

NCTU-EE IC LAB – Fall 2023

Lab02 Exercise

Design: Calculation on the coordinates

Data Preparation

1. Extract test data from TA's directory:
% tar -xvf ~iclabTA01/Lab02.tar
2. The extracted Lab02/ directory contains:
 - a. Practice
 - b. Exercise

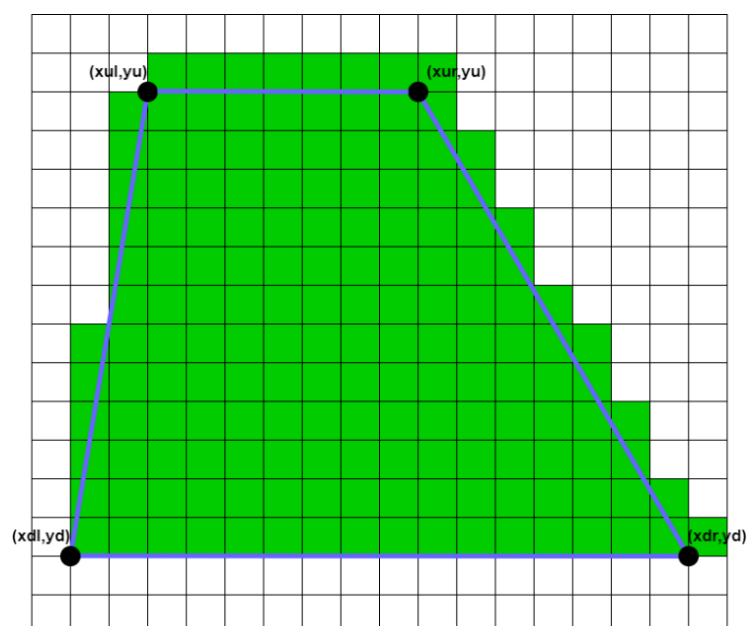
Design Description

Please design a circuit that supports three modes on a coordinate:

- (1) Trapezoid rendering.
- (2) Circle and line relationships.
- (3) Area computing.

Trapezoid rendering (Mode 0)

When the “in_valid” is at high level, pattern will send out the four sets of coordinates for the **trapezoid** in the order of (xul, yu), (xur, yu), (xdl, yd), (xdr, yd). After a period of circuit operations, the “out_valid” will be set to a high level, and the pattern will begin verifying the valid output coordinates (xo, yo) at every negative edge of the clock until “out_valid” is lowered. All valid output coordinates requirements are as follows:



In this design, you are required to **output the coordinates covered by the trapezoid.**

The output coordinate regulations:

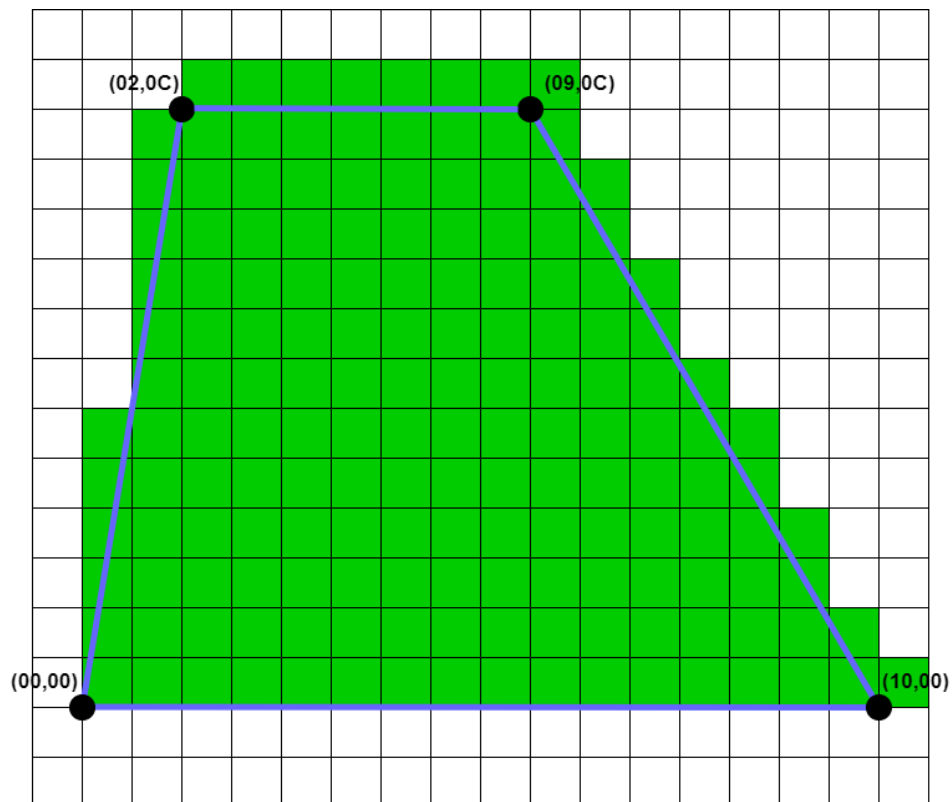
- (1) If the bottom left coordinate or bottom right coordinate of a square is covered by the trapezoid, please output the coordinates of the bottom left corner of that square.
- (2) If the trapezoid exactly covers the bottom left coordinate of the square, please output the coordinates of the bottom left corner of that square.
- (3) The output order is from left to right, from bottom to up.

Here is an example:

Assume you receive four sets of coordinates in order from the pattern as follows:

$(x_{ul}, y_u) = (02, 0C)$, $(x_{ur}, y_u) = (09, 0C)$, $(x_{dl}, y_d) = (00, 00)$, $(x_{dr}, y_d) = (10, 00)$.

The valid output is: $(00, 00)$ $(01, 00)$... $(10, 00)$, $(00, 01)$ $(01, 01)$... $(0F, 01)$, $(00, 02)$ $(01, 02)$... $(0E, 02)$, $(00, 03)$ $(01, 03)$... $(0E, 03)$, $(00, 04)$ $(01, 04)$... $(0D, 04)$, $(00, 05)$ $(01, 05)$... $(0D, 05)$, $(01, 06)$ $(02, 06)$... $(0C, 06)$, $(01, 07)$ $(02, 07)$... $(0B, 07)$, $(01, 08)$ $(02, 08)$... $(0B, 08)$, $(01, 09)$ $(02, 09)$... $(0A, 09)$, $(01, 0A)$ $(02, 0A)$... $(0A, 0A)$, $(01, 0B)$ $(02, 0B)$... $(09, 0B)$, $(02, 0C)$ $(03, 0C)$... $(09, 0C)$.



Circle and line relationships (Mode 1)

The relationships between circles and lines can be categorized into three types:

(1) Tangent, (2) Intersecting, (3) non-intersecting

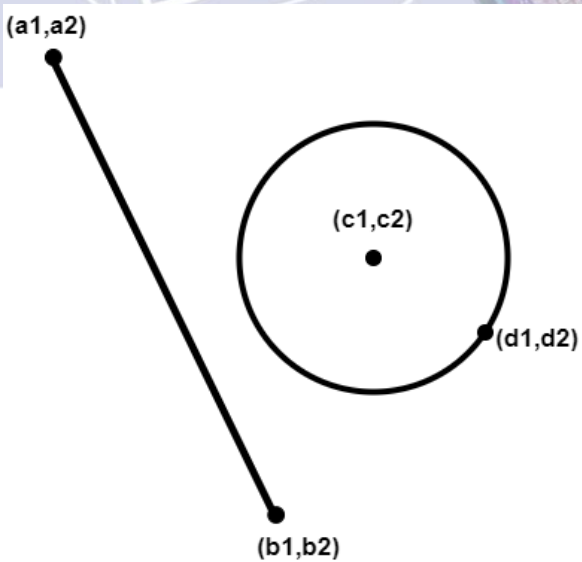
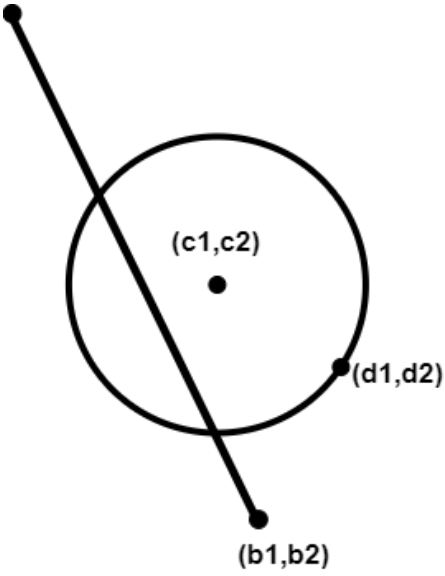
When in_valid is at high level, pattern will send out the four sets of coordinates in the **following order**:

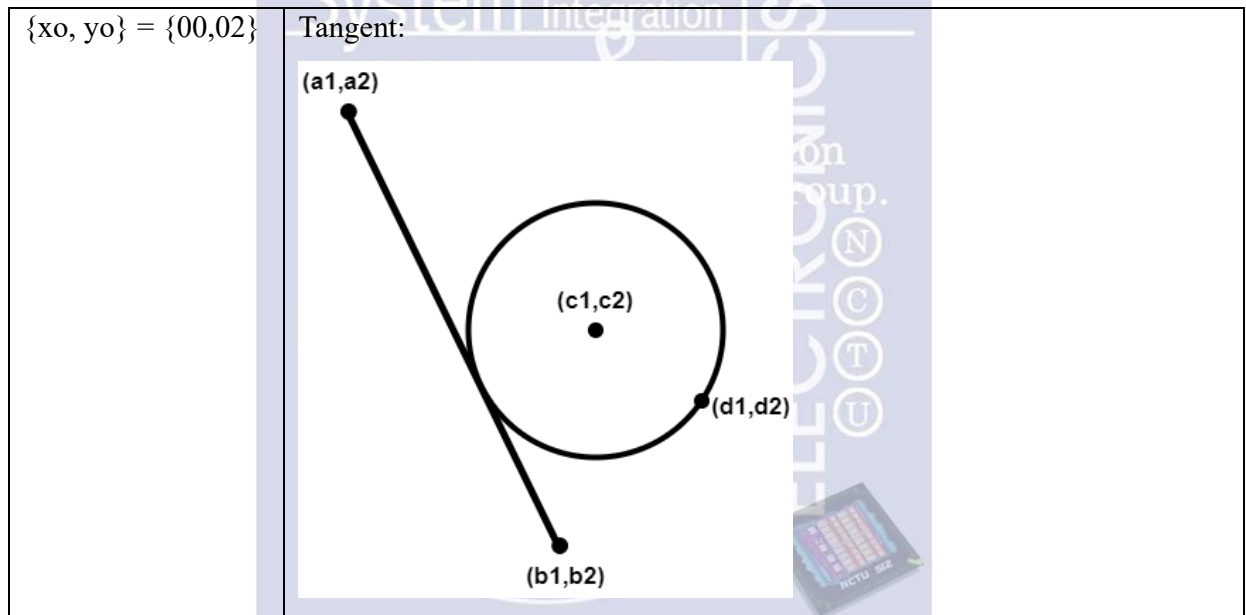
First, you will obtain the two points **(a1, a2)** and **(b1, b2)** on the line.

Next will be the center of the circle **(c1, c2)**, and finally, a point on the circle **(d1, d2)**.

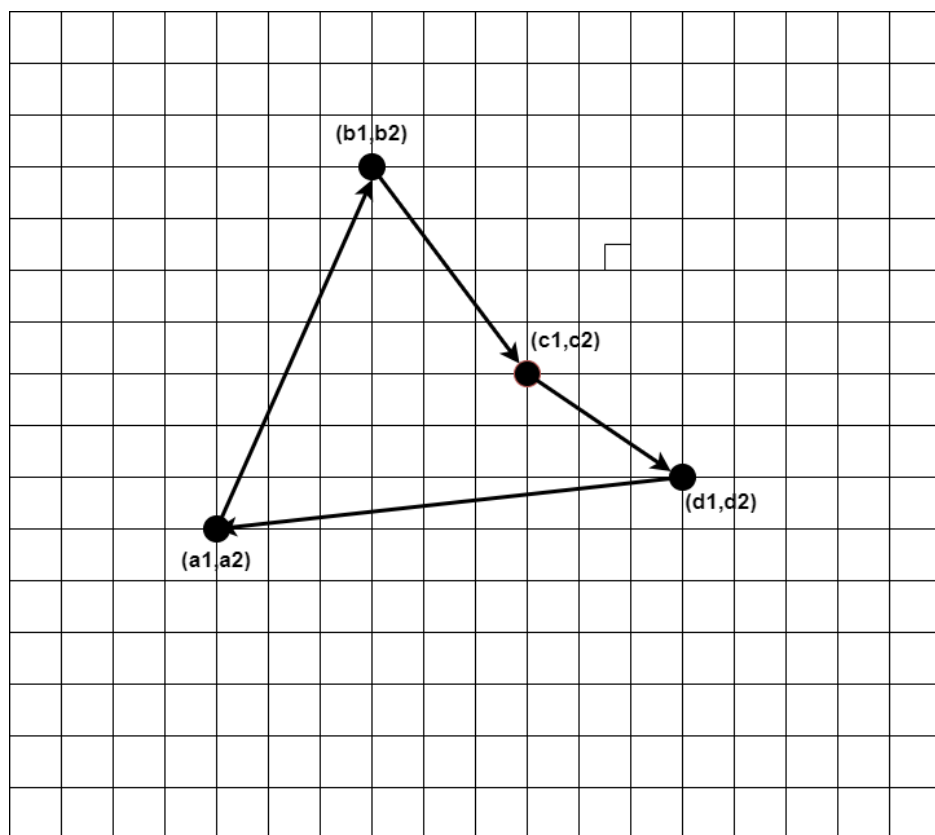
Please find out the relationships between circle and line:

■ To shorten the synthesis time for everyone, the input coordinates here will be limited to 6 bits.

Relation	Description
$\{x_0, y_0\} = \{00, 00\}$	non-intersecting: 
$\{x_0, y_0\} = \{00, 01\}$	Intersecting: 



Area computing (Mode 2)



When in_valid is at high level, pattern will send out the four sets of coordinates in the **following order**:

$(a1, a2) \Rightarrow (b1, b2) \Rightarrow (c1, c2) \Rightarrow (d1, d2)$

Please find out the area of the quadrilateral:

If the final answer has a decimal point, please round down!

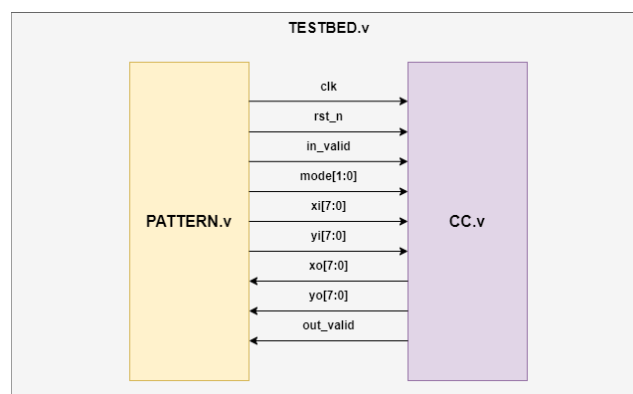
I/O specification

Signals name	Direction	Bit Width	Definition
clk	input	1	Clock.
rst_n	input	1	Asynchronous active-low reset.
mode	input	2	Mode 0: Do trapezoid rendering. Mode 1: Derive the relationships between circle and line. Mode 2: Derive the area.
in_valid	input	1	High when input signals are valid.
xi	input	8	Input of the X coordinate, in two's complement form.
yi	input	8	Input of the Y coordinate, in two's complement form.
out_valid	output	1	High when output is valid.
xo	output	8	Mode 0: Output of the trapezoid X coordinate, in two's complement form. Mode 1: Set to 0. Mode 2: Area [15:8]
yo	output	8	Mode 0: Output of the trapezoid Y coordinate, in two's complement form. Mode 1: Relationships outcome. Mode 2: Area [7:0]

Specifications

1. Top module name : CC (Filename: CC.v)
2. It is an **asynchronous** reset and **active-low** architecture. If you use synchronous reset (reset after clock starting) in your design, you may fail to reset signals.
3. The clock period of the design is fixed to **12ns**.
4. The next group of inputs will come in **2~5** cycles after your out_valid pull down.
5. The synthesis result of data type cannot include any **LATCH**.
6. After synthesis, you can check CC.area and CC.timing in the folder “Report”.
7. The slack in the timing report should be **non-negative** and the result should be **MET**.
8. The gate level simulation cannot include any timing violation.
9. The latency of your design in each pattern should not be larger than **100** cycles. The latency is the clock cycles between the falling edge of the **in_valid** and the rising edge of the **out_valid**.
10. **Any words with “error”, “latch” or “congratulation” can’t be used as variable name.**
11. The **out_valid** cannot overlap **in_valid**.

Block Diagram



Note

1. Grading policy:

RTL and gate-level simulation correctness: 70%

Performance (Area * Execution Cycle): 30%

2. Please submit your design through Lab02/09_SUBMIT/01_SUBMIT

- 1st_demo deadline: 2023/10/02(Mon.) 12:00:00
- 2nd_demo deadline: 2023/10/04(Wed.) 12:00:00
- If uploaded files **violate the naming rule**, you will get **5 deduct points**.

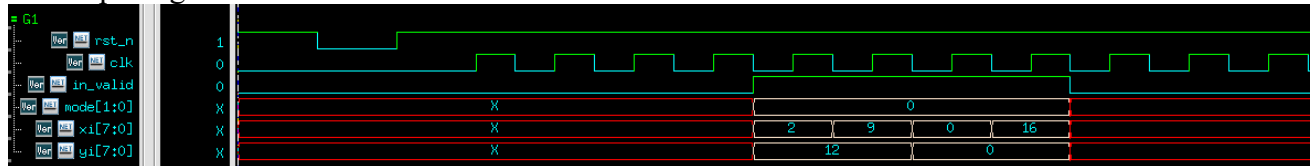
3. Template folders and reference commands:

01_RTL/	(RTL simulation)	<code>./01_run_vcs_rtl</code>
02_SYN/	(Synthesis)	<code>./01_run_dc_shell</code>
(Check the design if there's latch or not in <i>syn.log</i>)		
(Check the design's timing in /Report/ <i>CC.timing</i>)		
03_GATE /	(Gate-level simulation)	<code>./01_run_vcs_gate</code>
09_SUBMIT/(tar all your design)		<code>./00_tar</code>
09_SUBMIT/(submit files)		<code>./01_submit</code>
09_SUBMIT/(check files)		<code>./02_check</code>

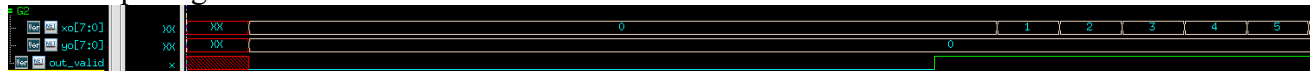
Sample Waveform

■ Trapezoid rendering:

■ Input signal:

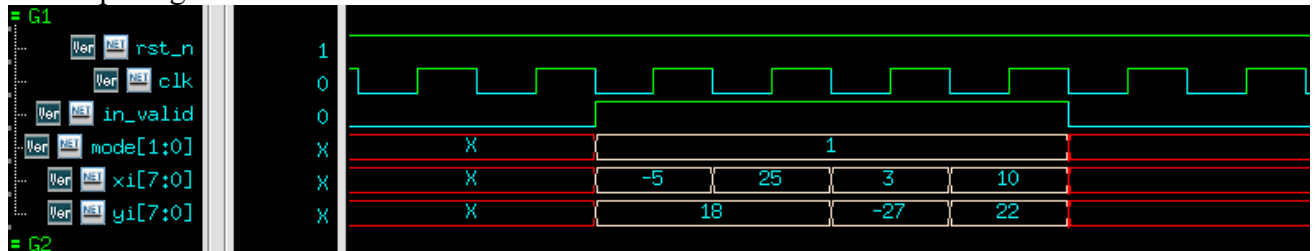


■ Output signal:

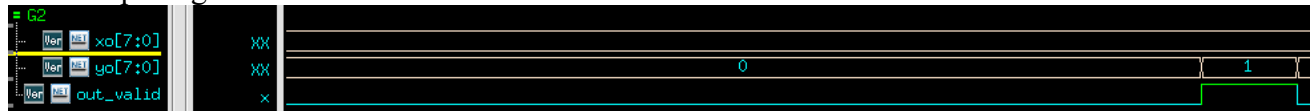


■ Circle and line relationships:

■ Input signal:

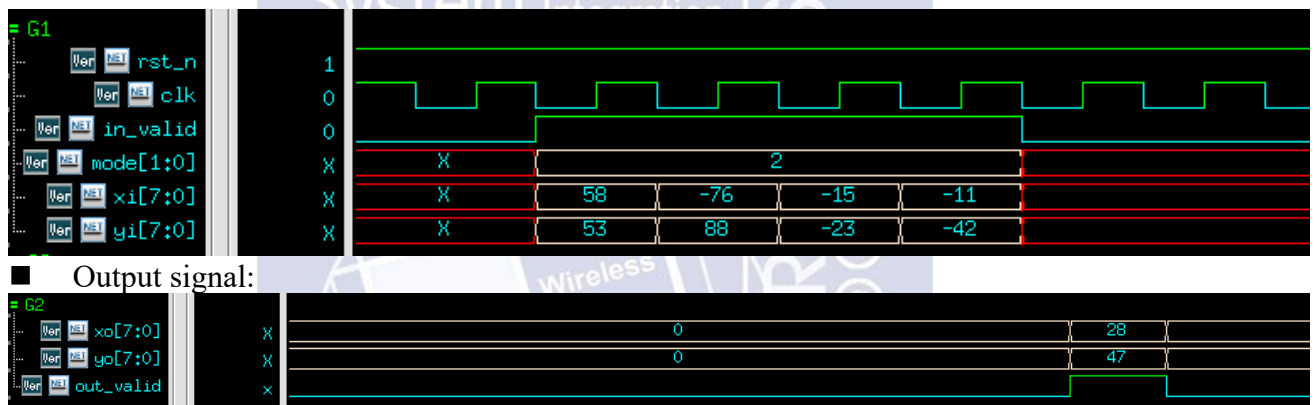


■ Output signal:



■ Area computing:

■ Input signal:



Appendix

1. To find out the distance from a point $P(x_0, y_0)$ to line $L: ax + by + c = 0$, you may need the following equation:

$$d(P, L) = \frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}}$$

2. You may use the Surveyor's formula to compute area

The Surveyor's Formula. If the vertices of a simple polygon, listed counterclockwise around the perimeter, are $(x_0, y_0), (x_1, y_1), \dots, (x_{n-1}, y_{n-1})$, the area of the polygon is

$$A = \frac{1}{2} \left\{ \begin{vmatrix} x_0 & x_1 \\ y_0 & y_1 \end{vmatrix} + \begin{vmatrix} x_1 & x_2 \\ y_1 & y_2 \end{vmatrix} + \dots + \begin{vmatrix} x_{n-2} & x_{n-1} \\ y_{n-2} & y_{n-1} \end{vmatrix} + \begin{vmatrix} x_{n-1} & x_0 \\ y_{n-1} & y_0 \end{vmatrix} \right\}.$$

Note that each oriented edge of the polygon corresponds to a 2×2 determinant in the surveyor's formula.