嵌入式系统导论实验报告

姓名	学号	班级	电话	邮箱
曹广杰	15352015	1501	13727022190	1553118845@qq.com

1. 实验题目

DOL实例分析与编程

2. 实验分析

1. example简介
   - **example是DOL配置之后，为使用者提供的使用示例**
   - **src内部储存表示各个处理模块的功能信息**
     - 实现的模块：包含.c文件和.h文件；
     - 模块的接口：`_`init（可选写，只执行一次）和`_`fire（必写，可能执行多次）；
   - **example1.xml文件表示模块之间的联系方式**
     - process表示框架。
     - sw_channel表示连接线。
     - connection表示连接关系，表示线的形态。
2. 进程定义

```
<process name="CodeBlockName">
    <port type="output" name="Port1_Name"/>
    <port type+"input"  name="Port2_name"/>
    <source type="c"    location="CodeBlockName.c"/>
</process>
```

Port1用于输出，而Port2用于输入

3. 通道定义

```
<sw_channel type="fifo" size="BufferSize" name="Channel_name">
    <port type="input"  name="port1"/>
    <port type="output" name="port2"/>
</sw_channel>
```

port依然表示端口，一个用于输入，另一个与输出

4. 连接定义

```
<connection name="ConnectionID">
  <origin name="Block1">
    <port name="Block1_port"/>
  </origin>
  <target name="Block2">
    <port name="Block2_port"/>
  </target>
</connection>
```

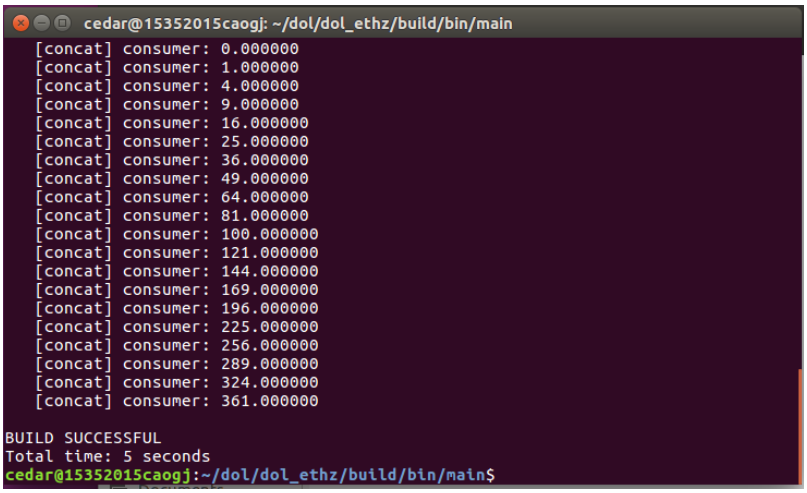
Block1是连接关系的起点，而Block2是连接关系的终点，连接线的两个端点分别为两个port

3. 实验过程

- [ ] 修改example1，使其输出3次方数

  运行测试所在路径：`~/dol/dol_ethz/build/bin/main`

  修改前输出结果如下：



  由于计算部分在square.c中：`i = i * i;`

  由此得到平方数，那么修改只需要改为`i = i * i * i;`即可——尽管如此，第一次输出的时候还是平方数，这是由于之前编译过的平方数信息依然存留在计算机文件夹内部，而DOL又没有make clean的功能，所以一定要手动删除example1文件夹（`~/dol/dol_ethz/build/bin/main`内），之后重新编译就可以获得理想的效果）。

- [ ] 修改example2，让3个square模块变成2个

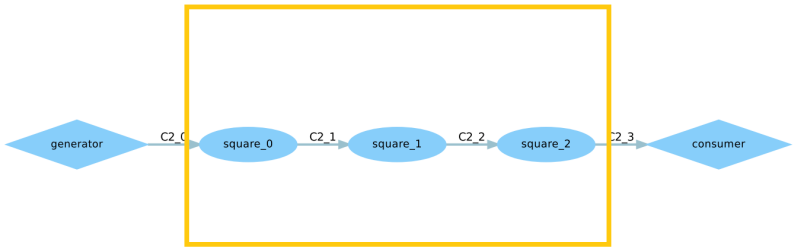
  1. 进入目录：`~/dol/dol_ethz`，运行编译语句：`sudo ant -f build_zip.xml all`
  2. 进入路径：`~/dol/dol_ethz/build/bin/main`，执行运行语句：`sudo ant -f runexample.xml -Dnumber=2`

     该目录以实验者的安装位置为准，笔者的安装目录如上罢了，事实上只要是进入dol_ethz文件夹与build/bin/main文件夹内运行即可
  3. 进入main目录下的example2文件夹内，找到“example2.dot”文件，双击打开（需实现安装xdot）

     安装xdot的命令行：

     > sudo apt-get install xdot

     得到文件如下示例：

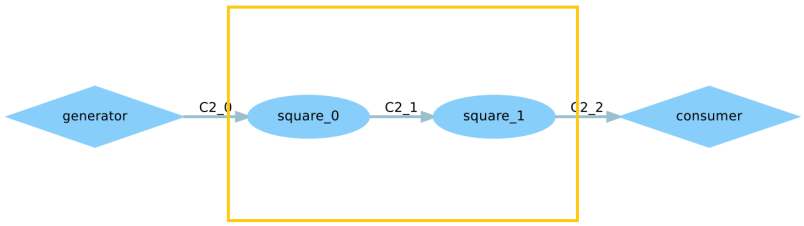


4. 进入main文件夹内部，找到example2的文件夹

> sudo rm -rf example2

5. 进入路径："/dol_ethz/examples/example2"，找到example2.xml文件，将前5行的所有数字3改为2
6. 如同步骤一的编译、执行。

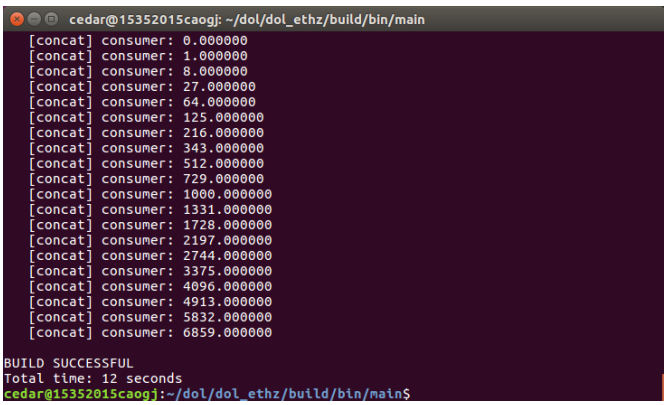
   可以得到输出的dot文件为example：



可以看到在dot文件中的square模块，很明显的变化——由3块变成2块。

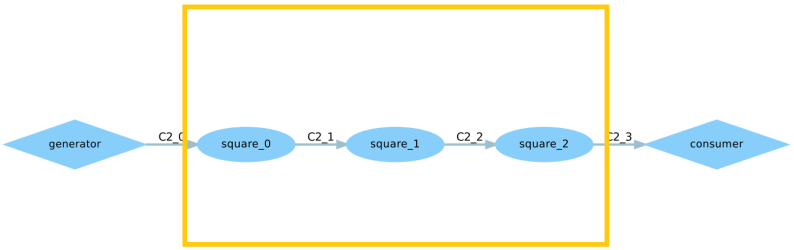
4. 实验结果汇总

- [ ] 3次方实验：

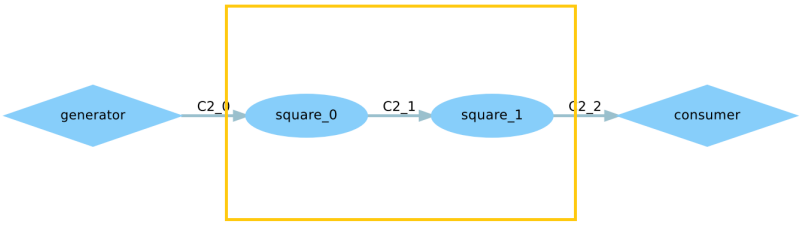


- [ ] 修改square数目的实验：

  修改之前：



修改之后：



可以很清楚地看到在修改之后的square模块的数量由3个变为了2个。

5. 实验心得

关于本次实验

本次实验开始对dol的环境的代码进行修改了，尽管对于代码的修改是需要对DOL的内部结构有一定的理解的，但是事实上要求并不高。

将example1中的平方计算转换为立方运算的实验中，只需要修改关键的计算语句即可。事实上对于DOL的全局结构要求并不高，不管我们怎么修改i的计算方式，在不修改原有的DOL结构的前提下，所有的计算和输出都将按照原本的状态运行，什么都不会影响。

在进行example2的实验的过程中，主要行为也就是找到example2的源文件——甚至都不需要阅读布局的内容，而直接把数值信息由3换成2即可，迭代输出不会有任何影响。

对于DOL的整体结构的感受，笔者在本次实验中对几个xml文件进行了修改，xml正是布局文件，由此笔者意识到，布局文件是dot文件生成的准则。

- 我们在DOL中编写的generator——无论是.c还是.h文件，都是对于某一种函数或者功能的实现
- 最后的拟合者使用xml对generator模块、consumer模块以及square模块按照语法实现连接即可
- 至此软件就会按照xml安排的顺序进行运行，而同时还可以查看dot文件了解整个系统的布局情况

关于编译原理

DOL是一个面向实时数据流和信号程序的软件，它基于多核处理器，具有独立性。

- 在计算层次，基于网络数据流模型[1]
- 在运行层次上，具有源对源的代码接口结构

使用数据流模型的优势

- 符合当前软件发展的趋势，可以利用当前软甲结构所提供的编译，分析工具
- 数据流的结构允许并行运算
- 对于软件分析，数据流的独有性质是确定性，如此就不需要全局的同步
- 对于软件的分析，模块的分类以及计算的精确拆分，符合分析模型的处理模式

DOL的映射结构

- 三者映射：进程、软件接口以及分享资源的列表
- DOL使用XML格式，用图形化界面对抽象的模块进行表述

综上，首先DOL允许使用者设计程序，但是又不必了解底层的信息。之后实现了功能性仿真框架，功能性仿真框架会从上层获得数据——即已经完成的编程中的数据。

使用DOL的映射结构，对映射操作进行映射，当然在这之前需要XML文件的权限请求，所以XML可以构图（见Chapter3），之后就是利用一系列的最优化映射计算将计算得到的结果映射到布局文件中，由此我们得到dot文件。

对应的编译过程：

条件，即首先需要使用的编译工具：

- ant（Apache Ant）
- java
- javac
- echo（始终用于输出中间信息）

过程：

1. 创建名为example1的布局文件——如果已经有存在的，并不删除或更新，需要手动更新
2. 从已有的C代码中获取代码
3. 综合已有的文件（.xml文件）并生成example1_flattened.xml文件
4. 对flatten2处理：
    - 运行DOL，获取process信息
    - 检查其他资源的信息（包括channel，各种resource...）
    - 整合（Generation）

5. 运行最后的Hds程序，并通过echo输出计算的结果

---

1. LEE,E.A. AND PARKS, T. M. 1995. Dataflow Process Networks. Proc. IEEE 83, 5, 773–799.↩