OR Gate Backup:

//OR

var OR\_array = [];

//var store\_function = document.getElementById('function\_selection');

function ORgate(OR\_line, time)

{

var OR\_currentLine;

var OR\_logicOutput;

var OR\_findLL;

/\*For loop below searches OR\_array for the line called, then stores the name to

use later.

\*/

for (var OR\_lineFinder = 0; OR\_lineFinder < OR\_array.length; OR\_lineFinder++)

{

if (OR\_line == OR\_array[OR\_lineFinder].name)

{

OR\_currentLine = OR\_array[OR\_lineFinder];

break;

}

}

/\*For loop finds the logic\_line the name belongs to and stores the logic state

the line is currently in.

\*/

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

{

if (logic\_lines[LL\_index].name == OR\_line)

{

OR\_findLL = 'logic\_lines[' + LL\_index + ']';

OR\_currentLine.prevlogic = eval(OR\_findLL).logic\_state;

break;

}

}

/\*For loop checks if each input line is HIGH If not, the logic state of the gate is

set to LOW and the for lop is immediately exited.

\*/

for (var OR\_inputIter = 0; OR\_inputIter < OR\_currentLine.inputs.length; OR\_inputIter++)

{

OR\_logicOutput = 'LOW';

if (eval(OR\_currentLine.inputs[OR\_inputIter]).logic\_state == 'HIGH')

{

OR\_logicOutput = 'HIGH';

break;

}

}

/\*If statement checks if the logic state of the line changed. If so, the current time

is stored into the current line's inputs

\*/

if (OR\_logicOutput != OR\_currentLine.prevlogic)

{

//If statement checks if the gate has delay or has already stabilized

if (eval(OR\_findLL).gateDelay != 0 &&

eval(OR\_findLL).gate\_delayLevel != 0 &&

eval(OR\_findLL).gate\_delayLevel != logic\_vary)

{

/\*If statement checks if:

The signal is decreasing and the gate needs to change into a HIGH

The Signal is increasing and the gate needs to change into a LOW

If any of these parameters are true, the current time is added to the inputs

array of the gate

\*/

if((OR\_currentLine.prev\_gateInc > eval(OR\_findLL).gate\_delayLevel &&

OR\_logicOutput == 'HIGH') ||

(OR\_currentLine.prev\_gateInc < eval(OR\_findLL).gate\_delayLevel &&

OR\_logicOutput == 'LOW'))

{

eval(OR\_findLL).inputs.push(time);

order\_existingArray(LL\_index);

}

}

//Adds an input into the gate's inputs array if if statement isn't true

else

{

document.getElementById('test').innerHTML = 'Gate has an input!';

eval(OR\_findLL).inputs.push(time);

order\_existingArray(LL\_index);

}

OR\_currentLine.prev\_gateInc = eval(OR\_findLL).gate\_increment;

}

}

function ORgate\_setup()

{

document.getElementById('demo').innerHTML = "Can you see me?";

var table\_inputs = "<table><tr><th>OR</th></tr>";

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

{

table\_inputs += "<tr><td class='inputTable'>" +

logic\_lines[table\_count].name +

"</tr></td>";

}

table\_inputs += "</table>" +

"<button id='createObject'>Finish Gate</button>";

document.getElementById('moreLogic').innerHTML = table\_inputs;

$(document).ready(function()

{

$(".inputTable").click(function()

{

$(this).toggleClass("selected\_input");

});

$("#createObject").click(function()

{

var IC\_nameLF = document.getElementById('create\_name').value;

var OR\_classCounter = document.getElementsByClassName('selected\_input');

var OR\_inputObject = {name: IC\_nameLF, inputs:[], prevlogic: '', prev\_gateInc: 0};

for (var iter\_classCounter = 0; iter\_classCounter < OR\_classCounter.length; iter\_classCounter++)

{

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

{

if (logic\_lines[LL\_index].name == OR\_classCounter[iter\_classCounter].innerHTML)

{

OR\_inputObject.inputs.push('logic\_lines['+ LL\_index +']');

break;

}

}

}

OR\_array.push(OR\_inputObject);

store\_function.value = 'ORgate("' + IC\_nameLF + '", logic\_timing)';

document.getElementById('moreLogic').innerHTML = "";

});

});

}