User manual SyncSimulator

# Operation

DUT has to be written on beforehand.

Run-button starts the simulation

The trackbar regulates the speed.

At each breakpoint, a thread waits a random time, so the thread-interleaving will vary.

checkbox "block at \_blk()" stops the simulation at each call of \_blk(), wait(), signal(), notify() or notify\_all()

A breakpoint for a line can be set for individual threads, but also for all threads at once (the checkbox on right).

The + button on the top activates that thread; it runs until it hits the next breakpoint. The o button indicates.

Red line indicator: thread has stopped at that line.

Yellow line indicator: last known location of the thread (so it is either running lines without breakpoints, or it is stuck in a wait())

# Device Under Test ("DUT")

Your software to be tested

Requirements:

* setup() to subscribe the functions that have to run as threads
* define variables whose contents need to be shown during execution

All lines with a wait() or signal() get automatically a breakpoint. You can add your breakpoints on other locations by calling \_blk()

# Classes

(very brief description, typically to be used as name-lookup (+ parameters))

* MySemaphore
  + \_\_init\_\_(init\_val=0,name="")
  + wait()
  + signal(n=1)
* MyMutex
  + \_\_init\_\_(name="")
  + wait()
  + signal()
* MyConditionVariable
  + \_\_init\_\_(mutex,name="")
  + wait()
  + notify()
  + notify\_all()
* MyBarrier
  + \_\_init\_\_(name="")
  + wait()
* MyLightswitch
  + \_\_init\_\_(sem,name="")
  + wait()
  + signal()
* MyInt, MyBool, MyString
  + always use member .v in the statements
* MyFifo (thread-unsafe FIFO queue)
  + \_\_init\_\_(maxsize,name="")
  + size()
  + peek()
  + get()
  + put()
* MyBag (thread-unsafe bag (i.e. set where duplicates are allowed))
  + \_\_init\_\_(maxsize,name="")
  + size()
  + contains()
  + get()
  + put()

# Room for improvement

M:

* Display of MyConditionVariable: a.o. status of the mutex, which threads are notified
* proper management of the mutex value, especially when a condition variable has released the lock
* when there are several threads blocked for a semaphore or mutex: setting priority & enforce that a specific thread will be released
* make a recording of a simulation
* reverse a simulation (especially when a deadlock has occurred unexpectedly…)

S:

* scrolling of the breakpoints for larger Dut code
* smaller panel for variables (now it has the same width as the code panel)
* Environment.py is quite large; perhaps split of Gui and MyXxx
* doxygen of the Environment.py code, and generate full doc

C:

* support for enum

W:

* Quit button doesn't work properly
* restart a DUT