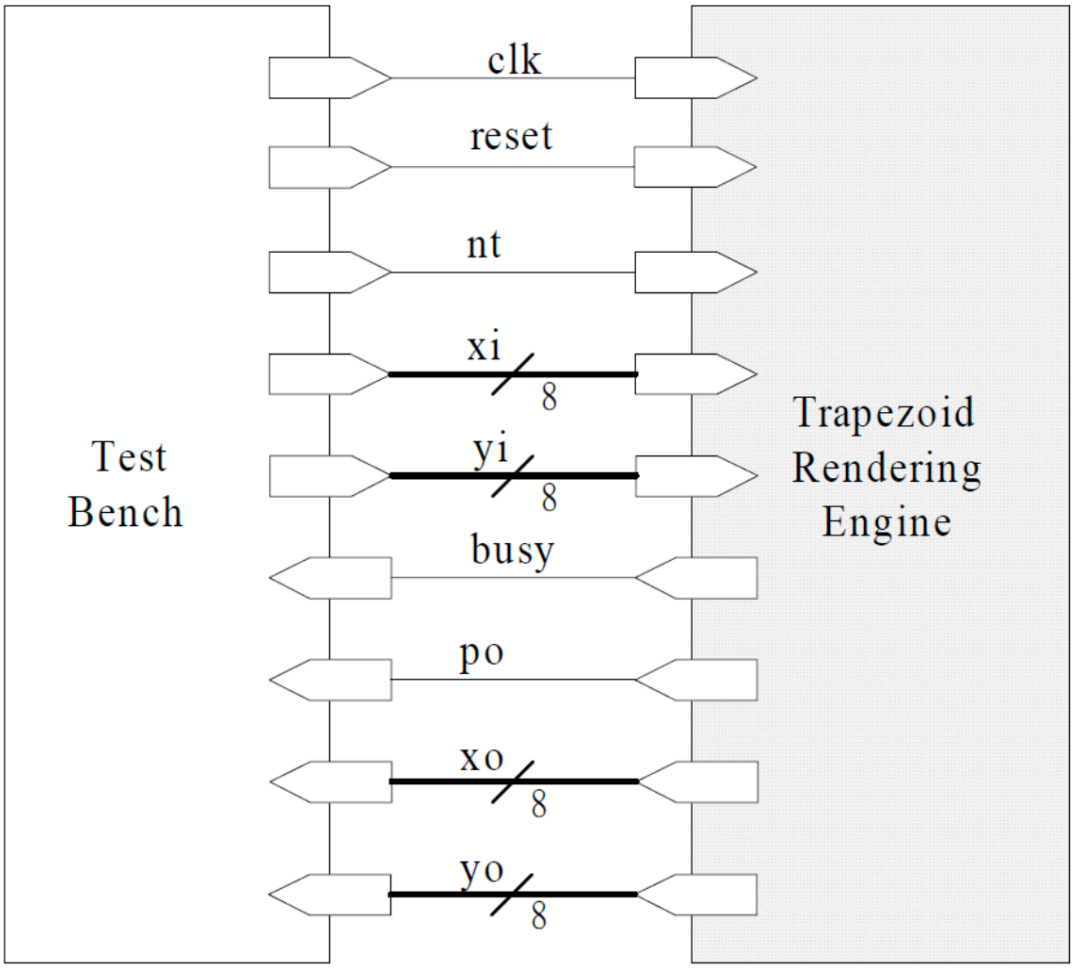
Trapezoid Rendering Engine

Abstract

A Trapezoid Rendering Engine (TRE) performs calculations on trapezoidal rendering. The TRE first receives the four coordinates (xul, yu), (xur, yu), (xdl, yd), and (xdr, yd) of the trapezoid. Then outputted the (xo, yo) coordinates that is covered by the trapezoid. In this design, we used the multiplier instead of divider to calculate the boundary coordinate. Using the multiplier decrease the design area and save the power.

Block Overview



I/O Interface

Signal name	Direction	Width	Description
reset	input	1	Active high asynchronous signal.
clk	input	1	Design in positive edge clock system.
nt	input	1	New trapezoidal input signal. When nt is pulled up to high, it indicates that four consecutive pairs of new trapezoidal coordinates will be input. nt is only pulled up when busy is low.
xi	input	8	Input of trapezoid x coordinate. In 2’s complement format.
yi	input	8	Input of trapezoid y coordinate. In 2’s complement format.
busy	output	1	Busy signal. When this signal is high, it indicates that the renderer circuit is currently performing computations and cannot accept new trapezoidal coordinate inputs. In other words, the nt signal will not receive any signals.
po	output	1	Output valid notification signal. When the po signal is high, it indicates that there is a sequence of coordinates being output on the output pins (xo and yo).
xo	output	8	Output of trapezoid x coordinate. In 2’s complement format.
yo	output	8	Output of trapezoid y coordinate. In 2’s complement format.

Functional Description

This design is a high-level asynchronous reset-triggered system. When the reset signal is pulled up to high, the system is reset. When the reset signal goes low, the circuit operates on the rising edge of the clk signal. The nt signal is triggered only when the busy signal is low. When the nt signal is pulled up to high, the testbench will start sequentially sending out the four pairs of coordinates for the trapezoids, following the order of (xul, yu), (xur, yu), (xdl, yd), (xdr, yd). The designed trapezoid rendering engine must pull up the busy signal to high before the last pair of coordinates (xdr, yd) is input to prevent the nt signal from rising again and sending out a new set of trapezoidal coordinates.

After a period of time, the circuit operation will raise the output valid signal, indicating that the subsequent signals appearing on the (xo, yo) output pins are valid and correct coordinates. At the same time as the po signal is raised, the testbench will start retrieving and verifying the output valid coordinates (xo, yo) on each rising edge of the clk signal until the po signal goes low.
Figure 1 shows an example of a trapezoid. The testbench will provide the four coordinates (xul, yu), (xur, yu), (xdl, yd), (xdr, yd) in sequence to find all the grid coordinates covered by this trapezoid. The (xo, yo) coordinates to be outputted, which are covered by this trapezoid, can be derived from the given formula.

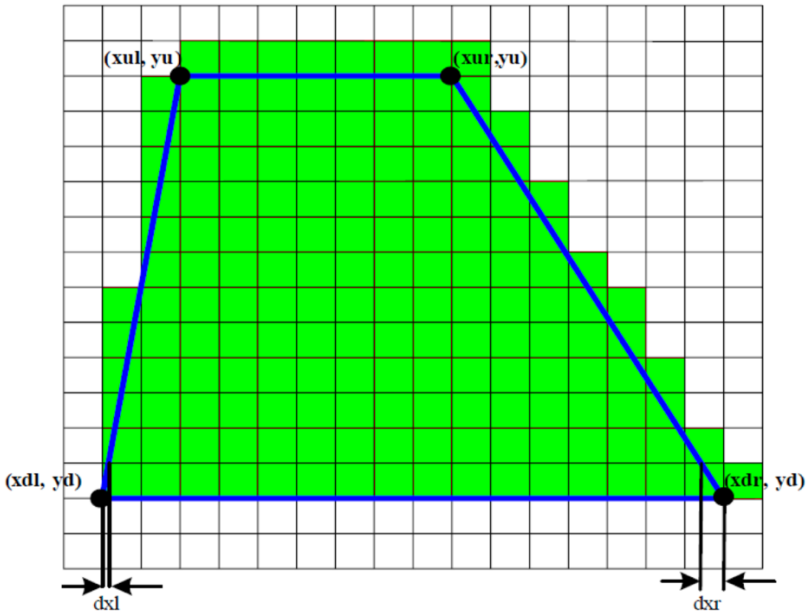


Figure 1: Example of a Trapezoid

$$dxl = (xul - xdl)/(yu - yd);$$
$$dxr = (xur - xdr)/(yu - yd);$$
where $yu - yd > 0;\ xdr - xdl > 0;\ xur - xul > 0;$
$$xl = xdl; xr = xdr;$$
$$for(yo = yd; yo \le yu; yo++)\{$$
$$\quad if(xl \in Integer) \qquad xo\_end = xr\ ;$$
$$\quad else\ if(xr \in Integer) \quad xo\_end = xr + 1;$$
$$\quad else \qquad xo\_end = \lceil xr \rceil\ ;$$
$$\quad for(xol = xl; xol \le xo\_end; xol++)\{$$
$$\qquad xo = \lfloor xol \rfloor;\quad yo = yo\ ;$$
$$\qquad output(xo, yo);$$
$$\qquad \}$$
$$\quad xl = xl + dxl; xr = xr + dxr;$$
$$\}$$

Algorithm in Design

To find the boundary coordinates, we start by deriving the equations of the left and right boundaries as ax + by = c. Since the y-values of the first row correspond to the inputted lower-left and lower-right points, we can directly output the first row. Based on the obtained equations, we repeat the following steps in a loop:

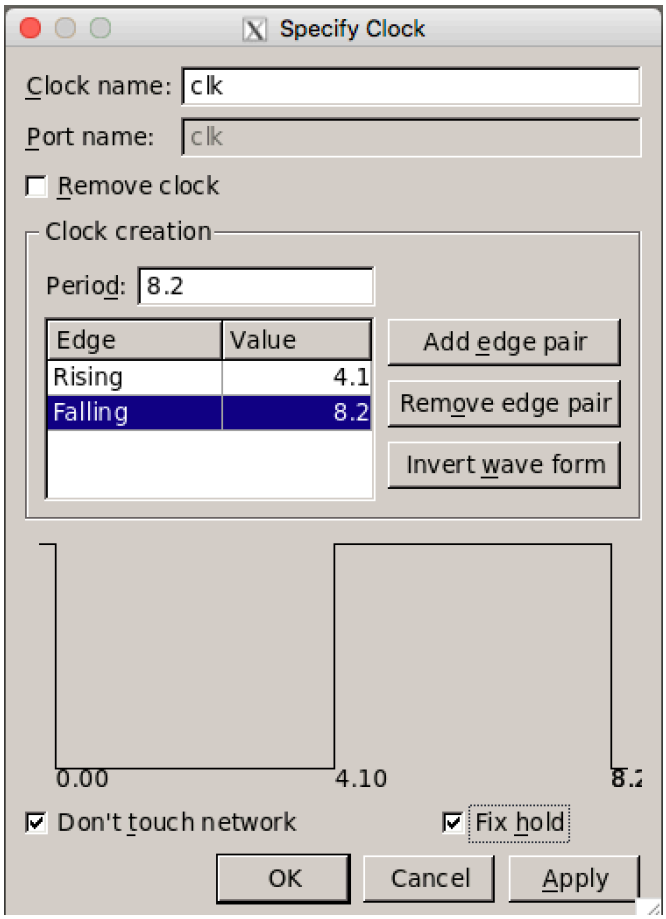
1. Increment the y-coordinate value by 1 for each iteration.
2. At the current y-value, find the coordinates of the left and right boundaries. Begin by searching for the left boundary. For each grid cell, substitute the coordinates of the lower-left and lower-right points into the equation. If the results have opposite signs, we have found the left boundary. Proceed to search for the right boundary using the same conditions as the left boundary search.
3. Once both boundaries are found, output the obtained coordinates to the output unit, which will output the corresponding row of the output. Additionally, store the coordinates of both boundaries in a register to serve as the starting point for the next iteration.

Verification

First, run each individual test fixture separately. If any errors occur, debug them using waveform simulation until each fixture is confirmed to be correct. Once each individual test fixture has been verified, run the test fixture containing all 17 sets and ensure that no errors are present.

Result

CLK: 8.2ns



Area: 24603.813570 um^2

```
Combinational area:              16188.103654
Buf/Inv area:                     1271.352580
Noncombinational area:            8415.708916
Macro/Black Box area:                0.000000
Net Interconnect area:      undefined  (No wire load specified)

Total cell area:                 24603.812570
Total area:                 undefined
```

Power:
Dynamic power:0.7302462(mW)
Static power:0.022291(mW)

```
Total Dynamic Power    = 730.2462 uW  (100%)

Cell Leakage Power     =  22.2910 uW
```

Power Group	Internal Power	Switching Power	Leakage Power	Total Power	(%)	Attrs
io_pad	0.0000	0.0000	0.0000	0.0000	(0.00%)	
memory	0.0000	0.0000	0.0000	0.0000	(0.00%)	
black_box	0.0000	0.0000	0.0000	0.0000	(0.00%)	
clock_network	0.0000	0.0000	0.0000	0.0000	(0.00%)	
register	0.6578	4.9234e-03	7.7477e+06	0.6705	(89.09%)	
sequential	0.0000	0.0000	0.0000	0.0000	(0.00%)	
combinational	3.6323e-02	3.1208e-02	1.4543e+07	8.2075e-02	(10.91%)	
Total	0.6941 mW	3.6131e-02 mW	2.2291e+07 pW	0.7525 mW		