

中山大学数据科学与计算机学院

嵌入式实验报告

(2016 学年秋季学期)

教学班级	M2	专业（方向）	移动互联网
学号	14353232	姓名	马发潮

本次实验是要学会 dol 的配置过程

Sudo apt-get update

Sudo apt-get install ant

Sudo apt-get install openjdk-7-jdk

Sudo apt-get install unzip

首先将一些必要的安装环境安装好，再开始进行配置

我们将 ta 给的两个文件夹放在了 /home/embed/下面

Cd embed/systemc-2.3.1 进入 systemc-2.3.1 的目录下

新建一个临时文件夹 objdir

Sudo mkdir objdir

进入该文件夹 objdir

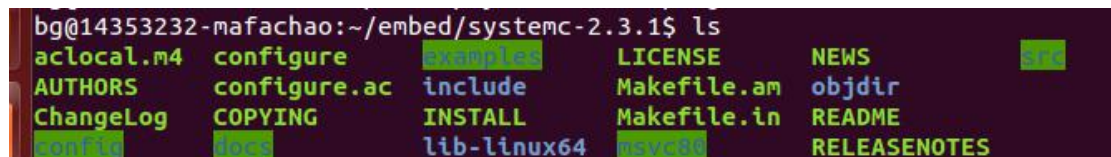
cd objdir

运行 configure

Sudo ../configure CXX=g++ --disable-async-updates

Sudo make install 编译 systemc

Ls 文件目录如下



\$cd ../dol 进入 embed 中的 dol 文件夹

修改 build_zip.xml 文件

找到下面这段话，就是说上面编译的 systemc 位置在哪里，

<property name="systemc.inc" value="YYY/include"/>

<property name="systemc.lib" value="YYY/lib-linux/libsystemc.a"/>

修改为

<property name="systemc.inc" value="/home/embed/systemc-2.3.1/include"/>

<property name="systemc.lib" value="/home/embed/systemc-2.3.1/lib-linux64/libsystemc.a"/>

然后是编译

sudo ant -f build_zip.xml all

约成功会显示 build successful

```
updatebuildxml1:
updatebuildxml2:
all:
BUILD SUCCESSFUL
Total time: 3 seconds
```

进入 build/bin/main 路径下

```
$cd build/bin/main
```

然后运行第一个例子

```
$sudo ant -f runexample.xml -Dnumber=1
```

```
[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
```

我们发现他的每个输出是每个 count 的平方，查看 example 中 example1 的代码我们可以知道，它的 square 写入端口的值是 $i*i$ ；我们将其改为 $i*i*i$ ，删除掉原来的 build 文件夹重新编译，查看效果

```

[concat] consumer: 0.000000
[concat] consumer: 1.000000
[concat] consumer: 8.000000
[concat] consumer: 27.000000
[concat] consumer: 64.000000
[concat] consumer: 125.000000
[concat] consumer: 216.000000
[concat] consumer: 343.000000
[concat] consumer: 512.000000
[concat] consumer: 729.000000
[concat] consumer: 1000.0000
[concat] consumer: 1331.0000
[concat] consumer: 1728.0000
[concat] consumer: 2197.0000
[concat] consumer: 2744.0000
[concat] consumer: 3375.0000
[concat] consumer: 4096.0000
[concat] consumer: 4913.0000
[concat] consumer: 5832.0000
[concat] consumer: 6859.0000

```

发现此时输出的是三次方了

然后运行第二个例子

`$sudo ant -f runexample.xml -Dnumber=2`

```

[concat] consumer: 0.000000
[concat] consumer: 1.000000
[concat] consumer: 256.000000
[concat] consumer: 6561.000000
[concat] consumer: 65536.000000
[concat] consumer: 390625.000000
[concat] consumer: 1679616.000000
[concat] consumer: 5764801.000000
[concat] consumer: 16777216.000000
[concat] consumer: 43046720.000000
[concat] consumer: 100000000.000000
[concat] consumer: 214358880.000000
[concat] consumer: 429981696.000000
[concat] consumer: 815730752.000000
[concat] consumer: 1475789056.000000
[concat] consumer: 2562890752.000000
[concat] consumer: 4294967296.000000
[concat] consumer: 6975757312.000000
[concat] consumer: 11019960320.000000
[concat] consumer: 16983563264.000000

```

我们可以发现它输出的是 2 的平方的三次方，查看 `square` 的代码，输出到端口的是二次方，查看 `example2.xml` 的代码我们可以发现，`square` 被迭代运行了三次，我们将迭代次数 `N` 的 `value` 改为 2，删除 `build` 文件夹重新编译

```
bg@14353232-marachao: ~/embed/d
[concat] consumer: 0.000000
[concat] consumer: 1.000000
[concat] consumer: 16.000000
[concat] consumer: 81.000000
[concat] consumer: 256.000000
[concat] consumer: 625.000000
[concat] consumer: 1296.000000
[concat] consumer: 2401.000000
[concat] consumer: 4096.000000
[concat] consumer: 6561.000000
[concat] consumer: 10000.000000
[concat] consumer: 14641.000000
[concat] consumer: 20736.000000
[concat] consumer: 28561.000000
[concat] consumer: 38416.000000
[concat] consumer: 50625.000000
[concat] consumer: 65536.000000
[concat] consumer: 83521.000000
[concat] consumer: 104976.000000
[concat] consumer: 130321.000000
```

我们可以发现，现在的 example2 输出的是 2 的平方的平方，square 的迭代次数为 2。
实验结束。