GEM5 Simulator in Full System Mode(3)

Yizi Gu

Tsinghua University

yizigu@gmail.com

November 3, 2014

Overview

1. The Python scripts in GEM5
   - Python scripts in GEM5
   - How to run Python script in GEM5
2. How to add a device
   - I/O device base classes
   - Add a I/O device

Python scripts in GEM5

We have known already that Python scripts are involved in GEM5.

Example

- For running simulation: fs.py, se.py ,etc.
- For configuration: O3_ARM_v7a.py, FSConfig.py, etc.
- For building simulation objects: Bus.py, Terminal.py,etc.

It is obvious that a Python interpreter has to be called so as to run these scripts. But how is the interpreter called?

Embedding Python in the application

src/main.cc

```
#include <Python.h>
#include "sim/init.hh"
int main(int argc, char **argv) {
    int ret;
    initSignals();
    Py_SetProgramName(argv[0]);
    // initialize embedded Python interpreter
    Py_Initialize();
    // Initialize the embedded m5 python library
    ret = initM5Python();
    if (ret == 0) {// start m5
        ret = m5Main(argc, argv);
    }
    Py_Finalize();
    return ret;
}
```

Deriving the class from I/O device base classes

I/O device base classes

$$\left.\begin{array}{r} PIO : BasicPioDevice \\ DMA : DmaDevice \end{array}\right\} \leftarrow PioDevice \leftarrow MemObject \leftarrow SimObject$$

Create[1]

- Object header file: src/dev/hellodevice.hh
- Object C++ file: src/dev/hellodevice.cc
- Parameters file: src/dev/HelloDevice.py

Modify[1]

- SConscript: src/dev/SConscript
- Configuration: configs/common/FSConfig.py

Cont'd

src/dev/hellodevice.hh

```
#pragma once
#include "dev/io_device.hh"
#include "params/HelloDevice.hh"
class BadDevice : public BasicPioDevice {
  private:
    std::string devname;
  public:
    typedef BadDeviceParams Params;
  protected:
    const Params *
    params() const {
        return dynamic_cast<const Params *>(_params);
    }
  public:
    BadDevice(Params *p);
    virtual Tick read(PacketPtr pkt);
    virtual Tick write(PacketPtr pkt);
};
```

Cont'd

src/dev/HelloDevice.py

```
from m5.params import *
from Device import BasicPioDevice

class BadDevice(BasicPioDevice):
    type = 'HelloDevice'
    cxx_header = "dev/hellodev.hh"
    devicename = Param.String("DeviceName")
```

Cont'd

Change in FSConfig.py

```
self.realview.attachOnChipIO(self.membus, self.bridge)
self.realview.attachIO(self.iobus)
self.realview.baddev= BadDevice(
    pio_addr=0x10000000000,devicename="baddev"
)
self.realview.baddev.pio = self.iobus.master
self.intrctrl = IntrControl()
```

Writing relevant driver

Requesting hte I/O port using *request_region* function

```
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/ioport.h>
#include <asm/io.h>
#define BASE 0x10000000000
#define SIZE 0x08
void cleanup_module(void) { release_region(BASE,SIZE); }
int init_module(void) {
    int t1;
    if(!request_region(BASE,SIZE,"hellodevice")) {
        printk(KERN_INFO, "unable ot get io port at 0x%8X\n"
    ,BASE);
        return -ENODEV;
    }
    outl(0,BASE);t1 = inl(BASE);
    printk(KERN_INFO,"read %d\n",t1);
    return 0;
}
```

Problem encountered

After insmod, the system complains that the device is not found:

```
# insmod hellodevice.ko
insmod hellodevice.ko
insmod: can't insert 'hellodevice.ko': No such device
#
```

Reference

 Ali Saidi,et al, *Using the M5 Simulator ASPLOS XIII*.

Thank you