Virtual Labs Project

Digital Logic and Processors

Design Documentation

Guide: Dr. P. J. Narayanan

Students:

Phani Krishna KSSS

Rajesh Chowdary

Contents

1. Introduction

Availability of resources limits learning by doing experiments in many educational institutions. Web-based and video-based courses address the access to good quality teachers. Virtual labs are intended to address the need for access to good lab facilities. They also provide a new methodology to convey and learn concepts using the power of visualization of ideas and computations. Virtual labs rely on an active engagement of the learner in the knowledge acquisition process. This project is aimed at promoting this new methodology and providing a glimpse into the exciting world of experimentation to a student/teacher unlimited by their physical location. The aim of this project is to build a virtual lab for Digital Logic and Design (DLD). It provides a web based interface for learning to build circuits of a user's choice. It provides a space where a user can play with simple gates and circuits by placing them in the circuit space, connecting them, specifying inputs and simulating them to see how they function.

This document describes the design structure of the DLD Virtual Labs Project. This document is meant for developers who intend to continue developing the project. It explains the data structures used, logic implemented and the reasons behind our choice of data structures or programming paradigms so that the next developers would find it easy to continue working on it.

2. System Overview

The entire project was developed in JAVA 6 using Netbeans IDE. The front end interface was developed using Java Swings, while the backend was implemented in Java. The project was developed in an Object Oriented Manner using separate classes for each item in the project. The project is in two parts – Front End code and Back end code.The back end part of the project has 4 main classes at the backend viz. Input, Output, Element and Circuit. Every other basic gate or an element in a circuit is a child class of the parent class Element. Class Element has all its member variables as protected so that they can be accessed or modified only by its sub classes and not any other external function. It also provides a set of set and get functions that can be used to read and write into those member variables. The sub classes have constructors and four other functions that override the definition of class Element. Infact, it does not implement four functions i.e. updateMatrix, updateLocation, draw and process functions and raises exception if a child class does not provide any implementation. The front end is a Java Applet that can be embedded into a browser so that the project can be used through a browser. It uses various widgets and features provided by Java Swings and integrate various parts of the backend to the user. It passes on user information to the backend for processing.

3. System Architecture

3.1. Class Input

An input is a specific node that holds a particular value that it passes on as input to its parent element. An input node gets its data value from either an output of a previously processed element or is fed directly from the user. It's class structure and function declarations are mentioned below. Every input in the circuit has its own unique "inputID" and is located at "inputIndex"-th input among its "ancestor", parent element's several inputs. It holds a value by name "dataValue". It is located at a specific "location", Point(x,y) in the 2D space of the circuit. Location coordinates are used for drawing the input using Java 2D graphics. Any input can be probed by setting the "probed" flag. Inputs can also have time varying input values which are denoted by label, "timePulseLabel". All the variables are private variables that are accessed or modified using set/get functions.

Table 1 : Class Input - Member Variables

Variable Name	Explanation
inputID	Unique integral id for each input node present in the circuit
inputIndex	The index of the input among several others for its parent element
ancestor	Parent Element to which this input belongs – of class Element
dataValue	The integral value (0/1) that this input node passes to its parent
sourceOutput	Output node of class Output that from which it gets its value
location	The downscaled 2D location in circuit space where it is situated
probed	Boolean value if set implies that the input value is being probed
timePulseLabel	The label that denotes a user defined time varying function that defines input value to be fed at that input location

Table 2: Class Input - Member Functions

Function Name	Explanation
Input()	Null Constructor used while loading a circuit from file
Input(id, index, parent)	Null Constructor used while loading a circuit from file
Input(id, index, parent, location)	Regular Constructor used when a new element is added to a circuit. This is called from Element constructor
setAncestor(parent)	Sets the parent element of this input as parent
setDataValue(value)	Sets dataValue to value
setSourceOutput(out)	Sets sourceOutput(out) i.e. this input gets its value from the output node out
setLocation(loc)	Sets the location of the input node in 2D space as loc
setInputIndex(ind)	Sets this input as the ind-th input to its parent element

setProbed(state)	Sets the probed state if the input is being probed
setTimePulseLabel(label)	Sets the input label to “label” i.e. the input gets its data value from the time varying function specified by label
getInputID()	Returns the integral value of the input’s id
getAncestor()	Returns the parent element that this input belongs to
getDataValue()	Returns the integral value of the data that this input holds or passes on to its parent element
getLocation()	Returns the location of the input in circuit space of Java Point data type
getInputIndex()	Returns the index of the input among its several others to its parent element
getTimePulseLabel()	Returns the label of the time varying function attached to this input
draw(g, p)	Draws the input at downscaled 2D location p using Java 2D graphics g

An input is drawn in 2D space using Java 2D graphics. A red rectangle is drawn at its location and a black line is drawn joining the input node to its corresponding parent element. A yellow label appears near each input node denoting the value it holds. It shows “X” until the input is attached some value among 0, 1 or user defined time pulse label. If the input is connected to some other output, this yellow label disappears and a black connection is drawn connecting the two. While being probed after simulation, a green label appears showing the data Value it passes on to its parent element.

3.2. Class Output

An output is a specific node that holds a particular value that it gets after its parent element has been procesesd. An output node passes its data value to either an input of another yet to be processed element or is given directly to the user. It’s class structure and function declarations are mentioned below. Every output in the circuit has its own unique “outputID” and is located at “outputIndex”-th output among its “ancestor”, parent element’s several outputs. It holds a value by name “dataValue”. It is located at a specific “location”, Point(x,y) in the 2D space of the circuit. Location coordinates are used for drawing the output using Java 2D graphics. Any output can be probed by setting the “probed” flag. Outputs can also have time varying output values which are denoted by label, “timePulseLabel” based on the input values its parent element inputs get. All the variables are private variables that are accessed or modified using set/get functions.

Table 3: Class Output - Member Variables

Variable Name	Explanation
outputID	Unique integral id for each output node present in the circuit
outputIndex	The index of the output among several others for its parent element
ancestor	Parent Element to which this output belongs – of class Element
dataValue	The integral value (0/1) that this output node passes to its parent
destinationInput	Input node of class Input to which it is connected & passes its value
location	The downscaled 2D location in circuit space where it is situated
probed	Boolean value if set implies that the output value is being probed

Table 4: Class Output - Member Functions

Function Name	Explanation
Output()	Null Constructor used while loading a circuit from file
Output(id, index, parent)	Null Constructor used while loading a circuit from file
Output(id, index, parent, location)	Regular Constructor used when a new element is added to a circuit. This is called from Element constructor
setAncestor(parent)	Sets the parent element of this output as parent
setDataValue(value)	Sets dataValue to value
addDestinationInput(inp)	Adds inp to the list of inputs to which it is connected or passes its value it gets after its parent's processed
addDestInpID(indID)	Adds inpID to the list of inputIDs to which it is connected
setLocation(loc)	Sets the location of the output node in 2D space as loc
setOutputIndex(ind)	Sets this output as the ind-thoutput to its parent element
setProbed(state)	Sets the probed state if the output is being probed
addDestInpId(id)	Adds this input id to its list of input ids to which it is connected or to which it passes its input value
getOutputID()	Returns the integral value of the output's id
getAncestor()	Returns the parent element that this output belongs to
getDataValue()	Returns the integral value of the data that this output holds or passes on to its parent element
getLocation()	Returns the location of the output in circuit space
getOutputIndex()	Returns the index of the output among its several others to its parent element
getDestinationInputsList()	Returns a list of all the input nodes to which this output is connected and feeds its data value
getDestInpIdsList()	Returns a list of the ids of input nodes to which this output is connected

draw(g, p)	Draws the output at downscaled 2D location p using Java 2D graphics g
delDestInpNode(inpNode)	Delete inpNode from the list of nodes to which this output is connected

An output is drawn in 2D space using Java 2D graphics. A red rectangle is drawn at its location and a black line is drawn joining the output node to its corresponding parent element. A yellow label appears near each output node denoting the value it holds. It shows "X" until the output is attached some value among 0, 1 or system defined time pulse label. If the output is connected to some other output, this yellow label disappears and a black connection is drawn connecting the two. While being probed after simulation, a green label appears showing the data Value it passes on to its parent element.

3.3. Class Element

An Element is a basic processing item in a circuit. It is a generic representation of any component that can process a given set of input values and pass on the outputs to a next set of elements for further processing or give the output directly to the user. An element has a certain number of inputs, certain number of outputs and a simulation logic that it uses to process its input values. In this project, Element is used as a parent class and every other processing component is inherited publicly from this function. All the member variables of Element class are protected so that they can be accessed in its child classes. All its functions are public and are declared in an ElementInterface interface so that every function is implemented by either class Element or one of its sub-classes.

For ease of design and implementation, we have 8 child classes that are inherited from class Element. They include a set of 7 basic gates i.e. OR, NOR, AND, NAND, XOR, XNOR and a GenericElement class that represents any processing component in general. The last data structure is used to import previously existing circuits as small processing elements in bigger circuits. Each of the 7 gates are drawn according to widely used standards using 2D graphics while the GenericElement is drawn as a black box with certain number of inputs and outputs based on its inbuilt Circuit that it represents. Except for processing and drawing logics, all these data structures have similar representations and properties represented in general by class Element. Among the functions declared in ElementInterface interface, all the set and get functions are implemented by class Element while the processInputs, updateMatrix, updateLocation and draw functions are overridden by every child class with its own implementation. Each child class has its own constructors that override class Element constructors based on its own unique properties for drawing, processing etc.

Table 5: Class Element - Member Variables

Variable Name	Explanation
elementID	Unique integral value given to each element present in the circuit
elementName	Name of the element [elementType<elementID>]
elementType	Type of the elment [And_Gate/Or_Gate/Generic_Element]
numInputs	Number of input nodes for this particular element
numOutputs	Number of output nodes for this particular element
maxIO	Value equaling (1 + max(numInputs, numOutputs)) This is used for drawing in case of Generic_Element type
Width	Width of the element (in pixels) – used for drawing
Height	Height of the element (in pixels) – used for drawing
inputList	Vector storing a list of all the input nodes for this element
outputList	Vector storing a list of all the output nodes for this element
Location	Downscaled 2D location of the element in circuit space
inbuiltCkt	The inherent circuit that was imported as an element in the case of Generic_Element – needed to process the element

Table 6: Class Element – Member Functions

Function Name	Explanation
Element()	Null Constructor used while loading a previously saved circuit
Element(id, type, inpID, numInp, outID, numOut, loc)	Constructor that creates an element with elementID(id) of elementType(type) with numInp inputs starting with inpID incrementally and similarly with outputs at loc in 2D space. Called when new element is created
setNumInputs(count)	Sets the number of inputs of the element to count
setNumOutputs(count)	Sets the number of outputs of the element to count
setElementType(typeName)	Sets the type of the element to the string typeName
setElementName(eleName)	Sets the name of the element to the string eleName
setInputAt(inp, ind)	Sets the ind-th input as input node inp
setOutputAt(out, ind)	Sets the ind-th output as output node out
addInput(inp)	Adds inp to the list of inputs to this element
addOutput(out)	Adds out to the list of outputs of this element
setLocation(loc)	Sets the location of this element to loc in 2D space
setInbuiltCkt(ckt)	Sets the inbuilt circuit to ckt in case of Generic Element
getElementID()	Returns an integral value of the id of this element
getElementName()	Returns a string value representing the name of element

getNumInputs()	Returns an integral value of the number of inputs
getNumOuptuts()	Returns an integral value of the number of outputs
getMaxIO()	Returns the integral value of the variable maxIO
getLocation()	Returns a Java Point location of this element in 2D space
getInputList()	Returns a vector of all input nodes to this element
getInputAt(ind)	Returns an Input data structure of the ind-th input to the element
getOutputList()	Returns a vector of all the output nodes to this element
getOutputAt(ind)	Returns an Output data structure of the ind-th output to the element
getInbuiltCircuit()	Returns a Circuit data type of the circuit that this element represents in case of Generic Element type
processInputs() *	Processes all the inputs according to its logic and populates the data values in output nodes
updateLocation(p) *	Updates the location of this element, its inputs and outputs to Java Point p from its previous location
updateMatrix(p, matrixType, matrixID, prev) *	Updates the matrixType and matrixID matrices of the circuit at location p and prev. It removes its values at location prev and assigns them at p.
draw(g, p) *	Draws the element, its inputs and outputs at location p in the 2D circuit space using Java 2D graphics g.

***** The last 4 functions (starred) are to be compulsorily overridden by all the child classes so that each child class has its own unique manner of processing function and drawing itself. Since the sizes and dimensions also vary from one element to the other, the update functions should also be overridden. If not, they throw UnsupportedOperationException from the Element class.

3.4. Class Circuit

A circuit is a collection of interconnected processing elements each of which hasa certain number of inputs and outputs. It has an integrated logic that cascadesacross elements connected to each other. Input Values can be specified at circuit level inputsand the circuit simulated to see output values at circuit level outputs. Each of the constituent elements is processed when all its input values are availableand the output values are passed on via connections to next level inputs. The connections between elements are not stored explicitly, but are drawn or understood with the help of sourceOutput variables in input nodes and destinationInputList variables in output nodes. Data values are passed on using these ids and connections for processing the circuit. A circuit has meshType and meshID variables that store ids and type of the element that exists at each location in 2D space and are used for drawing.

Table 7: Class Circuit - Member Variables

Variable Name	Explanation
circuitID	Unique integral value given to the circuit
circuitName	Name of the Circuit [circuit<circuitID>] – can also be set by user
numInputs	Number of inputs for the entire circuit – if treated as a black box
numOutputs	Number of outputs for the entire circuit – if treated as a black box
circuitInputsList	Vector of input nodes that act as inputs for the circuit as a whole
circuitOutputsList	Vector of output nodes that act as outputs for the circuit as a whole
inputElementSet	HashSet of elements each of which has atleast one circuit level input
outputElementSet	HashSet of elements each of which has atleast one circuit level output
meshType	An integral matrix array that stores 0 / the type of element(3) / input(1) / output(2) located at each location in the circuit space
meshID	An integral matrix array that stores -1 / the ID of the element / input / output present at each location in the circuit space
allElementsList	HashMap of all the elements present in the circuit with their ids as keys and their data structures as values
allInputsList	HashMap of all the inputs present in the circuit with their ids as keys and their data structures as values
allOuptutsList	HashMap of all the outputs present in the circuit with their ids as keys and their data structures as values

Table 8: Class Circuit – Member Functions

Function Name	Explanation
Circuit()	Null Constructor used while loading a circuit from a previously saved file
Circuit(id, name)	Constructor called when a new circuit instance is created while launching the applet
Circuit(orgCkt)	Copy Constructor used while importing a saved circuit as a generic element – copied to its inbuiltCkt variable
setCircuitName(cktName)	Sets the name of the circuit to cktName
setNumInputs(count)	Sets the number of inputs to the circuit to count
setNumOutptuts(count)	Sets the number of outputs to the circuit to count
addElement(newElement)	Adds newElement to the list of constituent elements in the circuit. Subsequently, it also adds its inputs and outputs to the corresponding list variables of the circuit
setMeshValue(x, y, type, id)	Sets the mesh matrix values at location (x,y) to an element of type “type” and id “id”
getCirciutID()	Returns an integral value of the id of the circuit

getCircuitName()	Returns a string value of the name of the circuit
getNumInputs()	Returns an integral value of the number of inputs to the circuit (at circuit level – if treated as a black box)
getNumOutputs()	Returns an integral value of the number of outputs to the circuit (at circuit level – if treated as a black box)
getCircuitInputsList()	Returns a list of all the input nodes present in the circuit
getCircuitOutputsList()	Returns a list of all the output nodes present in the circuit
getCircuitInputAt(ind)	Returns the ind-th input of the circuit
getCircuitOutputAt(ind)	Returns the ind-th output of the circuit
getInputElementSet()	Returns a hashset of elements that have atleast one circuit level input
getOutputElementSet()	Returns a hashset of elements that have atleast one circuit level output
getAllElementsList()	Returns a hashmap of all the elements present in the circuit with keys as their ids
identifyCircuitInputsOutputs()	Identifies the circuit level inputs and outputs i.e. those input nodes to which user has to feed input values and those output nodes that give the final values
processCircuit(genericFlag)	Processes the circuit cascading outputs to connected inputs. The genericFlag if set denotes that the inbuilt circuit of a generic element is being processed
deleteElement(delID)	Deletes the element with id delID from the circuit. It also removes all its connections to other elements
drawZigZag(g2d, src, dest)	Draws a zig-zag connection between src and dest points
draw(g2d)	Draws all the elements in the circuit using 2d graphics
loadCircuit(bufReader)	Loads a previously saved circuit file opened by bufReader into this circuit instance and returns a success statement
saveCircuit(filename, appendFlag)	Saves the current circuit into a file with the name “filename” in append mode if appendFlag is set. Append is done if the circuit has generic element that has to save its inbuilt circuit it represents inline.

A new circuit instance is created when the applet is launched by calling the second constructor mentioned above with a default circuit ID and circuit name. A circuit can be saved to any location and loaded from any location. A previously saved circuit file can be imported as a generic element in a new circuit by clicking on the IMPORT button and specifying the path. A circuit is drawn by iterating through all its elements, inputs and outputs and drawing them independently. Connections are drawn once the inputs and outputs are drawn. Labels showing inputs, outputs or data Values(if probed) are drawn after the inputs and outputs.

Simulation of the circuit is implemented by giving circuit level input elements their input values, processing them and feeding forward their output values. Then these values are fed to other elements which are processed later. Once all the elements in the circuit are processed, the circuit is processed and the output values are given to the user. Different locations in the circuit can be probed to see how the values are changing at those locations. They can also be seen on separate labels that show the values they hold.

Connections between inputs and inputs and outputs and outputs are not allowed. The mesh values are used to identify which element/input/output is present at the location where the user clicked. The id and type are retrieved and checked for valid connections or valid locations for placing elements. Using the ids of the starting and ending locations while making connections (appears in green after clicking CONNECT button), the corresponding parameters are updated in the data structures and used while drawing connections. Disconnecting an existing connection between two nodes is also similar. One should make a connection between the two (appears in red after clicking DISCONNECT button). This will modify the parameters in their data structures and the connection is removed in the next draw iteration (repaint()).

To probe a specific input or output, PROBE button can be clicked and then a specific location selected. This sets the probed flag for that particular component (input/ouput) and the data value label appears at that location showing the data value as it changes with each simulation. For inputs or outputs without any connections, labels appear automatically showing either X (no input specified/no output without simulation) or 0/1 (specified input / output after simulation).

4. Glossary

- Element

- Input
- Output
- Circuit
- Circuit Space
- Time Pulse Label
- Generic Element
- Java 2D Graphics
- Connection
- Load Circuit
- Save Circuit
- Import Circuit
- UnsupportedOperationException
- Location
- Java Point

5. References

- http://download.oracle.com/javase/tutorial/
- http://www.java2s.com/
- http://www.roseindia.net/