How the Simulator Works

William_Kaupinis@eightolives.com
Apr 20, 2010

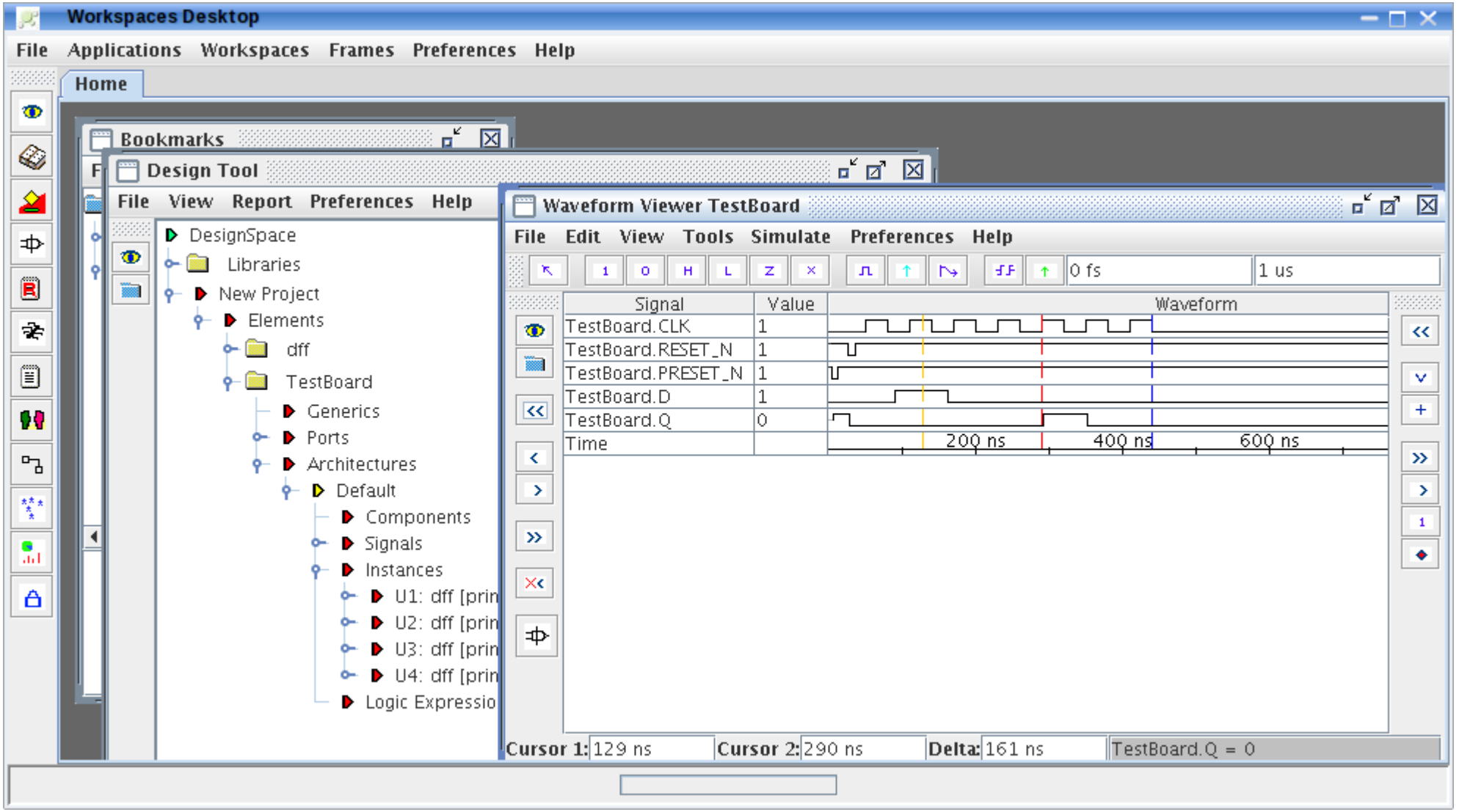
Abstract

This presentation describes the internal workings of the simulator used in the eightolives Workspaces Desktop.

Background

- The Workspaces Desktop toolkit includes DesignTool which is used to input and edit a design description and the WaveformViewer which is used to display and edit related waveform data.
  - Designs are internally captured using the Java based com.eightolives.Hardware package API
- A Simulator function is available to interact between the design and the WaveformViewer display

DesignTool and WaveformViewer



Hardware API Basics

- DesignTool captures a design using the Hardware API
  - An **Element** represents a logic expression or component either top level or internal and its interfaces
  - An **Architecture** represents the internal structure of components, interconnect and functionality
  - A **Port** is a connection object that interfaces an external Signal to an internal Signal
  - A **Signal** captures the set of connected Ports and is an object that has a Value

Signals Come in Different Flavors

- StdLogic is the basic digital signal
- StdLogicVector represents a multi-bit bus of StdLogic signals
- StdInteger represents an Integer value
- StdBoolean represents boolean true or false
- StdReal represents a Java floating point Double
- StdString represents a Java String object

A Signal's State is its Value

- A Signal's value can be expressed in two ways
  - A character value = 0, 1, L, H, Z, X, -, W, U
  - A Java Object Value = String, Integer, Double
- The basic simulation method attempts to assert a Value at some future Time using the Signal's setTo method
  - public void setTo(char c, Time t);
  - public void setTo(Value v, Time t);
  - These methods enter the data on the Simulator's FutureQueue
- Each Signal also maintains both its current Value and its next Value

Startup

- On startup, the Simulator calls the Top Design Element's uninitialize(SimQueue) method
- This method traverses all design Elements and is used to set each Signal value to its uninitialized State (typically 'U' for StdLogic) and informs the Signal as to the Simulator being used
- Elements forward the uninitialize command to its selected Architecture

Simulator Uses Several Queues

- The Simulator's FutureQueue contains proposed transitions of Signals scheduled at some future time
- CurrentQueue contains Signals that are changing at this point in Time
  - Adds the Signal's parent Element to the PrimitiveQueue
  - Adds the Signal to the NotifyQueue
- A NotifyQueue contains changed Signals that must propagate through their connected Ports
- A PrimitiveQueue contains Elements that must be executed to evaluate a new state for signals that have changed

Where do FutureQueue Events Come From?

- FutureQueue events can originate from
  - Waveforms that are interactively drawn on the WaveformViewer
  - Value Change Dump (vcd) files loaded at the start of simulation
  - Testbenches that provide stimulus and checking functionality which are part of the Top Element being simulated
  - Javascript scripts or interactions via a CommandProcessor window
  - The design's primitive models or models that override the execute() method

FutureQueue Processing

- The Current Time is set by the next pending event
- The FutureQueue events for the current time are executed by asserting each entry's Signal with the associated value
  - public void setTo(char c);
  - public void setTo(Value v);
- These Signal setTo functions immediately set the Signal's next value only and put the signal on the CurrentQueue
- The event is then removed from the FutureQueue
- The signal is added to the NotifyQueue

Updating CurrentValue

- The Simulation then asserts each Signal's endSimTick() method which
  - Sets the Signal's CurrentValue <= NextValue
  - Passes the information to the WaveformViewer for display
- The Simulator then processes CurrentQueue data

CurrentQueue Processing

- Until the CurrentQueue is empty
  - Each CurrentQueue signal has its parent Element added to the PrimitiveQueue
  - Each signal is commanded to notifyListeners()
    - A Signal's listeners are the Ports it is connected to
  - The signal is added to the NotifyQueue
  - The signal is removed from the CurrentQueue
- Then endSimTick() is issued to the NotifyQueue
- Then all Elements in the PrimitiveQueue have execute() asserted and endSimTick() is issued to the NotifyQueue

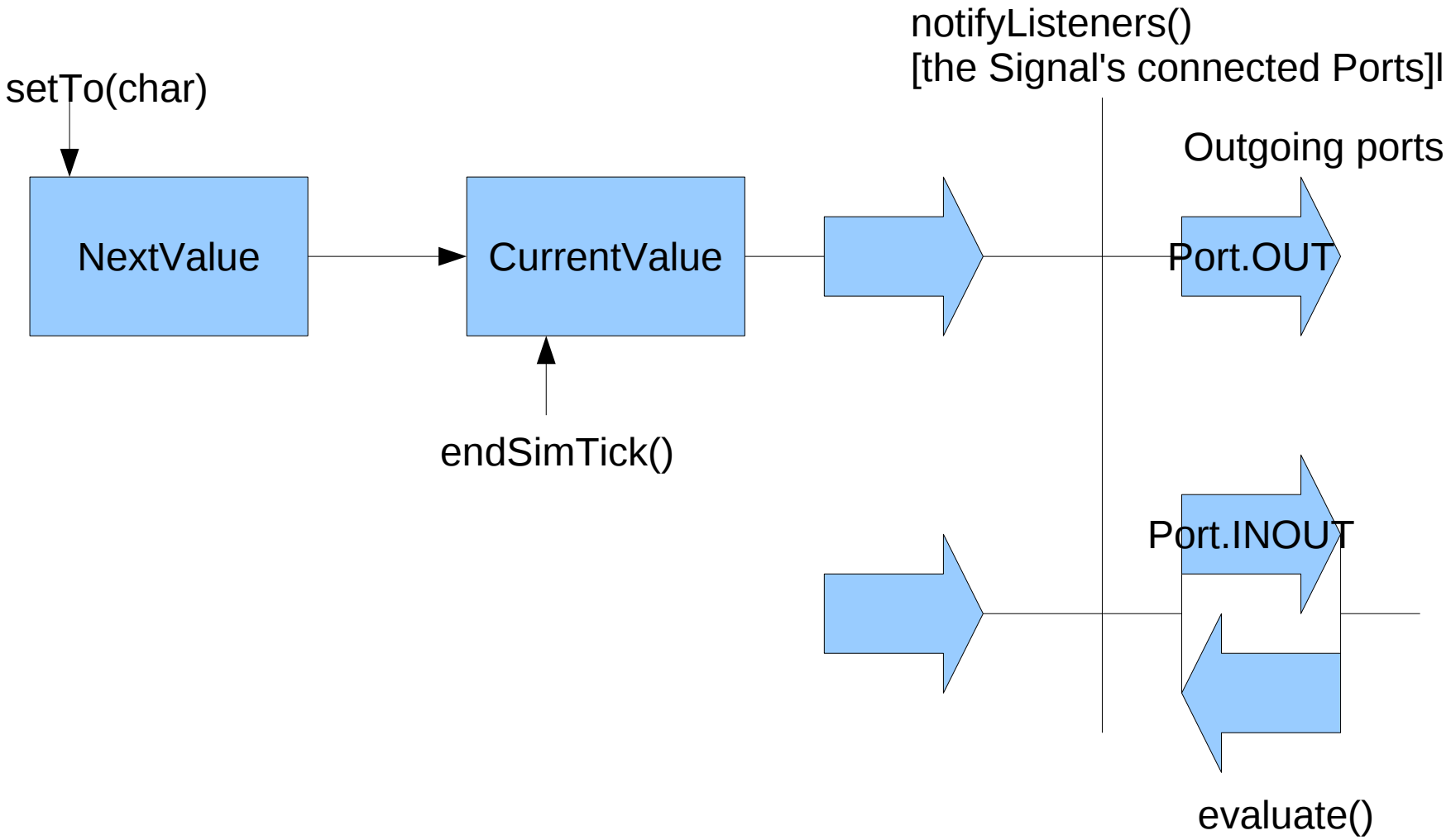
notifyListeners()

- If the Signal's Value has changed, all of the Ports connected to the Signal will have their evaluate() methods called
  - This propagates the signal change through the design hierarchy
- The parent Element of the Port will be added to the PrimitiveQueue

About Ports

- Ports as an interface connection have an internal Signal connection and an external Signal connection
- Ports have a mode defined:
  - IN – signal flow is from external to internal
  - OUT – flow is from internal to external
  - INOUT – flow can be both ways
- Ports of modes IN and INOUT can have an InternalDrivingValue which may differ from the internal Signal value
- Ports of mode OUT and INOUT can have an ExternalDrivingValue which may differ from the external Signal value

Value Flow



evaluate()

- The Port's evaluate(signal) method manages the propagation of signal value through the Port
- For StdLogic signals, a resolution function negotiates multiple active drivers on the output
- It issues setTo(char) command and endSimTick() command on the propagated signal
- Outgoing StdLogic signals are resolved

execute()

- The Element's execute() method is forwarded to the active Architecture where it is forwarded to all components and LogicExpressions
- Its intent is to implement functionality by processing the input Signal Values and generating outputs in the future using the output signal's setTo(Value, Time) methods
- An Element with ATTRIBUTE PRIMITIVE = true indicates that the item is simulatable

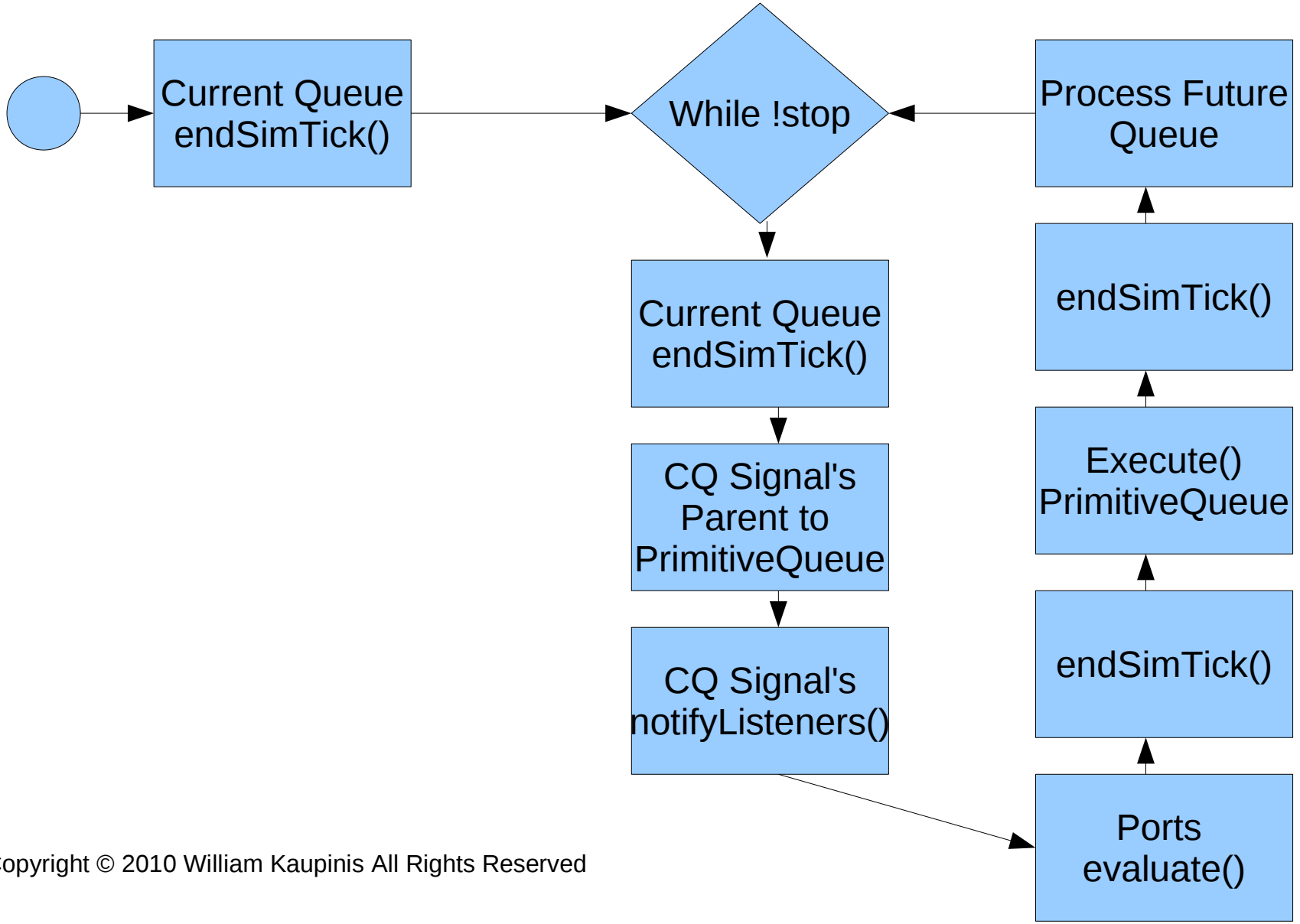
Simulation stops

- When a Maximum Simulation time is reached or
- When a specified duration has been reached or
- When the Stop button is pressed or
- When the FutureQueue is empty

Key Hardware Methods Summary

- Element
  - uninitialize(SimQueue); - sets signals to uninitialized values i.e. 'U'
  - execute(); - calls Architecture's execute, overridden in primitive models
- Architecture
  - uninitialize(SimQueue); - sets signals to uninitialized values i.e. 'U'
  - execute(); - overridden in primitive models, issues setTo(char, Time) commands
- Signal
  - setTo(char, Time) - puts item in FutureQueue
  - setTo(Value, Time) - puts item in FutureQueue
  - setTo(char) - immediately sets nextValue, puts Signal on CurrentQueue
  - setTo(Value) - immediately sets nextValue, puts Signal on CurrentQueue
  - endSimTick(); - currentValue <= nextValue, draws the change
  - notifyListeners() - call evaluate() on all connected Ports, Port's parent added to PrimitiveQueue
  - resolve(char, Signal) - resolution function for two StdLogic values
- Port
  - evaluate() - issues setTo(char), endSimTick() on propagated signals

Simulator Flow



For more information

- Check the tutorials at: http://www.eightolives.com/tutorials.htm
  - Workspaces Desktop Tool Overview
  - Modeling Hardware in Java
  - Simulating Javascript Models
- Read the Workspaces Desktop Users Manual
- Consult the com.eightolives.Hardware API