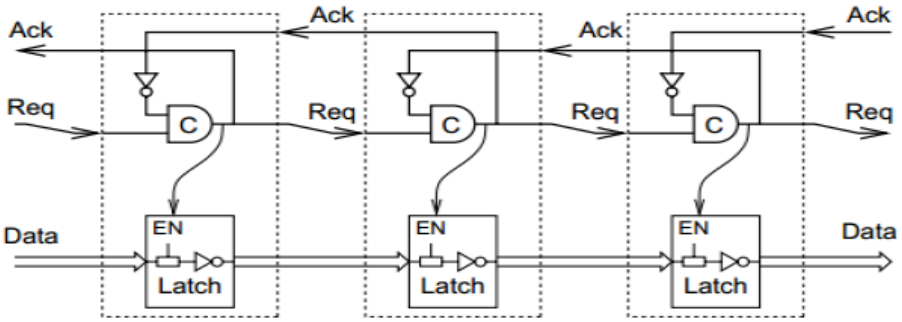
异步处理器设计实验报告

实验一 四段流水线设计

一、实验原理

本实验采用 C 单元实现四段握手协议，观察波形仿真。实验电路结构图如下所示：



图一 C 单元实现四段握手

二、实验步骤

本实验按照模块化设计思想进行设计。

Step1.muller_c 单元设计

Step2.Latch 设计

Step3.单个模块（图 1 虚线框）stage 设计

Step4.三个模块（stage）组合为顶层模块 stage3

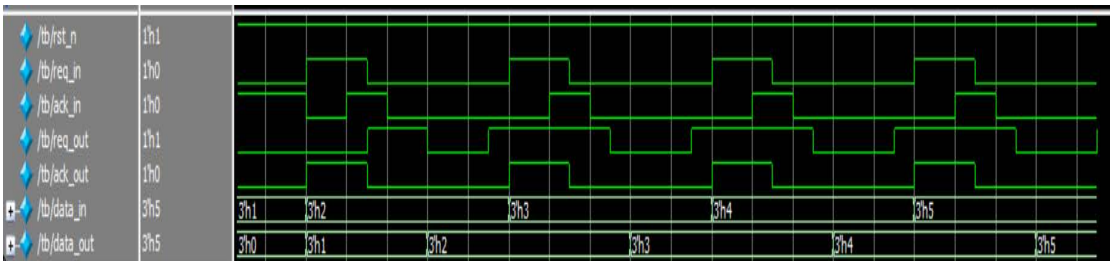
三、实验结果分析

1.编写上述设计对应的 testbench，输入数据 data_in 为 1，2，3,4,5，观察输出 data_out 的情况。

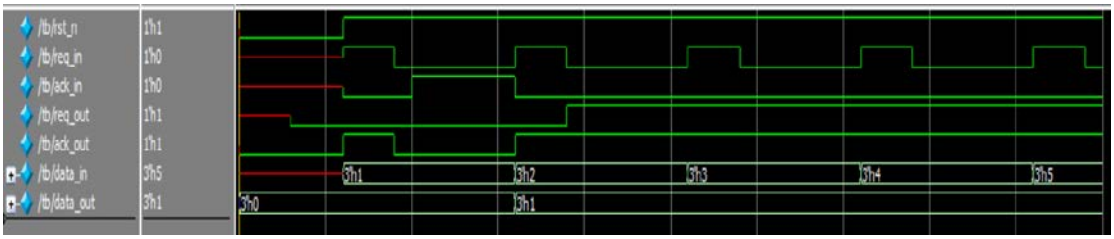
实验结果如下：

data_out 依次输出 1,2,3,4，5。

波形图如下：

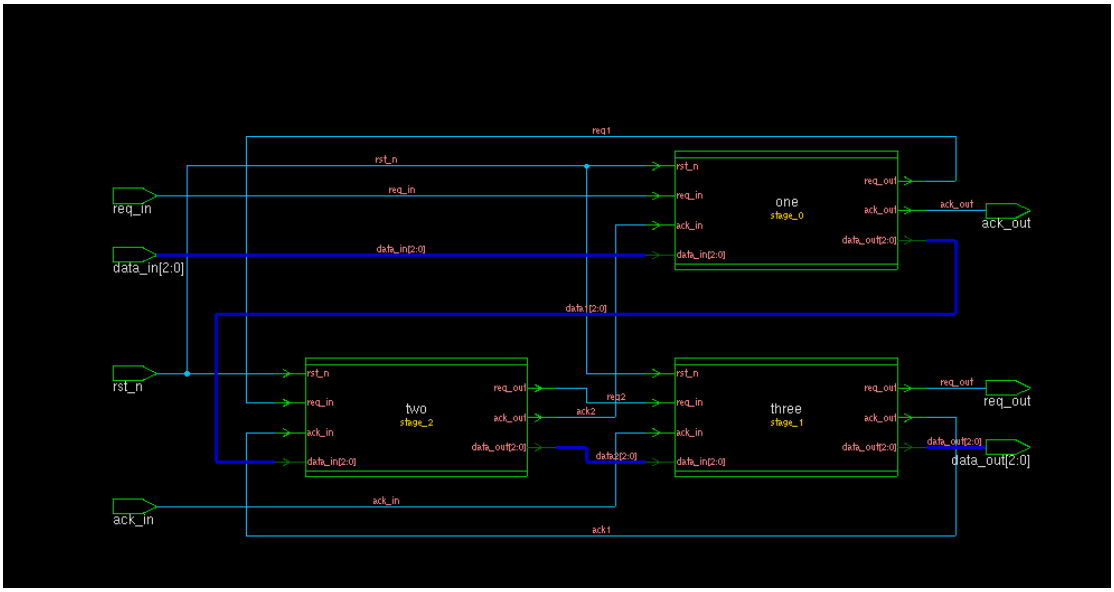


2.将应答 ack_in 始终置为 0，观察输出波形如下：

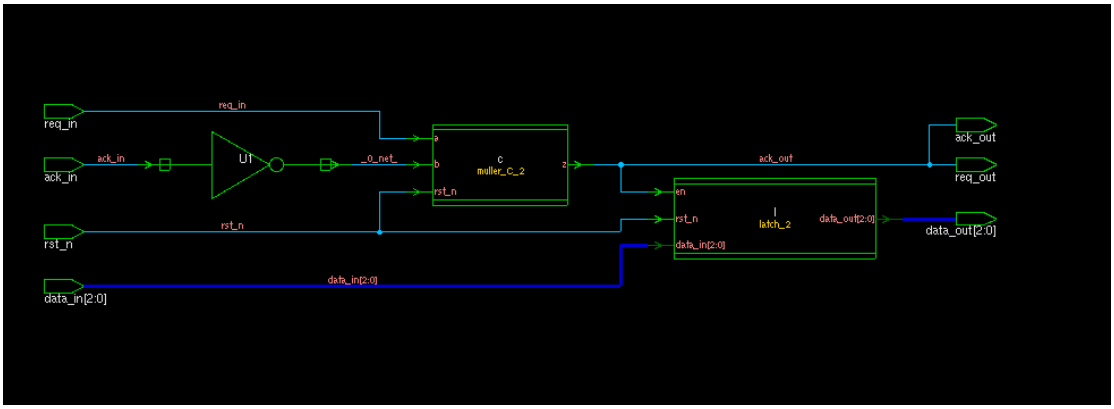


3.综合结果

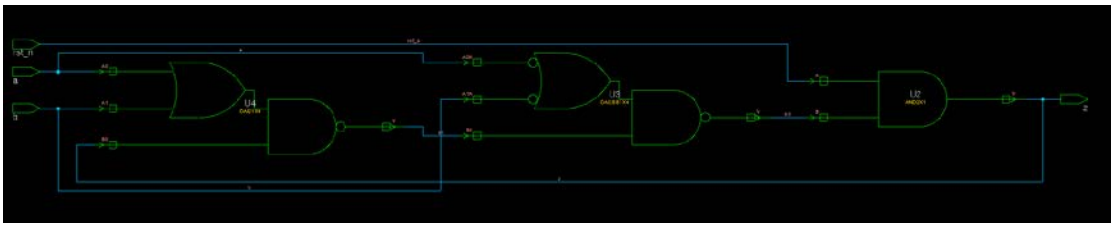
整体电路图如下：



Stage 电路图



Muller_C



实验二 petrify 实现四段流水线

一、实验原理

本实验采用的四段流水线 STG 图如图 1 所示。电路实现图如图 2。具体原理已在课堂学习，在此不再赘述。

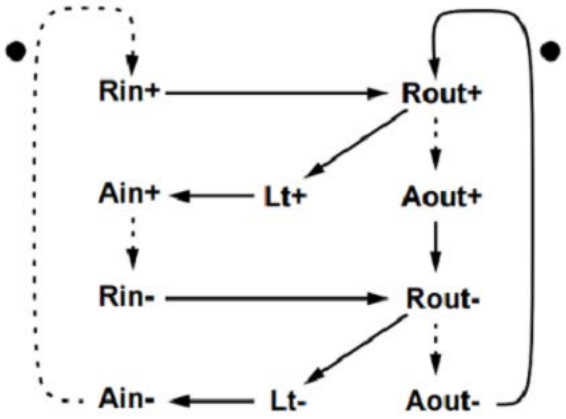


图 1 四段流水线 STG 图

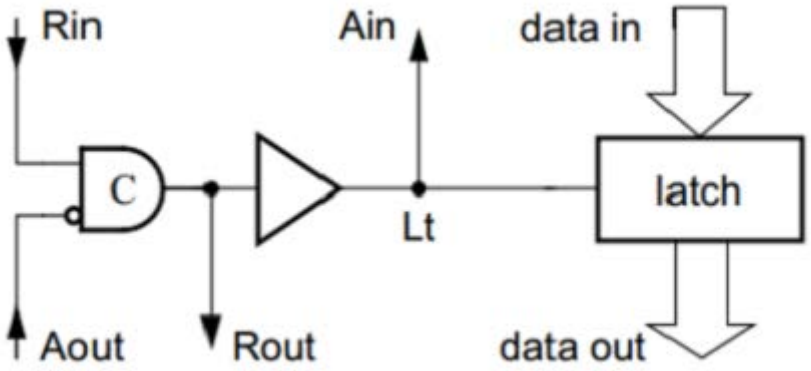


图 2 电路实现图

二、实验步骤

本实验按照模块化设计思想进行设计。

Step1. shakehand 单元设计

Step2. Latch 设计

Step3. 单个模块（图 2）stage 设计

Step4. 三个模块（stage）组合为顶层模块 stage3

下面详述 shakehand 单元的设计过程。

1. 将图一所示的 STG 图进行描述，得到 shakehand. g 文件

```
.inputs req_in ack_out
.outputs req_out ack_in Lt
.graph
req_in+ req_out+
req_out+ Lt+
req_out+ ack_out+
Lt+ ack_in+
ack_out+ req_out-
ack_in+ req_in-
req_in- req_out-
req_out- Lt-
req_out- ack_out-
Lt- ack_in-
ack_out- req_out+
ack_in- req_in+
.marking{<ack_out-,req_out+> <ack_in-,req_in+>}
.end
```

图 3 shakehand. g

2. 通过 petrify 工具将 shakehand. g 文件转为 shakehand. eqn 文件，命令为./petrify shakehand. g －eqn handshake. eqn －cg －no

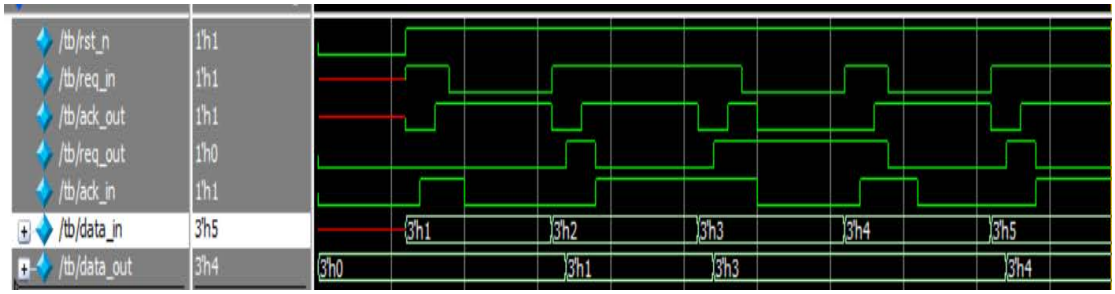
```
# EQN file for model handshake
# Generated by petrify 4.2 (compiled 15-Oct-03 at 3:06 PM)
# Outputs between brackets "[out]" indicate a feedback to input "out"
# Estimated area = 7.00

INORDER = req_in ack_out req_out ack_in Lt;
OUTORDER = [req_out] [ack_in] [Lt];
[req_out] = req_in (req_out + ack_out') + ack_out' req_out;
[ack_in] = Lt;
[Lt] = req_out;

# Set/reset pins: reset(req_out)
```

图 4 shakehand. eqn

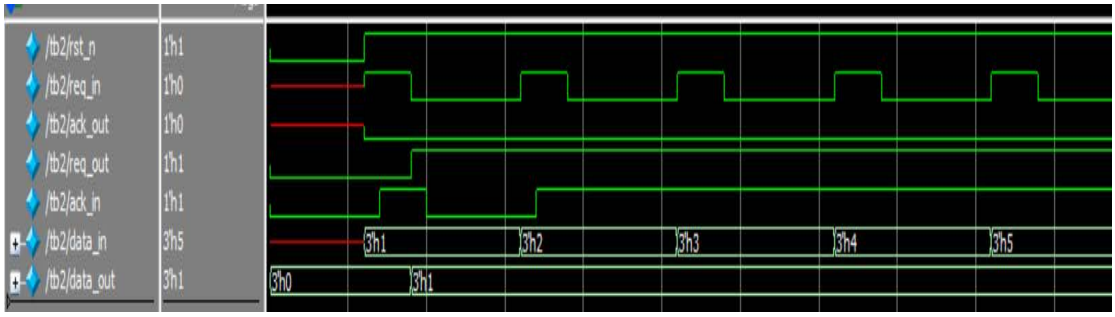
3. 将 shakehand. eqn 文件中的组合逻辑描述为 verilog 代码 shakehand. v

三、实验结果

1. 顺序输入 12345，顺序输出 12345。可见延迟。

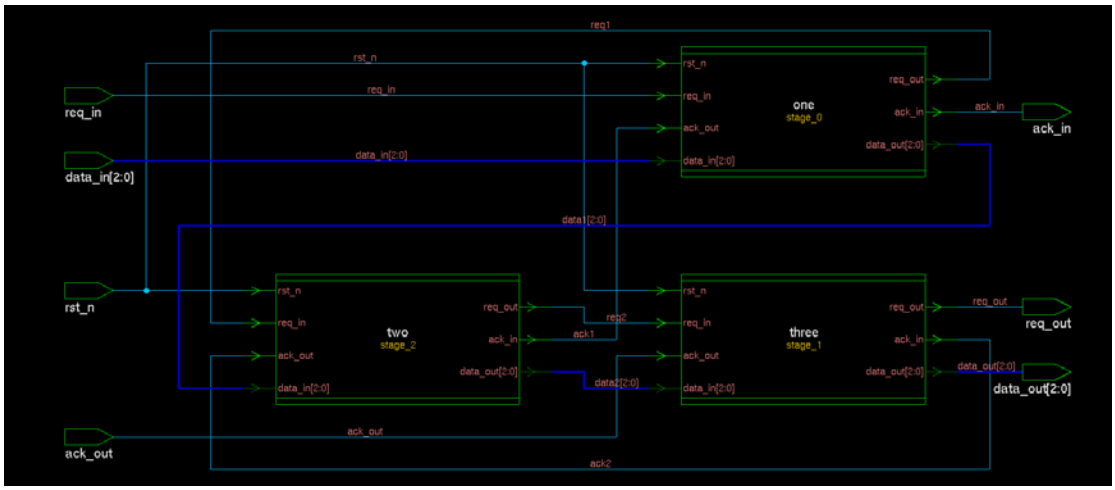


2. 应答时钟置为 0，结果流水线阻塞

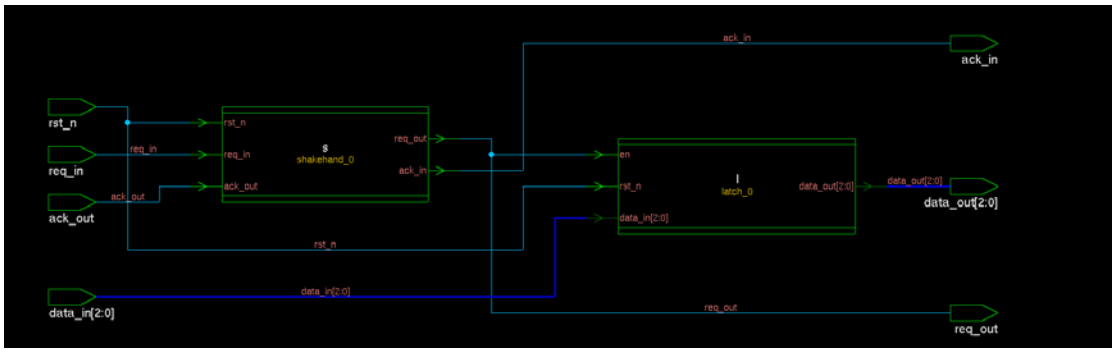


3. 综合结果

整体电路图



Stage 电路图



Shakehand 电路图