```python
import string, SvtSim, SvtVme, sys, time, whrandom
from SvtVme import *

def hexfiledata(fnam):
  d = string.split(string.join(map(lambda x: x[:-1], open(fnam).readlines())))
  return map(lambda x: string.atoi(x, 16), d)

def getevent(d, n):
  out = []
  for x in d:
    if n==0:
      out.append(x)
    if (x&0x600000)==0x600000:
      n = n-1
      if n<0: break
  return out

def fakehits(rng, maps):
  r = int(rng.random()*32768)
  patt = maps.patt(r)
  print "patt #%d (hex=%x)"%(r, r), patt
  hits = []
  for layer in range(len(patt)):
    ss = patt[layer]
    if ss<0: continue
    dssdz = maps.dSSdz(layer)
    barrel = 0
    if dssdz>0:
      barrel = ss/dssdz
    hit = maps.ssEdge(layer, ss) | barrel<<15 | layer<<18
    if layer==5:
      xfthit = (ss-100)*8 + 192*maps.wedge()  # icky icky icky
      curv = 0
      xfthit = xfthit | curv<<12
      hit = ss<<4 | layer<<18
      hits.append(hit)
      hits.append(xfthit | 0x200000)
    else:
      hits.append(hit | 0x200000)
    print "  layer %d barrel %d hit %x"%(layer, barrel, hit)
  p = 0
  for x in hits:
    p = p^x
  q = 0
  for i in range(23):
    q = q^(p>>i&1)
  bc = int(rng.random()*256)
  hits.append(0x600000 | bc | q<<8 | 0x1000)  # kludge: set invData for now
  print "  --> hits:", string.join(map(lambda x: "%6.6x"%(x), hits))
  return hits

def main():
  wedge = 6
  docs = not (len(sys.argv)>1 and sys.argv[1]=="skipcs")
  rng = whrandom.whrandom()
  rng.seed(3, 2, 1)
  if docs:
    print "initializing wedge crate ..."
    crate = "b0svt0%d"%(wedge/2)
    sc_metakey = "wja/spy_w%2.2dw%2.2d"%((wedge/2)*2, (wedge/2)*2+1)
    sc0_cstest(crate, 3, 1, sc_metakey, "")
    ams_cstest(crate, 8+9*(wedge%2), 1, "wja/ams_w%2.2d"%(wedge), "")
    amb_cstest(crate, 9+9*(wedge%2), 1, "wja/amb_w%2.2d_0"%(wedge), "")
    amb_cstest(crate, 10+9*(wedge%2), 1, "wja/amb_w%2.2d_1"%(wedge), "")
    hb_cstest(crate, 12+9*(wedge%2), 1, "wja/hb_w%2.2d"%(wedge), "")
    sc_cstest(crate, 3, 1, sc_metakey, "")
    print "running TF coldstart ..."
    tf_cstest("b0svt06", wedge+4, 0, "wja/tf_w%2.2d"%(wedge), "")
  print "sim TF ..."
  maps = SvtSim.WedgeMaps()
  maps.initFromMapSet(wedge, "nom_regular_XFTM_20010316")
  if 0:
    maps.writeRams()
```

```
  incable = SvtSim.Cable()
  amss = SvtSim.SimAMS(wedge)
  amss.plugInput(incable)
  hbs = SvtSim.SimHB()
  hbs.plugHitInput(incable)
  hbs.plugRoadInput(amss.outputCable())
  tfs = SvtSim.SimTF(wedge)
  tfs.plugInput(hbs.outputCable())
  outcable = tfs.outputCable()
  for bs in [amss, hbs, tfs]:
    bs.useMaps(maps)
  tf = Board("b0svt06", wedge+4, TF)
  mrg = Board("b0svt0%d"%(wedge/2), 7+9*(wedge%2), MRG)
  print "splitting input data into events ..."
  eventindex = 0
  events = []
  eventindex = 0
  while 1:
    d = fakehits(rng, maps)
    incable.copyWords(d)
    for bs in [amss, hbs, tfs]:
      bs.procEvent()
    print "tfin: ", string.join(map(lambda x: "%6.6x"%(x), hbs.outputCable().data()))
    print "tfout:", string.join(map(lambda x: "%6.6x"%(x), outcable.data()))
    print "real TF ..."
    mrg.init()
    mrg.setTmode()
    tf.setTmode()
    tf.setState(TF_RUNMODE, 0)
    tf.setState(TF_IGNR_HOLD, 1)
    tf.setState(TF_FRZGEN, 1)
    tf.resetSpy(TF_ISPY)
    tf.resetSpy(TF_OSPY)
    tf.setState(TF_FRZGEN, 0)
    tf.setState(TF_INIT, 1)
    tf.setState(TF_INIT, 0)
    tf.init()
    tf.setState(TF_RUNMODE, 1)
    mrg.sendDataOnce(d)
    tf.setTmode()
    tf.setState(TF_RUNMODE, 0)
    ispy = map(lambda x: x&0x7fffff, tf.readSpyTail(TF_ISPY))
    ospy = map(lambda x: x&0x7fffff, tf.readSpyTail(TF_OSPY))
    tf.setState(TF_FRZGEN, 0)
    print "tfin: ", string.join(map(lambda x: "%6.6x"%(x), ispy))
    print "tfout:", string.join(map(lambda x: "%6.6x"%(x), ospy))
    if ispy==hbs.outputCable().data():
      print "ispy OK"
    else:
      print "ispy NOT ok"
    if ospy==outcable.data():
      print "ospy OK"
    else:
      print "***********************************************************"
      print "***********************************************************"
      print "***********************************************************"
      print "ospy NOT ok ***********************************************"
      print "***********************************************************"
      print "***********************************************************"
      print "***********************************************************"
    eventindex = eventindex+1
    time.sleep(0.05)

if __name__=="__main__":
  main()
```