<!DOCTYPE html>

<html>

<head>

<title>Logic Analyzer</title>

<link rel="stylesheet" href="Logic Style.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>

</head>

<body>

<div id='logic'>

<div id="IC\_name"></div>

<canvas id='display' width='500' height='100'>

</canvas>

</div>

<p id='test'>Hello</p>

<p id='demo'>Hi</p>

<p id='demo1'>Hi </p>

<div id="clock\_control" class="clearfix">

<div id='pictureBox'>

</div>

<button onclick='start\_logicClock()'>Start Logic Clock</button><br>

<button onclick='stop\_logicClock()'>Stop Logic Clock</button><br>

<button onclick='reset\_logicClock()'>Reset Logic Clock</button>

</div>

<div id="functionList">

</div>

<div id="create">

<h3>Create IC</h3>

Name:<input id="create\_name" type="text"

placeholder="IC name goes here"><br>

Inputs:<input id="create\_inputs" type="text"

placeholder="Separate inputs with spaces"><br>

Logical HIGH:<input id="logical\_H" type="number"

placeholder="Enter value as a percent">%<br>

Logical LOW:<input id="logical\_L" type="number"

placeholder="Enter value as a percent">%<br>

<button onclick="functionLoader()">Add Logic Function</button><br>

<input type="text" id="function\_selection" value="null">

<div id="moreLogic"></div>

Logic Start:<select id="create\_startLogic">

<option value="false">LOW</option>

<option value="true">HIGH</option>

</select><br>

Inversion:<select id="create\_inversion">

<option value="false">false</option>

<option value="true">true</option>

</select><br>

Gate Delay:<input id="create\_gateDelay" type="number" value="0"><br>

<button onclick="IC\_objectCreator()">Create Object</button>

</div>

<div id="edit">

<h3>Edit IC</h3>

Change:<select id="edit\_IC"></select>

<button onclick="editor\_nameLoaderIter()">Load Names</button>

<button onclick="IC\_editor(edit\_IC.value)">Edit</button>

<hr>

<div id="editor\_container">

</div>

</div>

</body>

<script>

//Functions below are used to start/stop logic clock

var clock;

/\*Prevents the clock from constantly running if start clock button is

pressed more then once

\*/

var start\_control = 0;

function start\_logicClock()

{

if (start\_control == 0)

{

load\_names();

clock = setInterval(logic\_clock,25);

start\_control = 1;

}

}

function stop\_logicClock()

{

clearInterval(clock);

start\_control = 0;

}

/\*If the canvas size needs to be increased, increase the

canvas size first, then (re)draw the elements on canvas

\*/

//Using variables to make referring to canvas easier

var display = document.getElementById("display");

var logic = display.getContext("2d");

//Setting up variables

//[logic\_clock]

var logic\_time = 0;

//[delay\_calculator]

var increment = 0;

//Height difference between logic levels

var logic\_vary = 10;

//Space between logic lines

var logic\_space = 5;

/\*Array used as a container to easily cycle through multiple logic lines.

MUST BE DECLARED AFTER LOGIC LINE VARIABLES.

\*/

var logic\_lines = [];

function load\_names()

{

//The 6 lines below are used to set up the names of the logic lines

var line\_names = "";

for (var x = 0; x < logic\_lines.length; x++)

{

line\_names = line\_names + logic\_lines[x].name + "<br>";

}

document.getElementById("IC\_name").innerHTML = line\_names;

}

//Function used to advance logic time

function logic\_clock()

{

container\_checker(logic\_time);

logic\_time = logic\_time+1;

}

/\*The [container\_checker] checks the container size and will call

[expand\_canvas] if the contents come too close to the edge

\*/

function container\_checker(logic\_timing)

{

if (display.width-logic\_timing <= 10)

{

display.width = display.width+100;

expand\_canvas(logic\_timing);

}

for (var x = 0; x < logic\_lines.length; x++)

{

if (logic\_lines[x].logic\_function[0])

{

eval(logic\_lines[x].logic\_function[1]);

}

draw\_logic(logic\_lines[x], logic\_timing);

}

}

//[expand\_canvas] expands the size of canvas

function expand\_canvas(logic\_timing)

{

for (var x = 0; x < logic\_lines.length; x++)

{

logic\_lines[x].gate\_delayLevel = 0;

logic\_lines[x].gate\_increment = 0;

logic\_lines[x].prev\_logic = 0;

logic\_lines[x].gate\_signalChange = 0;

logic\_lines[x].input\_iter = 0;

for (var y = 0; y < 3; y++)

{

logic\_lines[x].logic\_output[y] = 0;

}

}

for (var redraw\_time = 0; redraw\_time < logic\_timing; redraw\_time++)

{

for (var x = 0; x < logic\_lines.length; x++)

{

draw\_logic(logic\_lines[x], redraw\_time);

}

}

}

//[draw\_logic] draws the elements on the canvas

function draw\_logic(IC\_line, time)

{

/\*Placing the canvas line before the first if statement creates the

logic switching line.

The two if statements below checks for inversion. If true or not,

the statements stores values to logic\_outut's not and flip so

it can be used in calculations later.

\*/

if (IC\_line.inverted)

{

IC\_line.logic\_output[0] = -logic\_vary;

IC\_line.logic\_output[1] = -1;

}

else

{

IC\_line.logic\_output[0] = 0;

IC\_line.logic\_output[1] = 1;

}

//if statement below checks if the logic starts high

if (time == 0 && IC\_line.start)

{

IC\_line.prev\_logic = 1;

IC\_line.logic\_output[2] = logic\_vary;

IC\_line.gate\_increment = IC\_line.gateDelay;

IC\_line.gate\_delayLevel = logic\_vary;

}

//Moves the logic context to the previous logic state of drawn line

logic.moveTo(time-1,IC\_line.location+IC\_line.logic\_output[3]());

//if statement checks if the input matches the time, in which...

if (time == IC\_line.inputs[IC\_line.input\_iter])

{

//The increment if the object increases by 1

IC\_line.input\_iter++;

//If there is no gate delay, this if statement draws a vertical line

if (IC\_line.gateDelay == 0)

{

logic.lineTo(time,IC\_line.location+IC\_line.logic\_output[3]());

}

//Else, values are stored used to draw gate delay logic later

if (IC\_line.prev\_logic == 0)

{

IC\_line.gate\_signalChange = 1;

IC\_line.logic\_output[2] = delay\_calculator

(

IC\_line,

IC\_line.gateDelay,

IC\_line.gate\_signalChange

);

IC\_line.prev\_logic = 1;

}

else

{

IC\_line.gate\_signalChange = -1;

IC\_line.logic\_output[2] = delay\_calculator

(

IC\_line,

IC\_line.gateDelay,

IC\_line.gate\_signalChange

);

IC\_line.prev\_logic = 0;

}

}

/\*The else if statement checks if change reached its max or min value.

If so, change caps at its max/min value and the flag signal\_change

resets so the else if statement below stops affecting change.

Otherwise, change (in/de)creases based on the logic change.

\*/

else if (IC\_line.gate\_signalChange != 0)

{

IC\_line.logic\_output[2] = delay\_calculator

(

IC\_line,

IC\_line.gateDelay,

IC\_line.gate\_signalChange

);

if (IC\_line.gate\_signalChange == 1 &&

IC\_line.logic\_output[2] == logic\_vary)

{

IC\_line.gate\_signalChange = 0;

}

else if (IC\_line.gate\_signalChange == -1 &&

IC\_line.logic\_output[2] == 0)

{

IC\_line.gate\_signalChange = 0;

}

}

logic.lineTo(time,IC\_line.location+IC\_line.logic\_output[3]());

logic.stroke();

logic\_stateChecker(IC\_line);

}

function delay\_calculator(IC\_line, delay\_limit, direction)

{

//if statement used to draw vertical line

if (delay\_limit == 0)

{

if (direction == 1)

{

IC\_line.gate\_delayLevel = logic\_vary;

}

else

{

IC\_line.gate\_delayLevel = 0;

}

}

//else statement used to draw gate delay

else

{

IC\_line.gate\_increment = IC\_line.gate\_increment + direction;

IC\_line.gate\_delayLevel = logic\_vary \* (IC\_line.gate\_increment / delay\_limit);

}

return IC\_line.gate\_delayLevel;

}

//Function below determines if logic is HOGH or LOW

function logic\_stateChecker(IC\_line)

{

/\*logic\_levelPercentage is used to pre-emtively calculate how high

-- in percentage -- the logic is

\*/

var logic\_levelPercentage;

logic\_levelPercentage = IC\_line.gate\_delayLevel / logic\_vary \* 100;

//The if statement recalculates logic\_levelPercentage if the line is inverted

if (IC\_line.inverted)

{

logic\_levelPercentage = Math.abs(logic\_levelPercentage-100);

}

//First if satement checks if the line has no HIGH nor LOW threashold

if (IC\_line.logic\_change[0] == 0 &&

IC\_line.logic\_change[1] == 0)

{

if (IC\_line.gate\_delayLevel == logic\_vary)

{

IC\_line.logic\_state = "HIGH";

}

else

{

IC\_line.logic\_state = "LOW";

}

}

//Second if statement checks if the line has ONE HIGH or LOW threashold

else if (IC\_line.logic\_change[0] != 0 && IC\_line.logic\_change[1] == 0 ||

IC\_line.logic\_change[0] == 0 && IC\_line.logic\_change[1] != 0)

{

if (logic\_levelPercentage > IC\_line.logic\_change[0] ||

logic\_levelPercentage > IC\_line.logic\_change[1])

{

IC\_line.logic\_state = 'HIGH';

}

else

{

IC\_line.logic\_state = "LOW";

}

}

//Third if statement shecks if the line has a HIGH and LOW threashold

else if (IC\_line.logic\_change[0] != 0 &&

IC\_line.logic\_change[1] != 0)

{

if (logic\_levelPercentage > IC\_line.logic\_change[0])

{

IC\_line.logic\_state = 'HIGH';

}

else if (logic\_levelPercentage < IC\_line.logic\_change[1])

{

IC\_line.logic\_state = 'LOW';

}

}

}

//Resets the logic timer and canvas

function reset\_logicClock()

{

display.width = display.width;

logic\_time = 0; //Time is reset

//Array values are reset

for (var x = 0; x < logic\_lines.length; x++)

{

logic\_lines[x].gate\_delayLevel = 0;

logic\_lines[x].gate\_increment = 0;

logic\_lines[x].prev\_logic = 0;

logic\_lines[x].gate\_signalChange = 0;

logic\_lines[x].input\_iter = 0;

//Id statement used to reset logic level for gate functions

if (logic\_lines[x].logic\_function[0])

{

logic\_lines[x].logic\_state = "LOW";

}

for (var y = 0; y < 3; y++)

{

logic\_lines[x].logic\_output[y] = 0;

}

}

stop\_logicClock();

}

</script>

<script src="IC Object Creator.js"></script>

<script src="Load Logic Function.js"></script>

<script src="Function Logic Directory.js"></script>

<script src="AND Gate.js"></script>

</html>