WC,VC} = \mathbf{R} \cdot \mathbf{T}$$

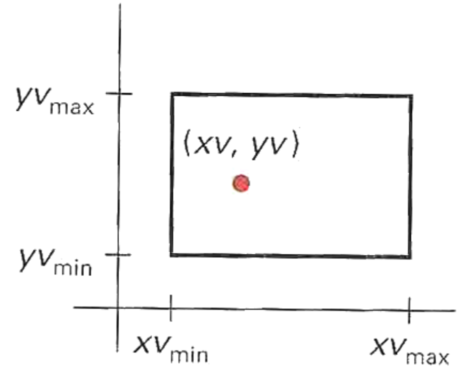


Window-to-viewport coordinate transformation

- **transfer to the viewing reference frame**
  - choose the window extents in viewing coordinate
  - select the viewport limits in normalized coordinate
- **to maintain the same relative placement in the viewport as in the window**

$$\frac{xv - xv_{\min}}{xv_{\max} - xv_{\min}} = \frac{xw - xw_{\min}}{xw_{\max} - xw_{\min}} \qquad \frac{yv - yv_{\min}}{yv_{\max} - yv_{\min}} = \frac{yw - yw_{\min}}{yw_{\max} - yw_{\min}}$$





- **Thus**

$$xv = xv_{\min} + (xw - xw_{\min})sx$$

$$yv = yv_{\min} + (yw - yw_{\min})sy$$

Where,

$$sx = \frac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}}$$

$$sy = \frac{yv_{\max} - yv_{\min}}{yw_{\max} - yw_{\min}}$$

Window-to-viewport coordinate transformation

- Eight coordinate values that define the window and the viewport are just constants.
- Express these two formulas for computing (vx,vy) from (wx,wy) in terms of a translate-scale-translate transformation N.

$$\begin{pmatrix} vx \\ vy \\ 1 \end{pmatrix} = \begin{pmatrix} wx \\ wy \\ 1 \end{pmatrix}$$

- where

$$N = \begin{bmatrix} 1 & 0 & xv_{\min} \\ 0 & 1 & yv_{\min} \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} \dfrac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}} & 0 & 0 \\ 0 & \dfrac{xv_{\max} - xv_{\min}}{xw_{\max} - xw_{\min}} & 0 \\ 0 & 0 & 1 \end{bmatrix} . \begin{bmatrix} 1 & 0 & -xw_{\min} \\ 0 & 1 & -yw_{\min} \\ 0 & 0 & 1 \end{bmatrix}$$

Clipping Operations

- Clipping
  - Any procedure that identifies those portions of a picture that are either inside or outside of a specified region of space
- Applied in World Coordinates
- Adapting Primitive Types
  - Point
  - Line
  - Area (or Polygons)
  - Curve

Point Clipping

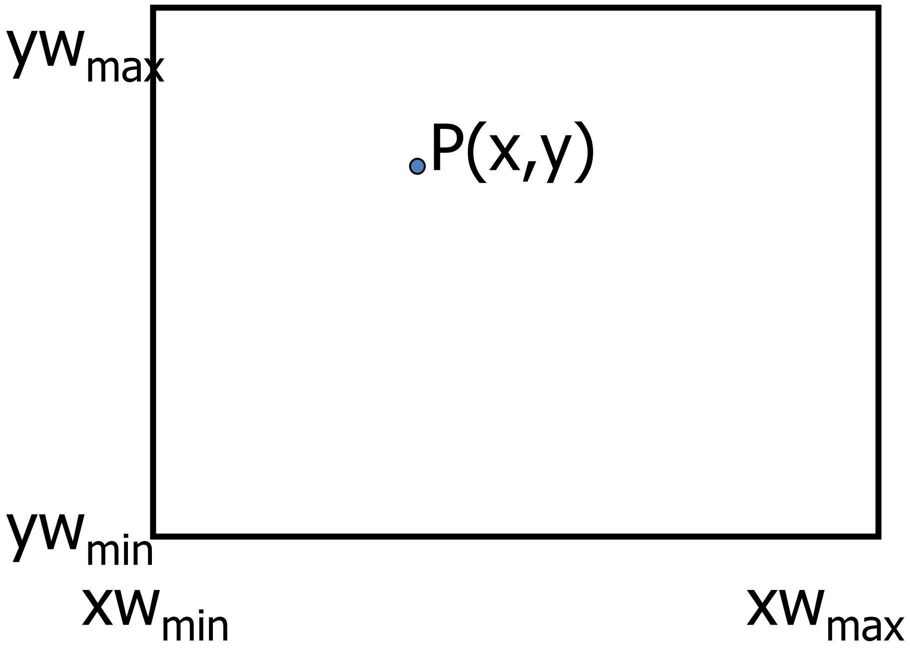
- Assuming that the clip window is a rectangle in standard position
- For a clipping rectangle in standard position, we save a 2-D point P(x,y) for display if the following inequalities are satisfied:

$$x_{\min} \leq x \leq x_{\max}$$

$$y_{\min} \leq y \leq y_{\max}$$

- If any one of these four inequalities is not satisfied, the point is clipped (not saved for display)
- Where $x_{\min}, x_{\max}, y_{\min}, y_{\max}$ define the clipping window.

Point Clipping



If P(x,y) is inside the window?

$$xw_{\min} \leq x \leq xw_{\max}$$

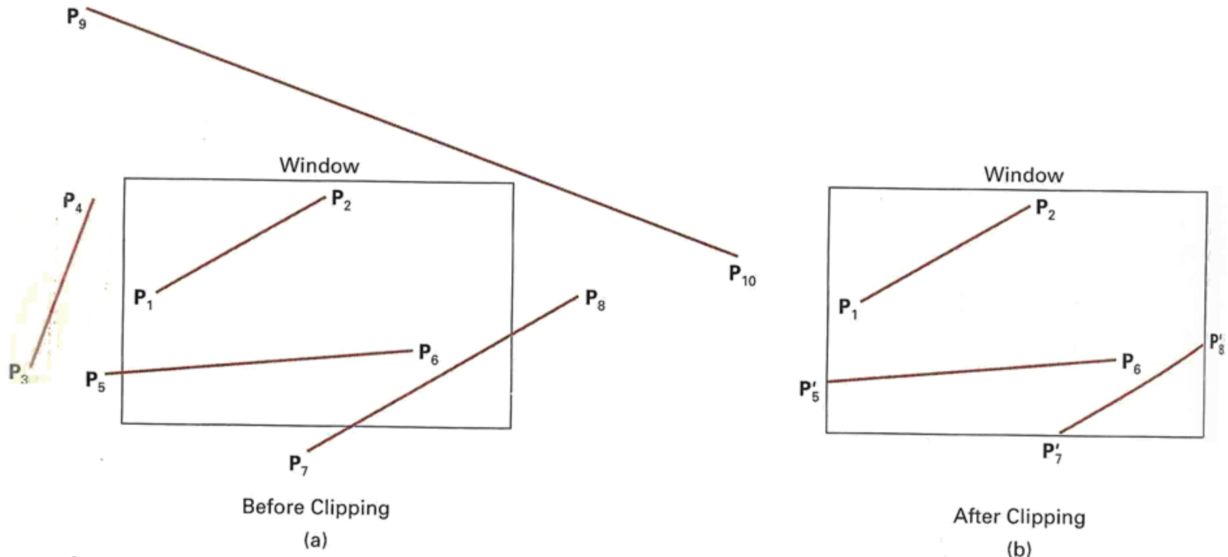
$$yw_{\min} \leq y \leq yw_{\max}$$

Line clipping

- **Line clipping procedure**
  - test a given line segment to determine whether it lies completely inside the clipping window
  - if it doesn't, we try to determine whether it lies completely outside the window
  - if we can't identify a line as completely inside or completely outside, we must perform intersection calculations with one or more clipping boundaries

Line clipping

- Checking the line endpoints ⇒ inside-outside test



- **Line clipping**
  - **Cohen-Sutherland line clipping**
  - **Liang-Barsky line clipping**

Cohen-Sutherland Algorithm

- Divide the line clipping process into two phases:
  - Identify those lines which intersect the clipping window and so need to be clipped.
  - Perform the clipping
- All lines fall into one of the following clipping categories:
  - Visible: Both end points of the line lie within the window.
  - Not visible: The line definitely lies outside the window. This will occur if the line from (x1,y1) to (x2,y2) satisfies any one of the following inequalities:

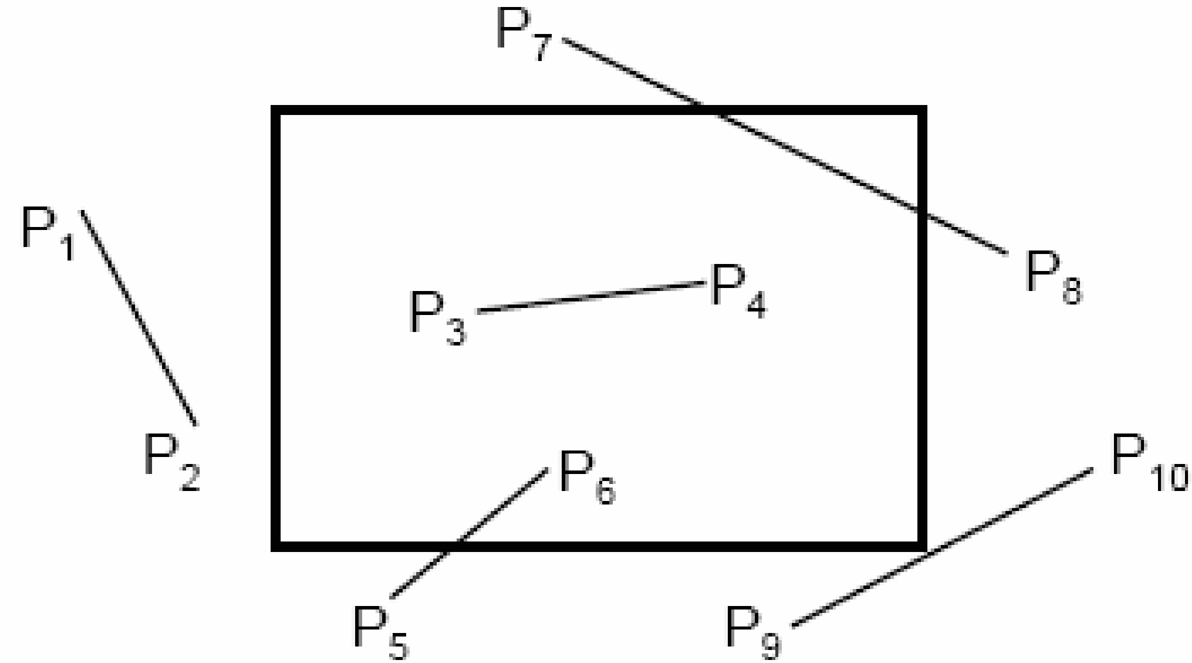
$$x_{1,} x_2 > x_{\max} \qquad y_1, y_2 > y_{\max}$$

$$x_{1,} x_2 < x_{\min} \qquad y_1, y_2 < y_{\min}$$

  - Clipping candidate: the line is in neither category 1 nor 2

Cohen-Sutherland Algorithm

Find the part of a line inside the clip window

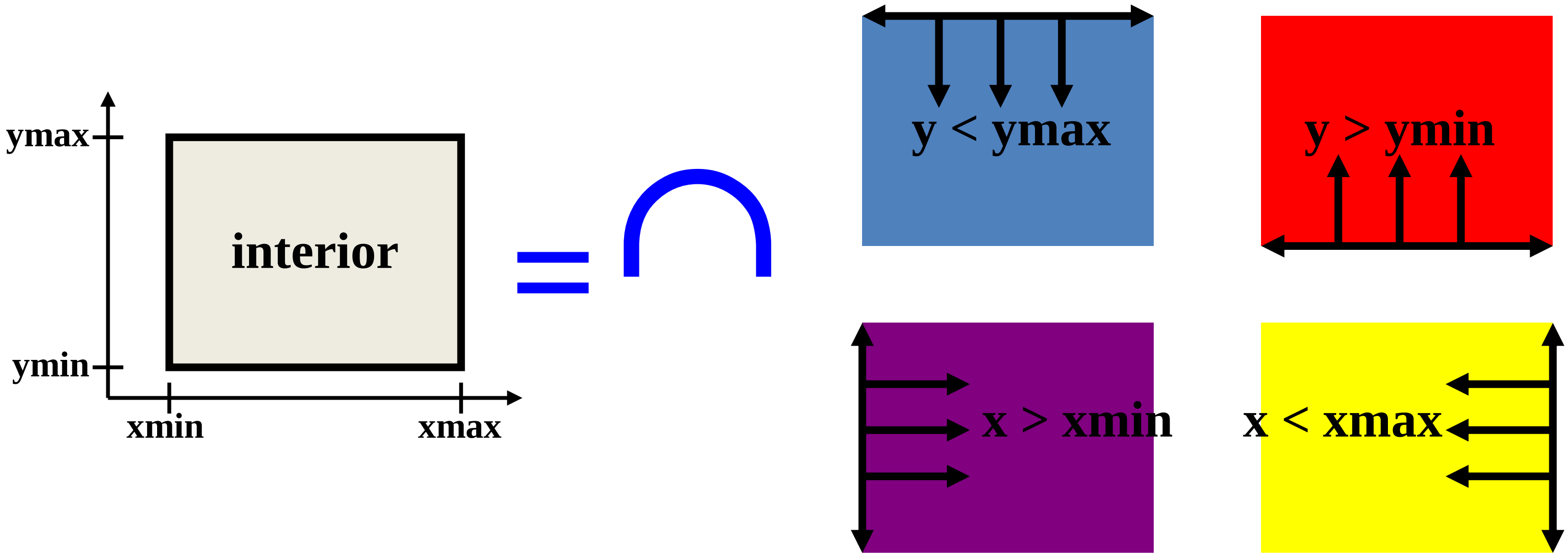


$p_3 p_4$ is in category 1(Visible)

$p_1 p_2$ is in category 2(Not Visible)

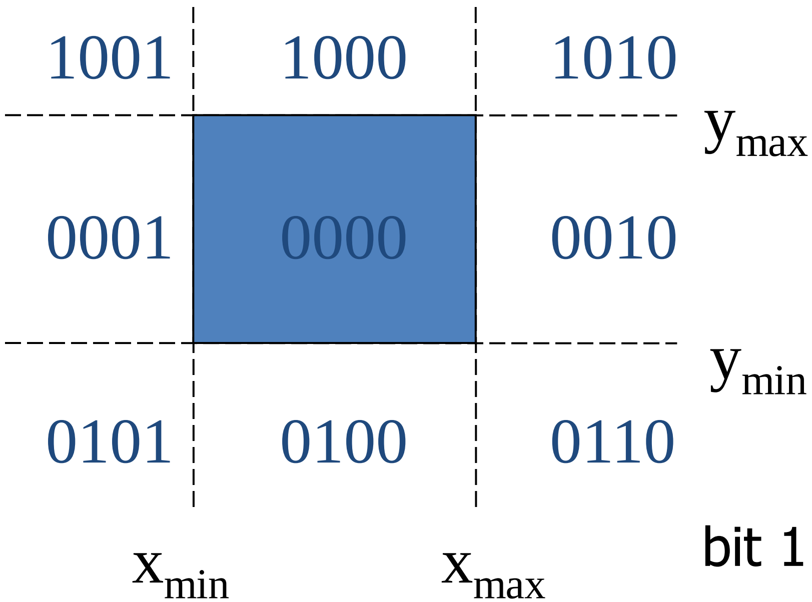
$p_5 p_6, p_7 p_8, p_9 p_{10}$ is in category 3(Clipping candidate)

Cohen-Sutherland Algorithm



Cohen-Sutherland Algorithm

- Assign a four-bit pattern (Region Code) to each endpoint of the given segment. The code is determined according to which of the following nine regions of the plane the endpoint lies in.

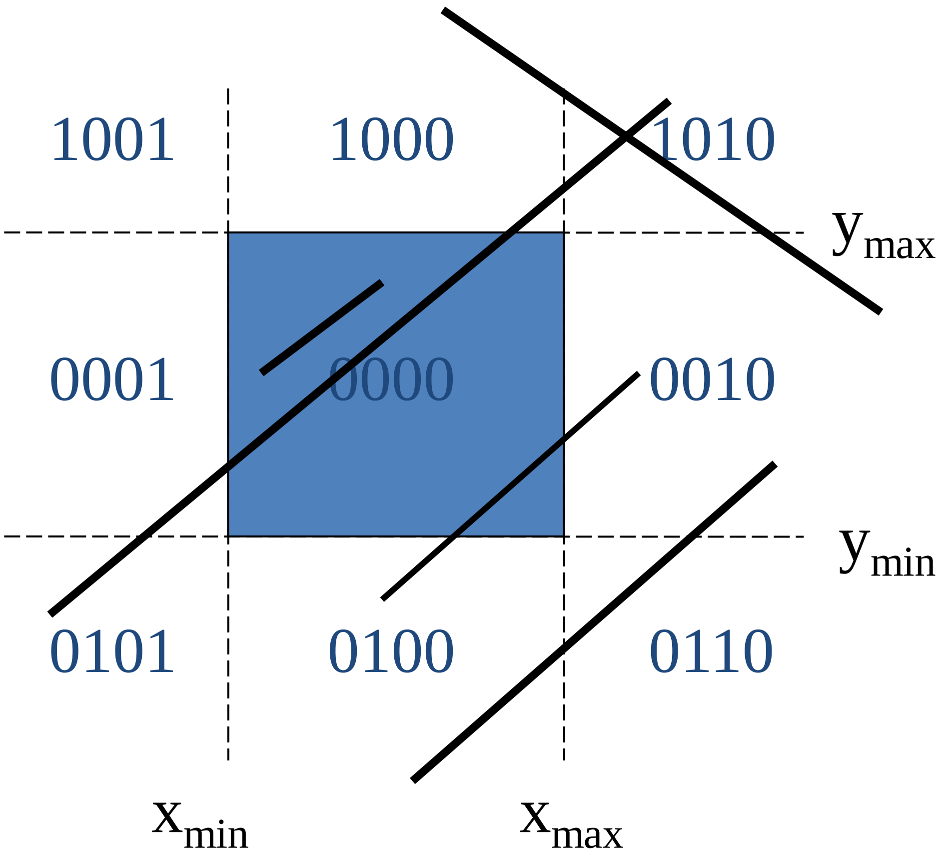


bit 1 : bit 2 : bit 3 : bit 4

Top : Bottom : Right : Left:

- Of course, a point with code 0000 is inside the window.

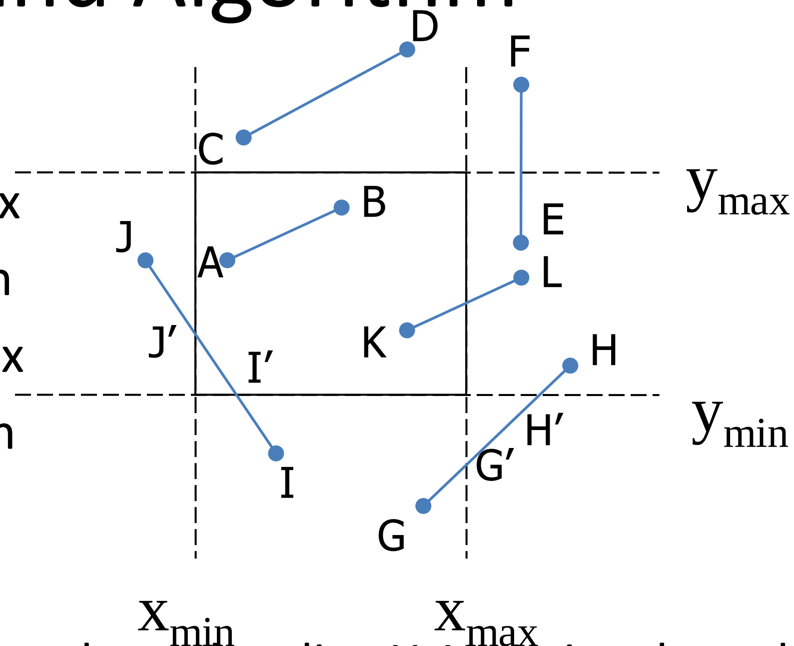
Cohen-Sutherland Algorithm



Cohen-Sutherland Algorithm

- if both endpoint codes are 0000 → the line segment is visible (inside).
- the logical AND of the two endpoint codes
  - not completely 0000 → the line segment is not visible (outside)
  - completely 0000 → the line segment maybe inside (and outside)
- Lines that cannot be identified as being completely inside or completely outside a clipping window are then checked for intersection with the window border lines.

Cohen-Sutherland Algorithm



- Consider code of an end point
  - if bit 1 is 1, intersect with line y = Ymax
  - if bit 2 is 1, intersect with line y = Ymin
  - if bit 3 is 1, intersect with line x = Xmax
  - if bit 4 is 1, intersect with line x = Xmin

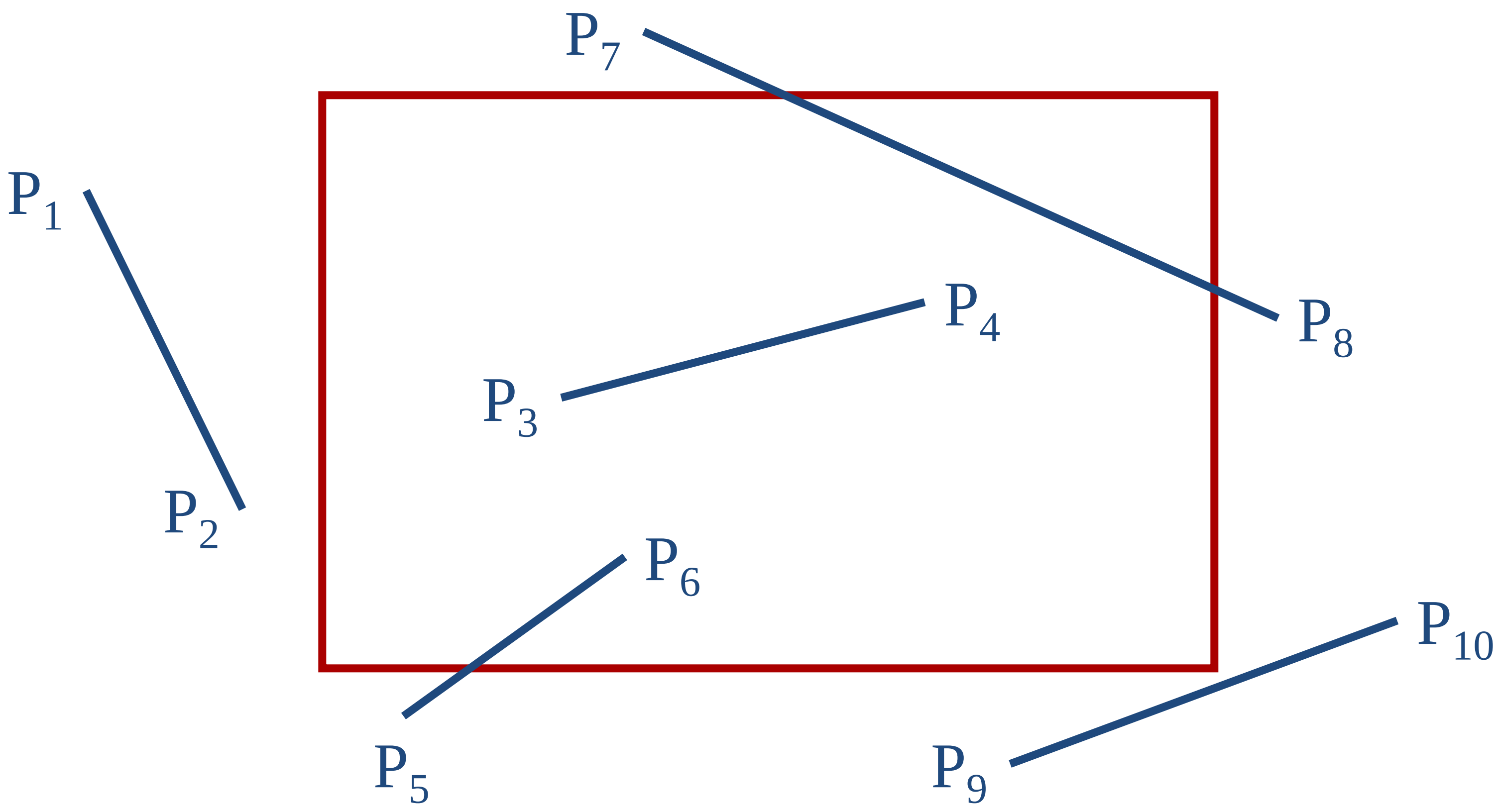
- Consider line CD.
  - If endpoint C is chosen, then the bottom boundary line Y=Ymin is selected for computing intersection
  - If endpoint D is chosen, then either the top boundary line Y=Ymax or the right boundary line X=Xmax is used.
  - The coordinates of the intersection point are:
    - y = y0 + m(x-x0)
    - x = Xmax or Xmin if the boundary line is vertical or
    - x = x0 + 1/m(y-y0) Xmin if the boundary line is horizontal
    - Y= Ymax or Ymin , Where $m = \dfrac{y_{end} - y_0}{x_{end} - x_0}$

Cohen-Sutherland Algorithm

- Replace endpoint (x1,y1) with the intersection point(xi,yi), effectively eliminating the portion of the original line that is on the outside of the selected window boundary.
- The new endpoint is then assigned an updated region code and the clipped line re-categoriged and handled in the same way.
- This iterative process terminates when we finally reach a clipped line that belongs to either category 1(visible) or category 2(not visible).

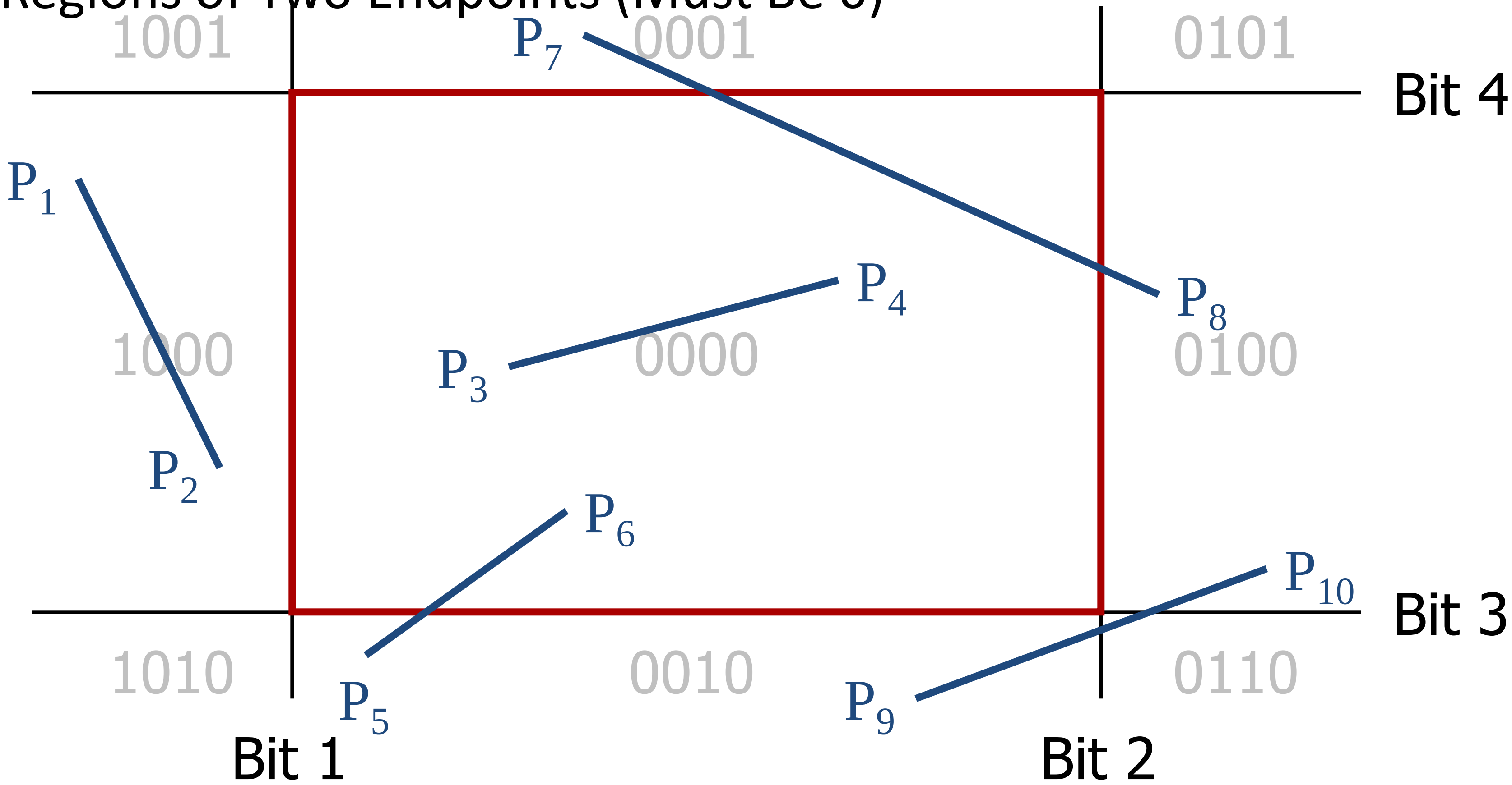
Cohen-Sutherland Line Clipping

- Use Simple Tests to Classify Easy Cases First



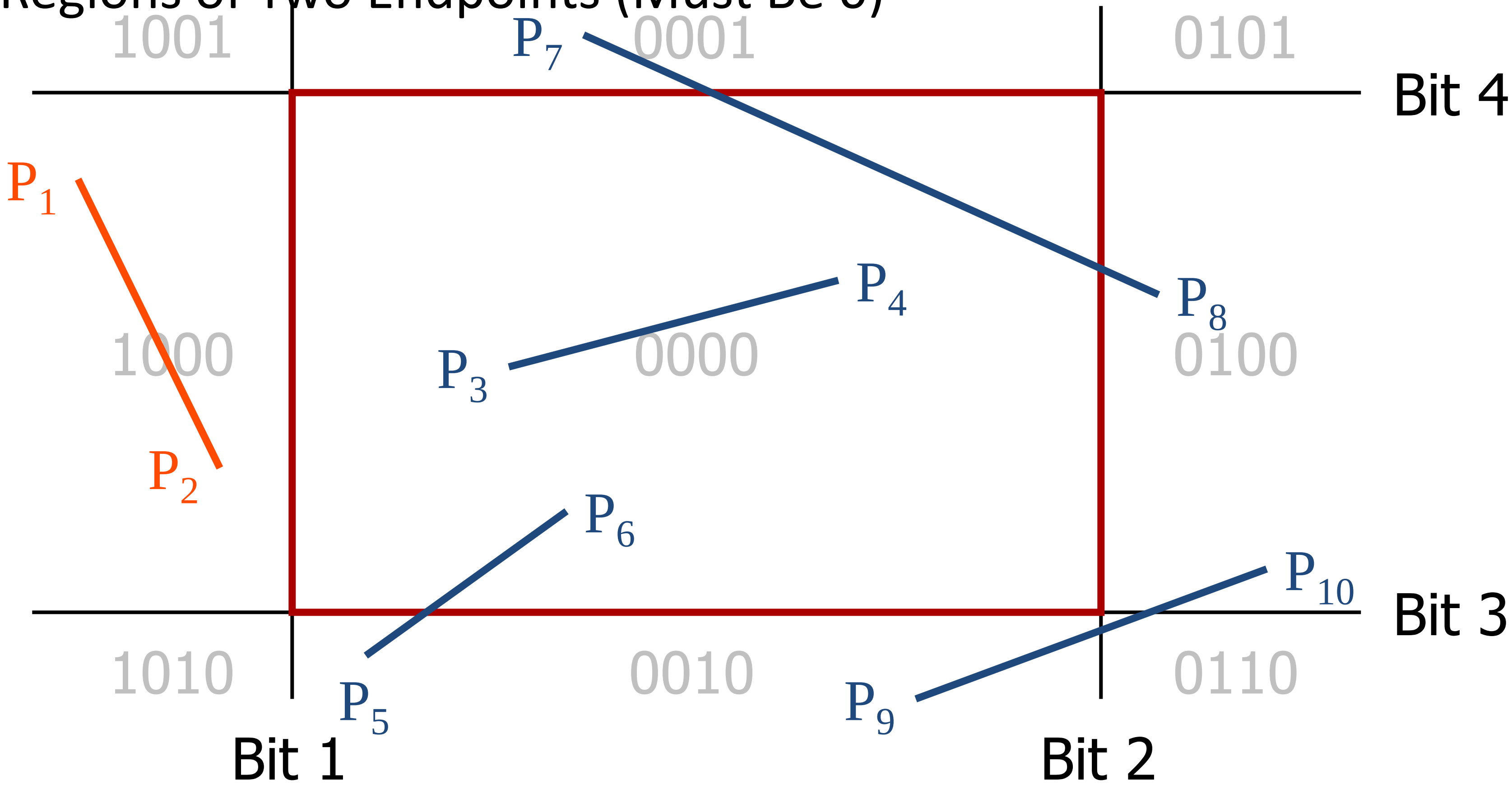
Cohen-Sutherland Line Clipping

- Classify Some Lines Quickly by AND of Bit Codes Representing Regions of Two Endpoints (Must Be 0)



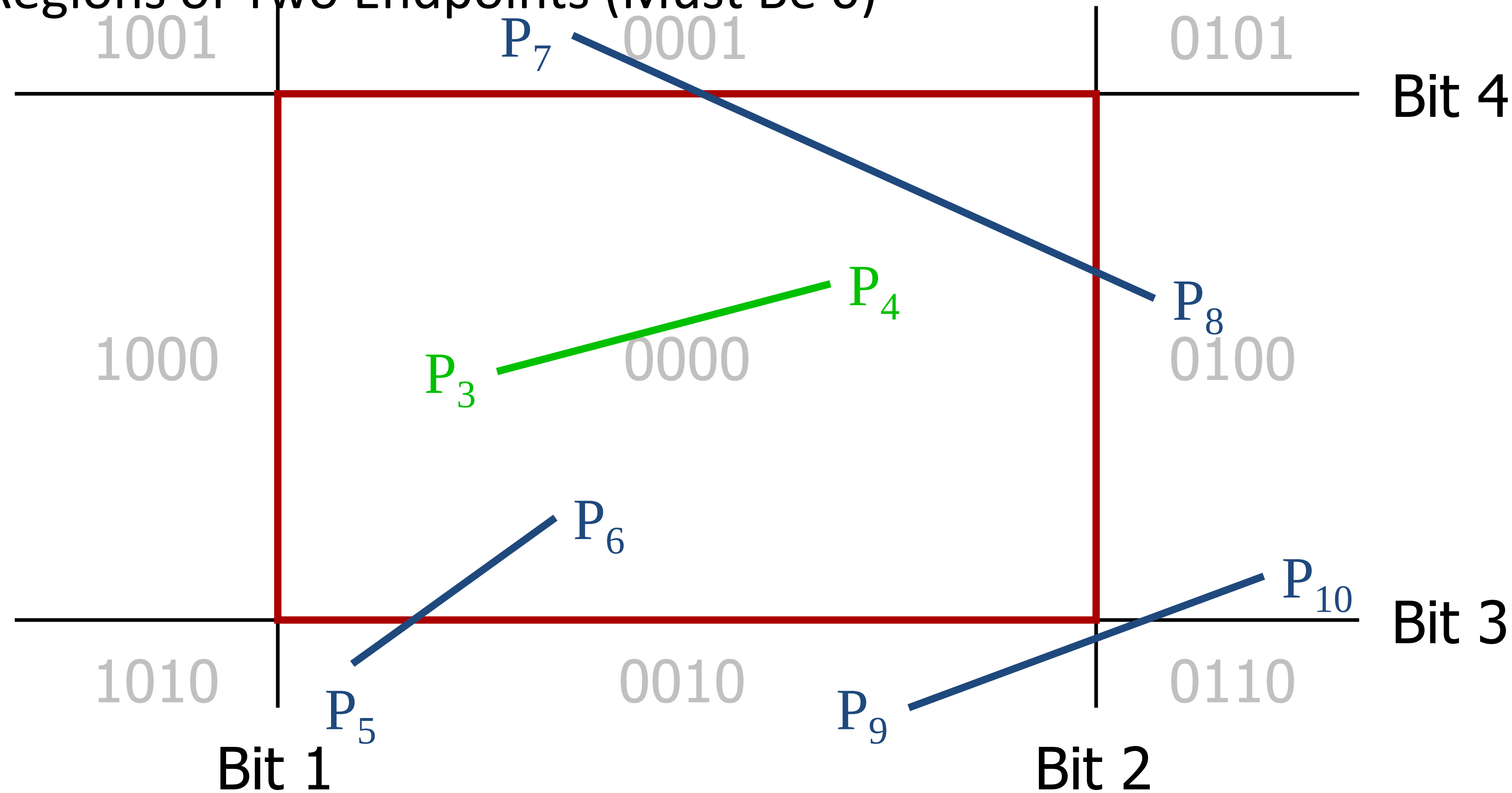
Cohen-Sutherland Line Clipping

- Classify Some Lines Quickly by AND of Bit Codes Representing Regions of Two Endpoints (Must Be 0)



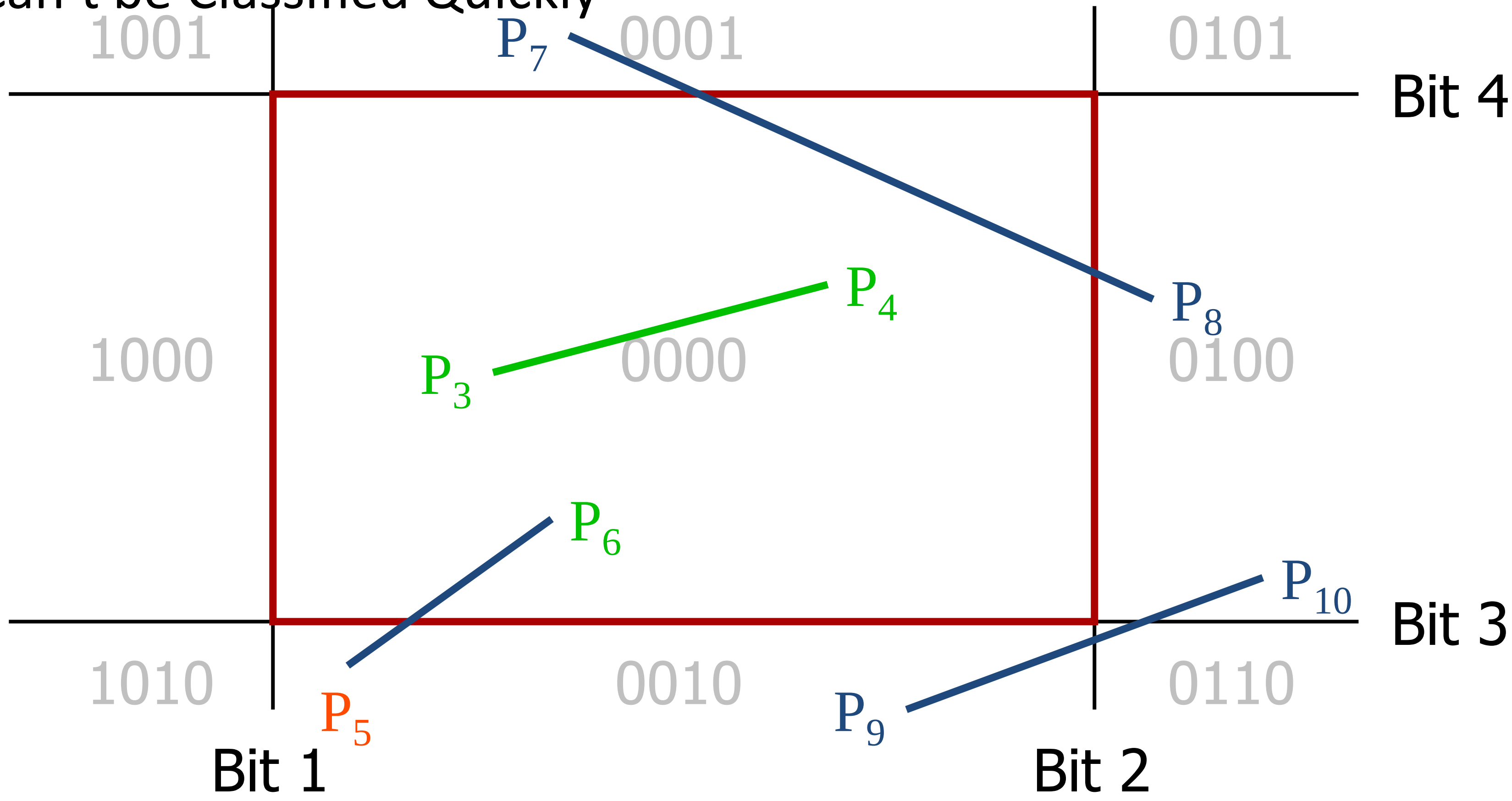
Cohen-Sutherland Line Clipping

- Classify Some Lines Quickly by AND of Bit Codes Representing Regions of Two Endpoints (Must Be 0)



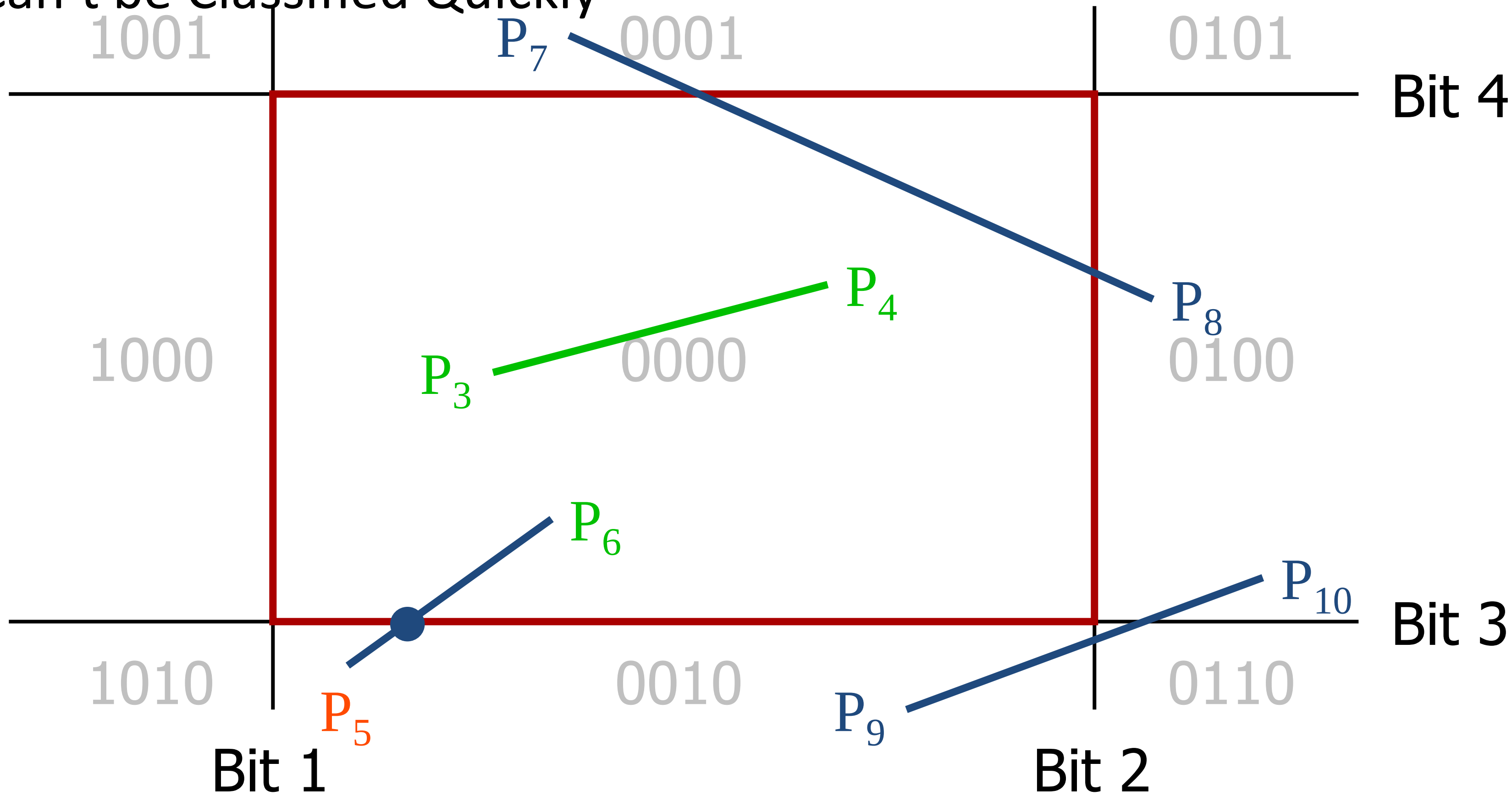
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



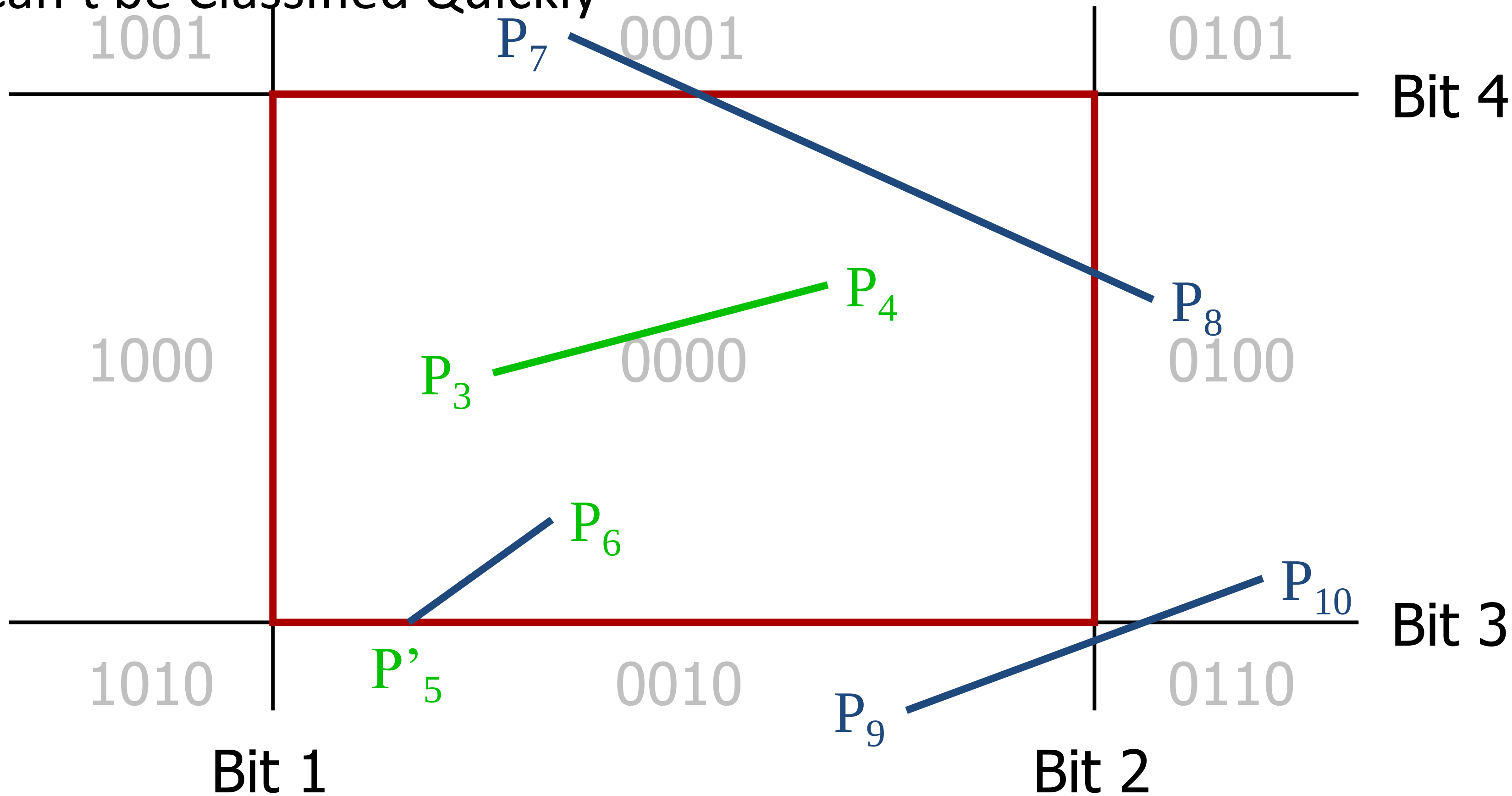
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



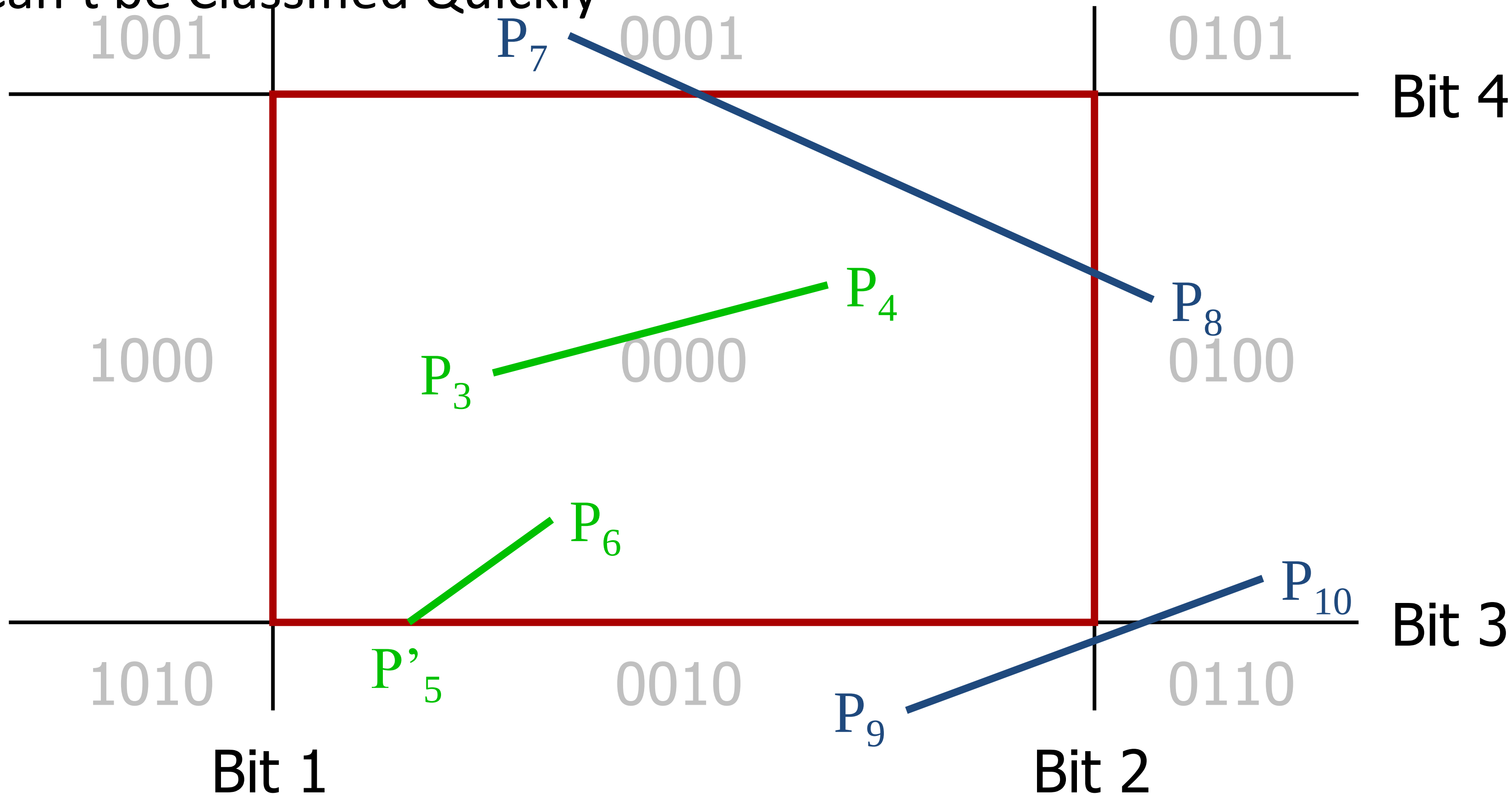
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can’t be Classified Quickly



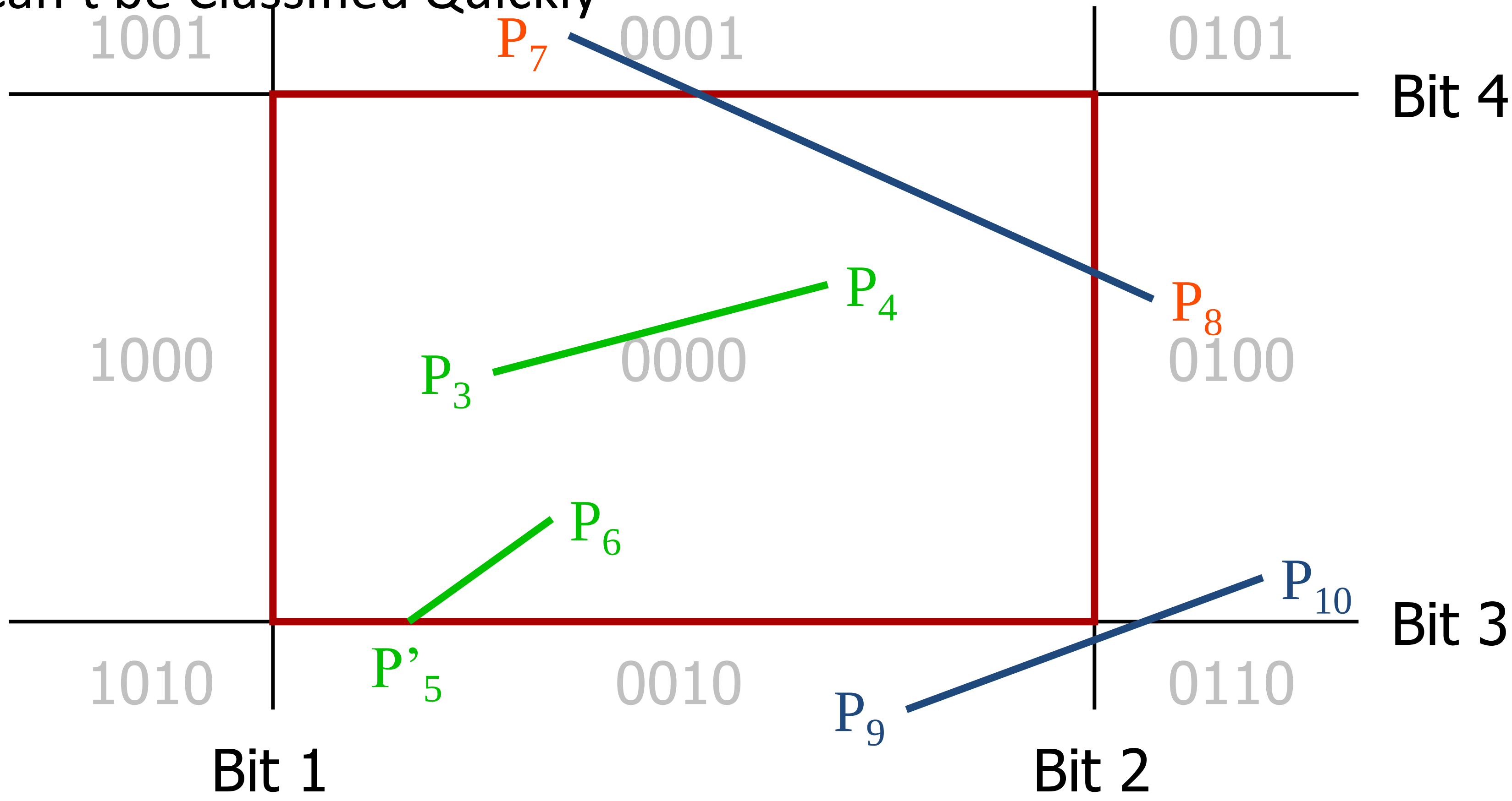
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can’t be Classified Quickly



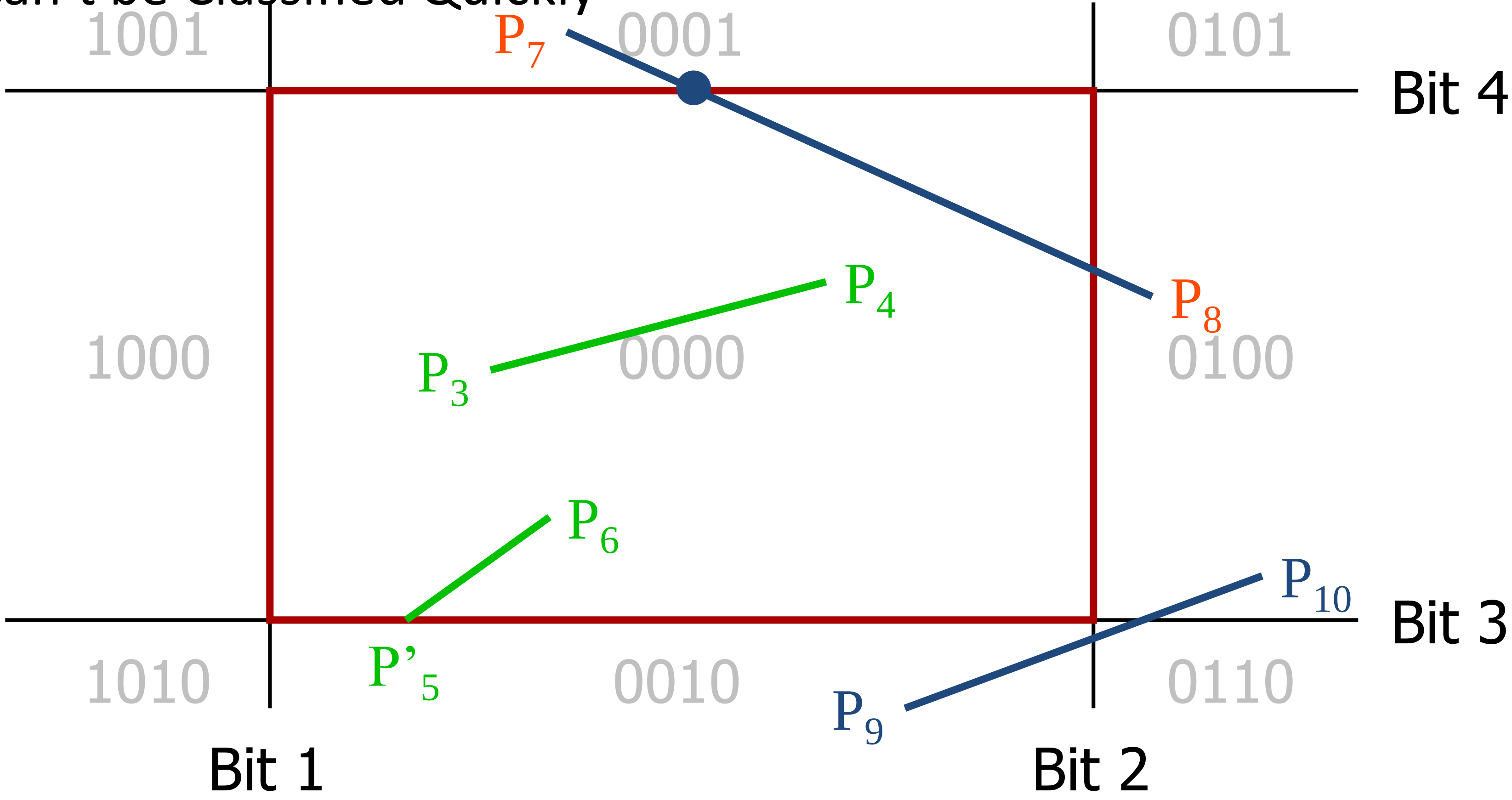
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



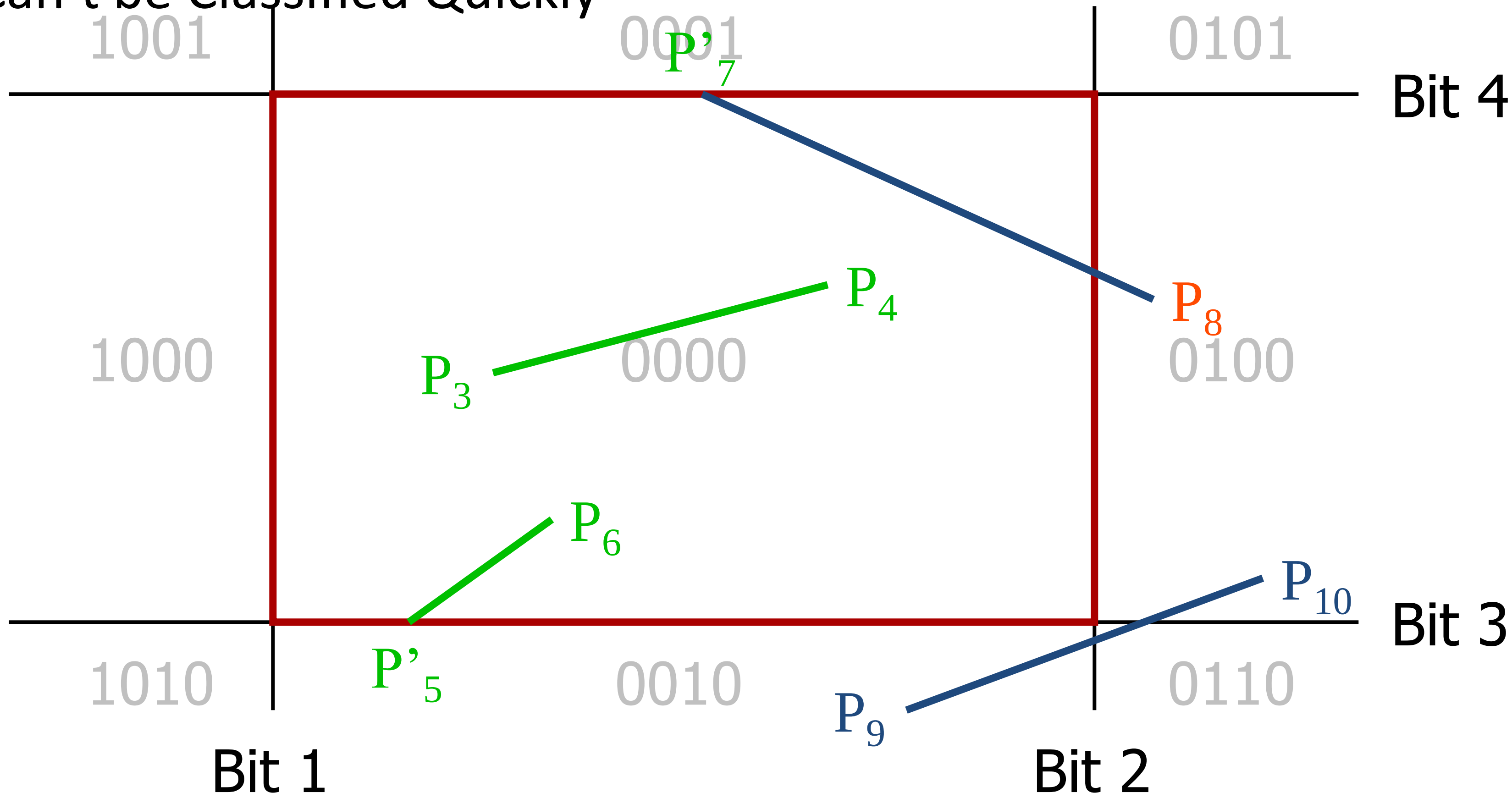
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



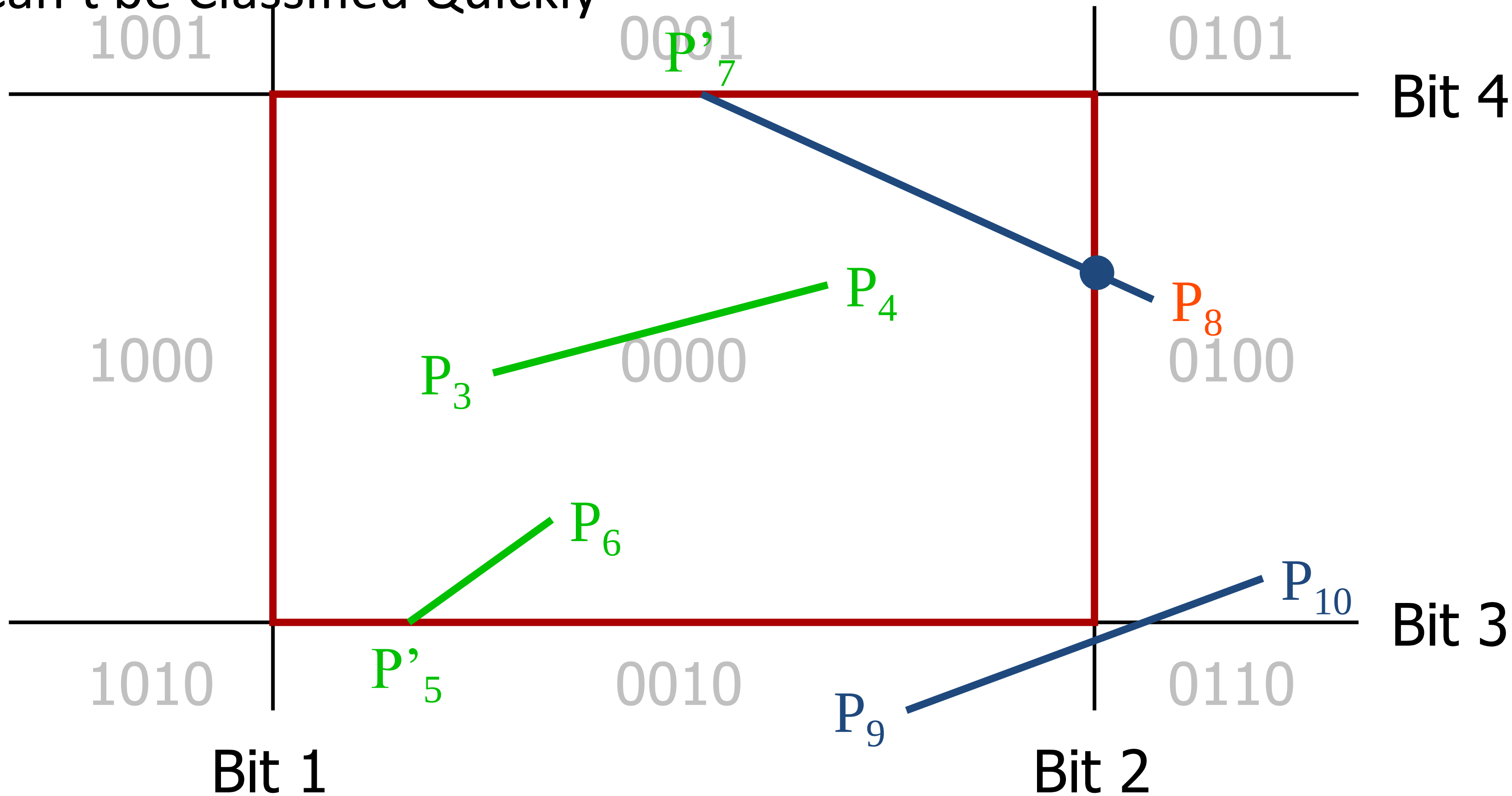
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



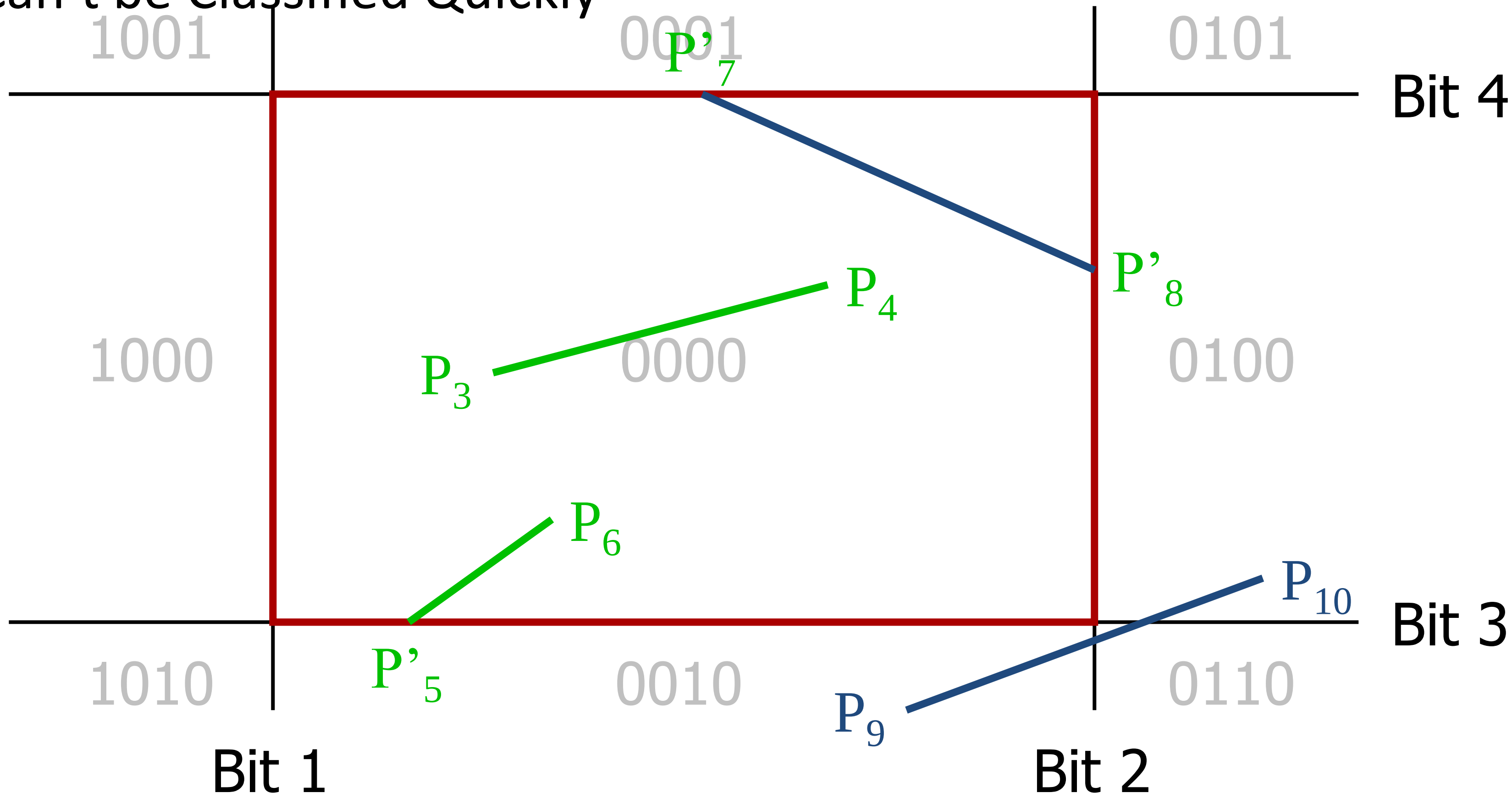
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can’t be Classified Quickly



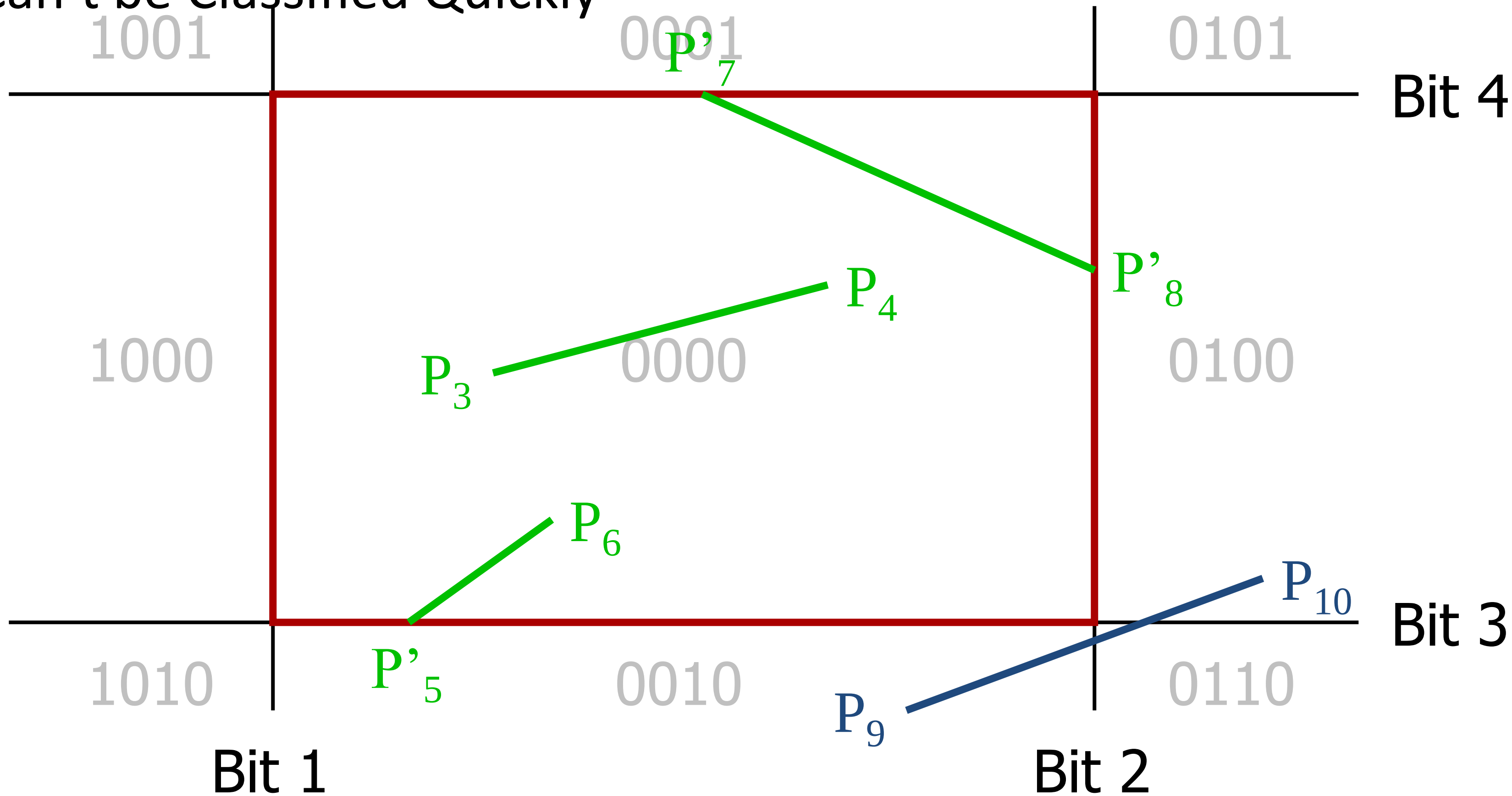
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can’t be Classified Quickly



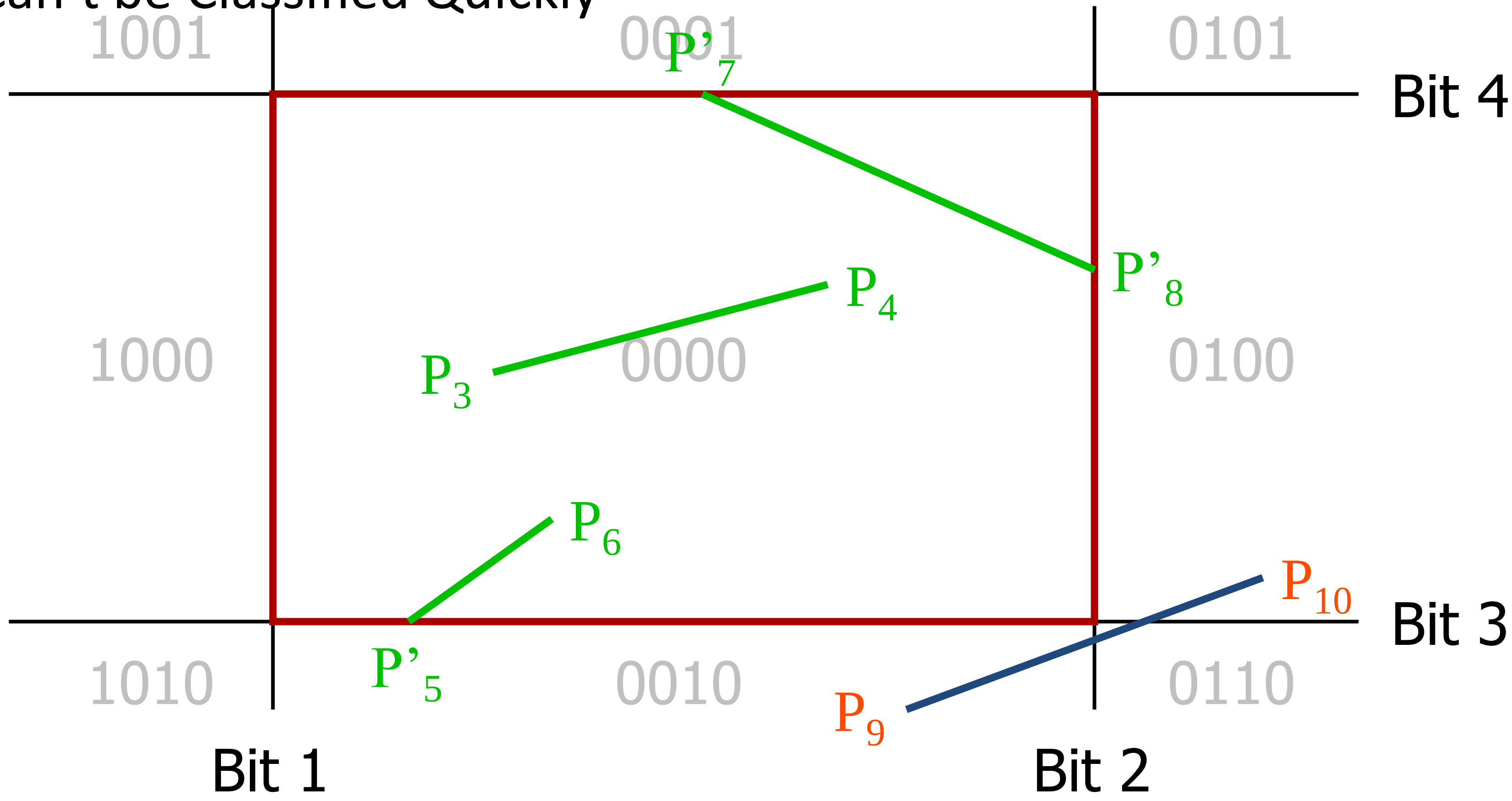
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



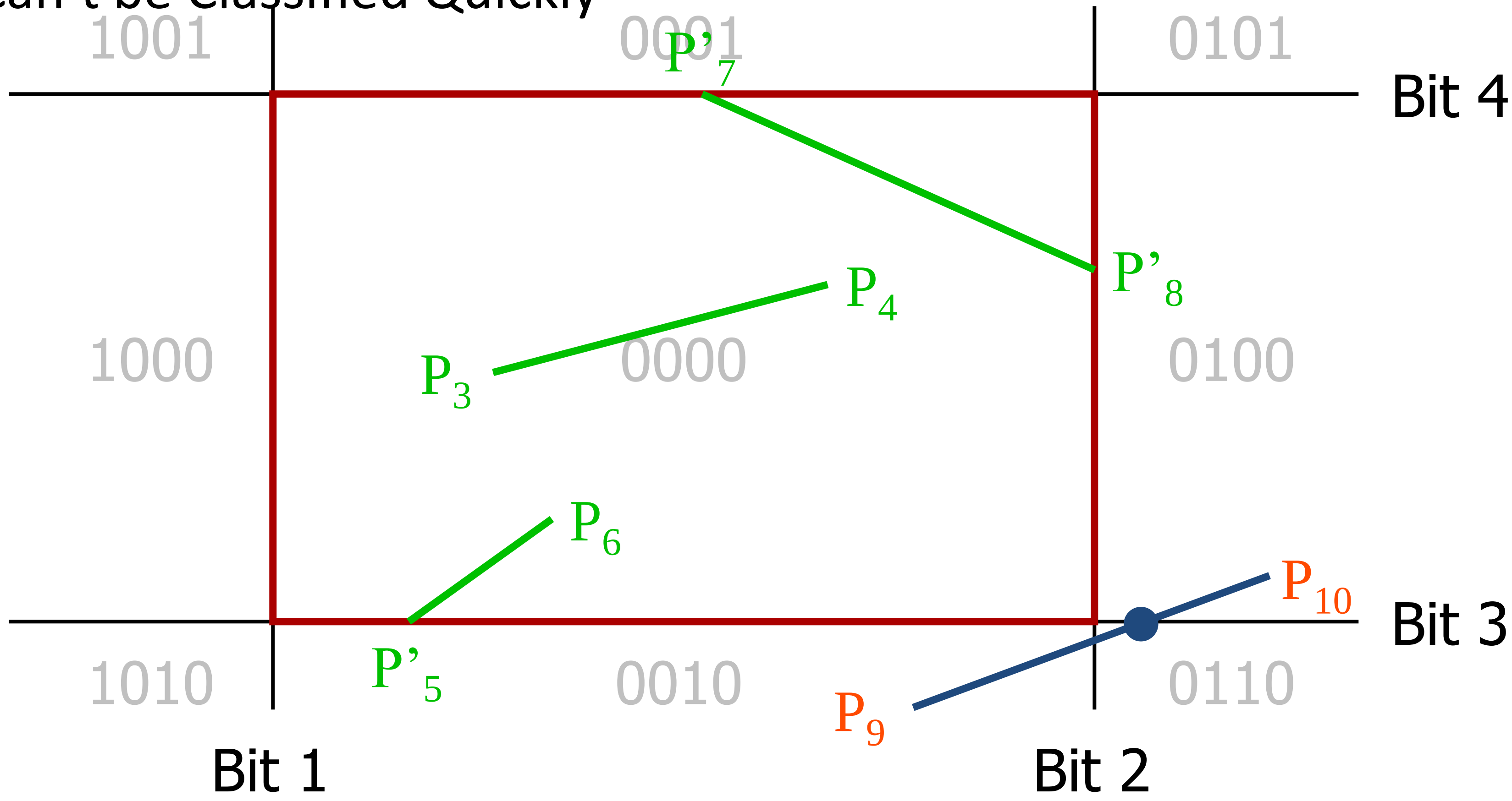
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



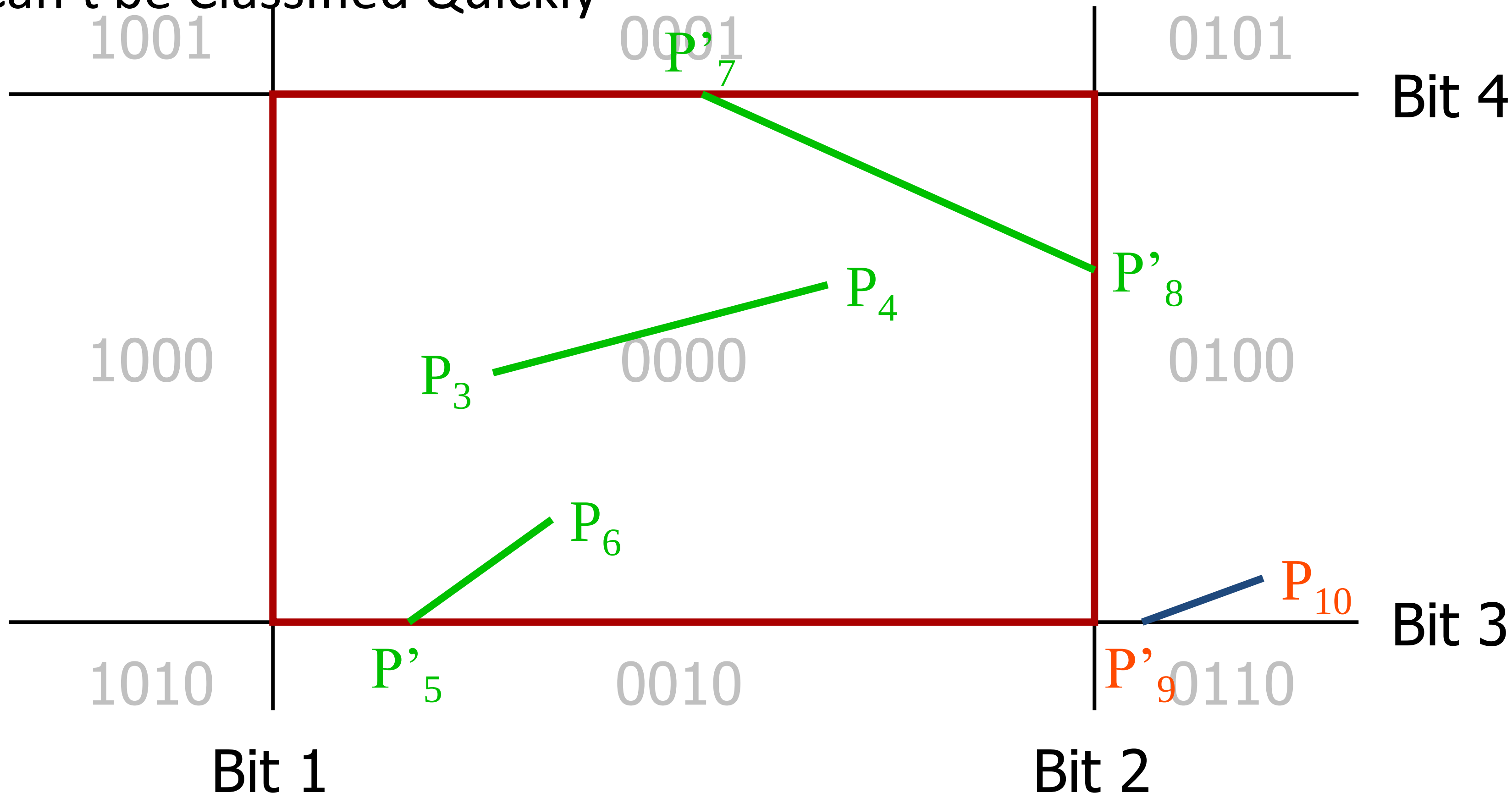
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



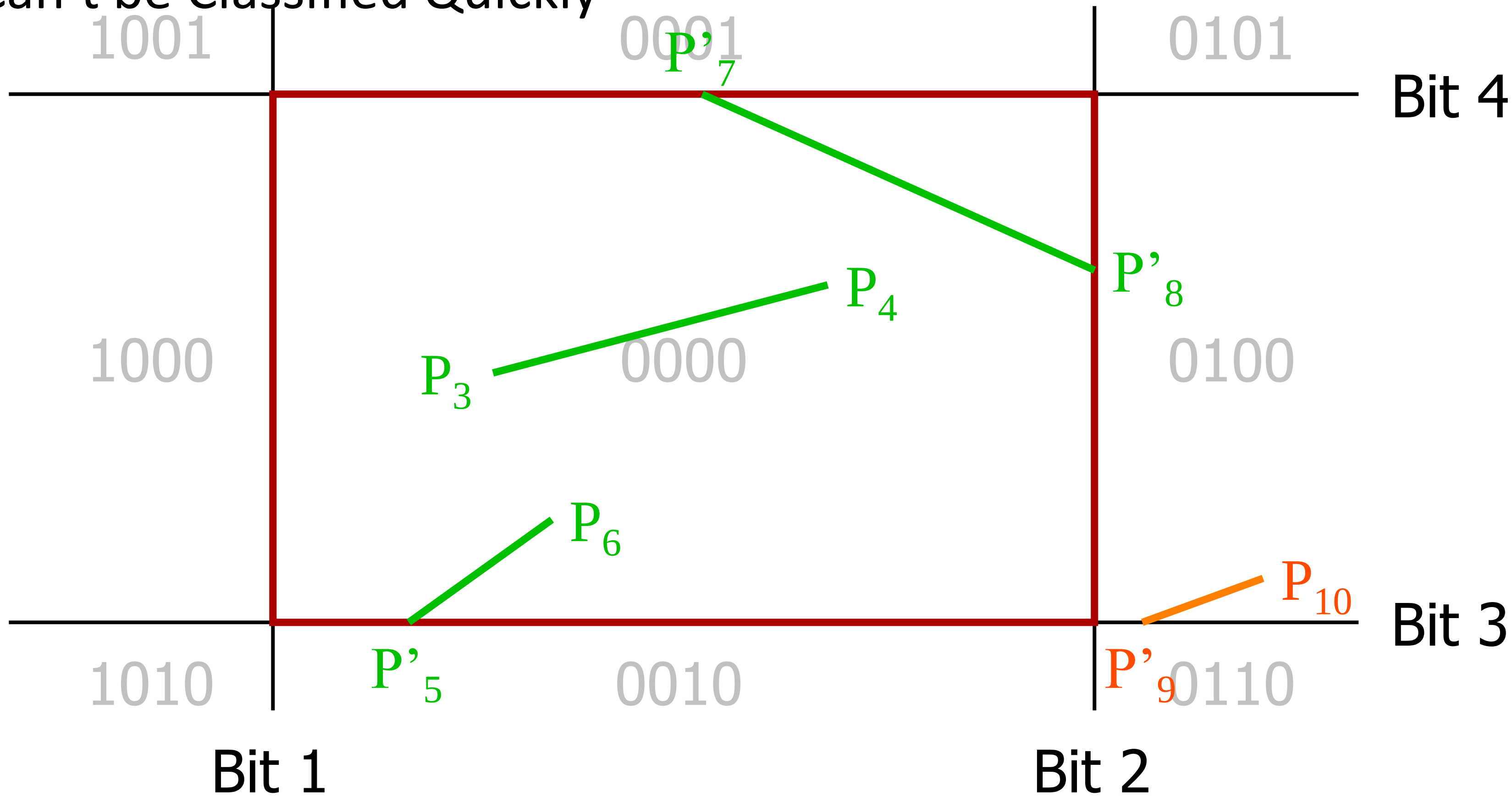
Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly



Cohen-Sutherland Line Clipping

- Compute Intersections with Window Boundary for Lines That Can't be Classified Quickly