一. 实验题目

DOL开发环境配置

二. 实验目的

1. **学习如何配置ubuntu环境**
2. **学习如何在ubuntu下程序的安装与调试**

三. 实验工具

1. **Make工具简介**
   - 在Linux和Ubuntu环境中，make工具主要被用来进行工程编译和程序链接
   - Makefile文件：告诉make以何种方式编译源代码和链接程序
   - make通过比较对应文件（规则的目标和依赖）的最后修改时间，来决定哪些文件需要更新、那些文件不需要更新。
2. **Ant工具简介**
   - Ant是一种基于Java的build工具
   - Ant用Java的类来扩展
   - Ant本身就是这样一个流程脚本引擎，用于自动化调用程序完成项目的编译，打包，测试等
     - Ant的优点
       - 跨平台性。Ant是纯java语言编写的，所示具有很好的跨平台性
       - 操作简单。Ant是由一个内置任务和可选任务组成的。Ant运行时需要一个XML文件(构建文件)
       - 容易维护和书写，结构清晰
       - Ant可以集成到开发环境中
3. **Java与javac简介**
   - 用途：编译或执行java代码
   - javac命令用来编译java文件
   - java命令可以执行生成的class文件
4. **dol和systemc**

   DOL是一种能够自动将应用映射到多处理器SHAPES处理平台上的框架。DOL由三部分组成：

   - **DOL应用程序编程接口：** DOL定义了一组计算和通信程序，可以对SHAPES平台的分布式并行应用程序进行编程。使用这些例程，应用程序员可以编写程序，而不需要有关底层架构的详细知识。
   - **DOL功能模拟：** 为了向程序员提供测试应用程序的可能性，已经开发了功能模拟框架。除应用程序的功能验证外，该框架还用于在应用程序级别获取性能参数。
   - **DOL映射优化：**DOL映射优化 的目标是将应用程序的一组最佳映射计算到SHAPES架构平台上。在第一步中，已经定义了基于XML的规范格式，允许在抽象级别描述应用程序和架构。

   System C是一种软/硬件协同设计语言，一种新的系统级建模语言。它包含了一系列C++的类和宏，并且提供了一个事件驱动的模拟核，使得系统的设计者能够用C++的词法模拟并行的进程，特别是在SoC系统中。SystemC是在C++的基础上扩展了硬件类和仿真核形成的，由于结合了面向对象编程和硬件建模机制原理两方面的优点，这可以使SystemC在抽象层次的不同级进行系统设计系统硬件部分可以用SystemC类来描述，其基本单元是模块（modul）模块内可包含子模块、端口和过程，模块之间通过端口和信号进行连接和通讯。

本次实验就是DOL通过systemC运行进行并化处理，生成3个进程分别是生产者，平方和消费者。进行运行输出。

四. 实验过程

1. **安装配置DOL所需要的环境**
    1. 环境的配置（先更换国内源为好）

```
$   sudo apt-get update
$   sudo apt-get install ant
$   sudo apt-get install default-jdk
$   sudo apt-get install unzip
$   sudo apt-get install default-jre
$   sudo apt-get install graphviz
$   sudo apt-get install xdot
```

    2. 获取DOL和systemc

```
$   sudo wget http://www.accellera.org/images/downloads/standards/systemc/systemc-2.3.1.tgz
$   sudo wget http://www.tik.ee.ethz.ch/~shapes/downloads/dol_ethz.zip
```

2. **安装DOL**
    1. 新建DOL文件夹，解压dol_ethz和systemc

```
$   sudo mkdir dol
$   sudo unzip dol_ethz.zip -d dol
$   sudo tar -zxvf systemc-2.3.1.tgz
```

    2. 编译systemc，进入systemc文件夹，新建临时文件夹objdir，configure设置编译编译参数

```
$   cd systemc-2.3.1
$   sudo mkdir objdir
$   cd objdir
$   sudo ../configure CXX=g++ --disable-async-updates
```

    此处如果提示没有g++只需要执行下面命令

```
$   sudo apt-get install g++
```

    3. 新建DOL文件夹，解压dol_ethz和systemc

```
$   sudo mkdir dol
$   sudo unzip dol_ethz.zip -d dol
$   sudo tar -zxvf systemc-2.3.1.tgz
```

4. 编译systemc，进入systemc文件夹，新建临时文件夹objdir，使用configure设置编译编译参数

```
$   cd systemc-2.3.1
$   sudo mkdir objdir
$   cd objdir
$   sudo ../configure CXX=g++ --disable-async-updates
```

configure后效果如下：

```
Build settings:
  Enable compiler optimizations  : yes
  Include debugging symbols      : no
  Coroutine package for processes: QuickThreads
  Disable async_request_update   : yes
  Phase callbacks (experimental) : no
  Additional settings            :

------------------------------------------------------------------------
WARNING: The selected SystemC library configuration is non-conforming
         to IEEE Std. 1666-2011. See INSTALL.
------------------------------------------------------------------------
```

5. 使用make进行编译

```
$   sudo make install
$   cd ..
$   ls
```

编译之后目录如下：

```
root@DESKTOP-CHONOR:~/systemc-2.3.1# ls
aclocal.m4  config        COPYING   include      LICENSE      msvc80  README
AUTHORS     configure     docs      INSTALL      Makefile.am  NEWS    RELEASENOTES
ChangeLog   configure.ac  examples  lib-linux64  Makefile.in  objdir  src
```

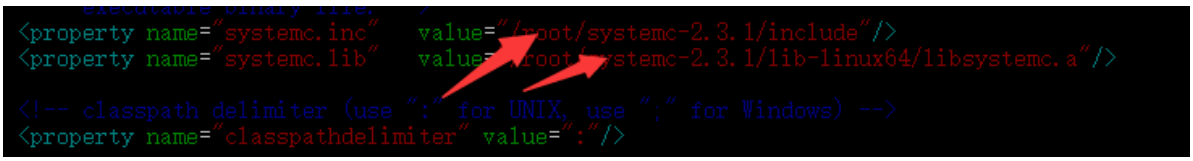
6. 记录工作路径用于之后修改build_zip.xml

```
$   sudo pwd
```

```
root@DESKTOP-CHONOR:~/systemc-2.3.1# pwd
/root/systemc-2.3.1
```

7. 修改build_zip.xml

```
$   cd ../dol
$   sudo vim build_zip.xml

<property name="systemc.inc" value="YYY/include"/>
<property name="systemc.lib" value="YYY/lib-linux/libsystemc.a"/>
######把YYY改成记录的路径
######如果是X64改lib-linux为lib-linux64
```



8. 编译DOL

```
$   sudo ant -f build_zip.xml all
```

编译结果如下：

```
Buildfile: /root/dol/build_zip.xml

showantversion:
     [echo] Use Apache Ant(TM) version 1.9.6 compiled on July 8 2015.

showjavaversion1:
     [echo] Use Java version 1.8.0_131 (required version: 1.5.0 or higher).

showjavaversion2:

config:
     [echo] Create new dol.properties file.
     [copy] Copying 1 file to /root/dol/bin
      [jar] Updating jar: /root/dol/bin/dol.jar
   [delete] Deleting: /root/dol/bin/dol.properties

compile:
     [echo] Create build directory tree.
     [copy] Copying 1 file to /root/dol/build/bin/main

updatebuildxml:

updatebuildxml1:

updatebuildxml2:

all:

BUILD SUCCESSFUL
Total time: 3 seconds
root@DESKTOP-CHONOR:~/dol#
```

9. 测试DOL

```
$   cd build/bin/main
$   sudo ant -f runexample.xml -Dnumber=1
```

测试结果如下：

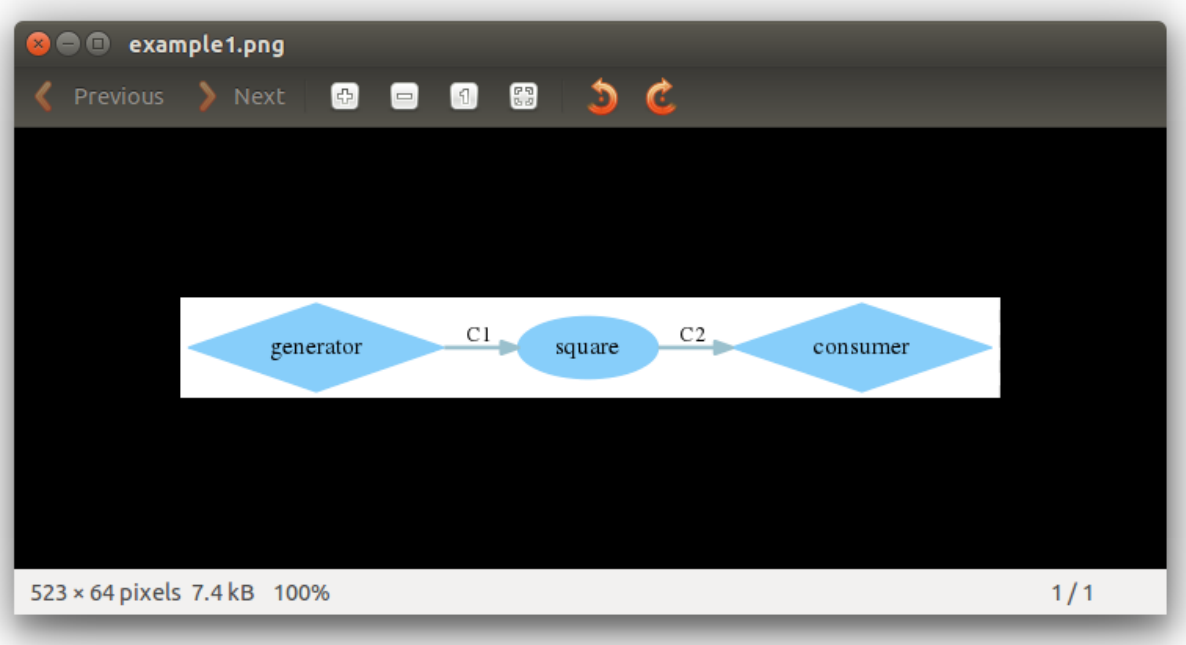
```
do12:

execute:
     [echo] Make HdS application.
     [exec] make: Nothing to be done for 'all'.
     [echo] Run HdS application.
   [concat] consumer: 0.000000
   [concat] consumer: 1.000000
   [concat] consumer: 4.000000
   [concat] consumer: 9.000000
   [concat] consumer: 16.000000
   [concat] consumer: 25.000000
   [concat] consumer: 36.000000
   [concat] consumer: 49.000000
   [concat] consumer: 64.000000
   [concat] consumer: 81.000000
   [concat] consumer: 100.000000
   [concat] consumer: 121.000000
   [concat] consumer: 144.000000
   [concat] consumer: 169.000000
   [concat] consumer: 196.000000
   [concat] consumer: 225.000000
   [concat] consumer: 256.000000
   [concat] consumer: 289.000000
   [concat] consumer: 324.000000
   [concat] consumer: 361.000000

BUILD SUCCESSFUL
Total time: 8 seconds
root@DESKTOP-36KH67B:~/do1/build/bin/main#
```

10. 使用DOT生成图片

```
$   cd example1
$   dot -Tpng example1.dot -o example1.png
```

此处png可以改成jpg等 -T后面也需要同步改(之前如果没有安装graphviz)此时会提示安装，运行之后dol/build/bin/main/example1 中会产生一张图片。命令行界面需要就比较麻烦需要安装一个图片查看器才能查看，图形界面也可以直接安装xdot直接查看。

图片如下：



五. 代码分析

1. 直接

```
$   sudo cd examples/example1
```

我们能看到相应的example1的xml和example1的代码

看到global里面只有

```
#define LENGTH 20
```

这里定义了我们上面的运行次数

在看到generator的执行

```
    if (p->local->index < p->local->len) {
        float x = (float)p->local->index;
        DOL_write((void*)PORT_OUT, &(x), sizeof(float), p);
        p->local->index++;
    }

    if (p->local->index >= p->local->len) {
        DOL_detach(p);
        return -1;
    }
```

没干什么就是一直在把数(当前次数)送出去然后次数++，次数到达设定上限后结束。

然后看到square的执行部分

```
    if (p->local->index < p->local->len) {
        DOL_read((void*)PORT_IN, &i, sizeof(float), p);
        i = i*i;
        DOL_write((void*)PORT_OUT, &i, sizeof(float), p);
        p->local->index++;
    }

    if (p->local->index >= p->local->len) {
        DOL_detach(p);
        return -1;
    }
```

读数然后将数平方然后送出，次数++，到达次数上限结束

在看到consumer执行部分

```
    if (p->local->index < p->local->len) {
        DOL_read((void*)PORT_IN, &c, sizeof(float), p);
        printf("%s: %f\n", p->local->name, c);
        p->local->index++;
    }

    if (p->local->index >= p->local->len) {
        DOL_detach(p);
        return -1;
    }
```

读数然后输出消费者，和读到的数，次数++ 达到次数上限后结束。

最后看到example1.xml

```
<!-- processes -->
  <process name="generator">
    <port type="output" name="1"/>
    <source type="c" location="generator.c"/>
  </process>
.................................
```

这里有3个进程创建对应上面三个文件，在.dot中分别对应3个图形

```
  <!-- sw_channels -->
    <sw_channel type="fifo" size="10" name="C1">
    <port type="input" name="0"/>
    <port type="output" name="1"/>
  </sw_channel>
.................................
```

此处在.dot文件中看起来就是2条线，其实实际作用就是输入输出通道。

```
<!-- connections -->
<connection name="g-c">
  <origin name="generator">
    <port name="1"/>
  </origin>
  <target name="C1">
    <port name="0"/>
  </target>
</connection>
```

此处就是讲输入输出通道和上面的3个进程分别对应连接起来。

六. 实验感想

本次实验就是简单的安装配置dol，主要就是配置的过程需要注意一下，其他的可以无脑安装，其实这种安装方式说真的完全可以做一个全自动的...没有什么大的问题，所有常见的问题也都是日常安装软件的时候经常看到的各种诡异的问题。因为日常进入Ubuntu直接root权限，所以没有出现权限问题导致的编译问题。

实验中唯一的问题是这东西教程最后只有到生成.dot文件，这然我这个命令行的不知如何找到图，最后去找了一下dot输出成图片的方法然后从ubuntu中把图拖出来。不过用图形界面就好办多了，直接打开。

这里的代码看起来也不是很复杂，很容易就能看出他是干什么的。