# VHDL作业：

## hw1

1、调查国内IC设计行业现状。有哪些好的设计公司，薪水情况，产品范围等。

2、寻找并安装软件：modelsim, quartus.

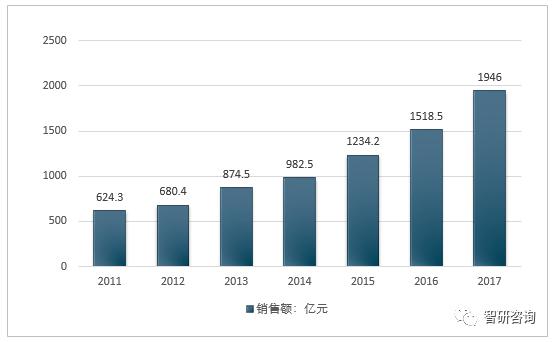
答：

**题目1.IC行业现状：**

IC设计是将系统、逻辑与性能的设计要求转化为具体的物理版图的过程，也是一个把产品从抽象过程一步步具体化、直至最终物理实现的过程。整个IC设计过程可以粗略的分为确定项目需求、系统级设计、前端设计、后端设计四部分。

2017年IC设计行业销售预计为1945.98亿元，比2016年的1518.52亿元增长28.15%。按照美元与人民币1:6的汇率，全年销售达到292.63亿美元，在全球集成电路设计业所占比重将再次大幅提高。

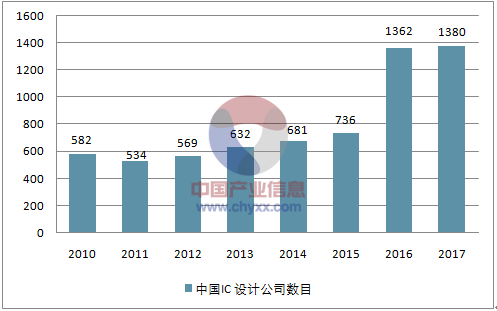
2017年我国IC设计行业销售额走势



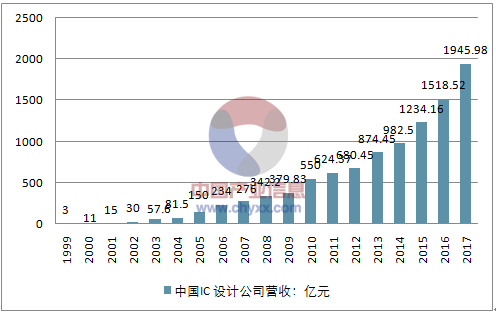
近年来，我国IC设计产业呈现井喷式增长的势头。截至2016年底，我国共有IC设计企业1362家，而2015年仅有736家，同比增长率高达85%。我国至今已有11家企业跻身全球IC设计企业前50强。

中国内芯片设计企业数量趋于稳定，截止2017年10月底，全国共有约1380家设计企业，比去年的1362家多了18家，总体变化不大。

2009-2017 年中国IC 设计公司数目（2017 为2017 年10 月）



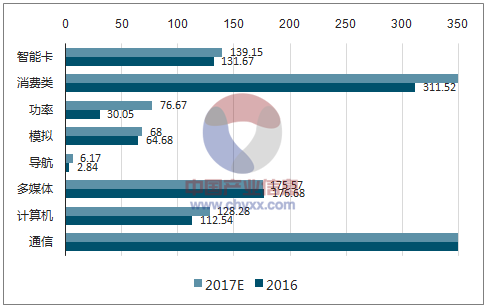
2014-2016 年中国IC 设计公司营收情况



从设计业销售变化情况来看， 2017年全行业销售预计为1945.98亿元，比2016年的1518.52亿元增长28.15%，在全球集成电路设计业所占的比重再次大幅提高。

2017年我国芯片设计业的从业人员规模与2016年的基本相同，略有增长，大约为14万人。因而得出人均产值139万元人民币。

2016-2017 年中国不同方向IC 设计的收入规模（单位：亿元）



截止2017年10月，据统计，产品领域分布情况来看，在通信、计算机、多媒体、导航、模拟、功率和消费电子等8个领域中，有5个领域的企业数量增加，3个领域的企业数量下降。从事通信芯片设计的企业从2016年的241家增加到266家，销售额增长了30.7%，达到899.74亿元。从事多媒体芯片设计的企业从去年的43家到增长72家，销售总额下降了0.63%，达到175.57亿元；从事模拟芯片设计的企业数量从219家减少到180家，销售额增长了5.14%，达到68亿元。从事功率器件设计的企业从77家增加到82家，销售额增长了155.14%，达到76.67亿元，为2017年增长最快的产品领域；从事消费类电子芯片设计的企业数量从上年的589家增加到610家，销售增长45.2%，达452.33亿元，继续保持了2016年的快速增长势头。从事智能卡芯片设计企业从上年的69家减少到62家，但销售总和上升了5.68%，达到139.15亿元。 从事计算机芯片设计的企业数量从去年的107家减少到85家，但销售提升了13.99%，达到128.28亿元。

IC设计企业，未来建议关注指纹识别芯片龙头企业汇顶科技、模拟IC设计公司圣邦股份、安防芯片设计的富瀚微等，这些公司研发实力强、创新能力强，有望着芯片国产化的东风伴随着国内市场一起成长。

**好的设计公司，薪水情况，产品范围：**

好的设计公司，像产业链上游的Fujitsu、Cisco、Synopsys、NXPSemiconductors

(Shanghai)Ltd.等，中下游的TI、ADI、Microchip、Maxim、Xilinx、Altera、Osram等，当然国内也有设计做得不错的企业，如中芯国际、炬力集成、华虹、中兴以及台湾的凌阳科技等。

1、锐迪科微电子有限公司

简介：锐迪科微电子有限公司成立于2004年4月，公司总部位于上海浦东张江高科技园区，并在北京设有研发设计分部，主要从事射频IC的设计、开发、制造、销售并提供相关技术咨询和技术服务。

网址：www.rdamicro.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612229086.jpg)

2、国民技术股份有限公司

简介：国民技术股份有限公司于2000年源于国家“909”集成电路专项工程成立，2010年4月在创业板上市（代码：300077），是我国安全芯片行业领军企业。

网址：www.nationz.com.cn

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612250905.jpg)

3、北京中星微电子有限公司

简介：中星微电子致力于开发先进的数字多媒体芯片技术，并成功地将“星光中国芯”系列芯片产品推向了国内外市场，应用于个人电脑、宽带、移动通讯、信息家电等高速成长的多媒体应用领域。

网址：www.vimicro.com.cn

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612274640.jpg)

4、展讯通信(上海)有限公司

简介：展讯通信致力于无线通信及多媒体终端的核心芯片、专用软件和参考设计平台的开发，为终端制造商及产业链其它环节提供高集成度、高稳定性、功能强大的产品和多样化的产品方案选择。

网址：www.spreadtrum.com.cn

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612296827.jpg)

5、华为海思半导体有限公司

简介：海思半导体成立于2004年10月，前身是创建于1991年的华为集成电路设计中心。其产品覆盖无线网络、固定网络、数字媒体等领域的芯片及解决方案，成功应用在全球100多个国家和地区；在数字媒体领域，已推出SoC网络监控芯片及解决方案、可视电话芯片及解决方案、DVB芯片及解决方案和IPTV芯片及解决方案。

网址：www.hisilicon.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612318892.jpg)

6、炬力集成电路设计有限公司

简介：炬力集成电路设计有限公司创立于2001年，2005年在美国纳斯达克挂牌上市,是一家致力于集成电路设计与制造的大型半导体技术集团。全球最大的个人便携多媒体SOC供应商之一。

网址：www.actions-semi.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612338688.jpg)

7、联芯科技有限公司

简介：联芯科技有限公司是大唐电信科技产业集团在集成电路设计板块的核心企业，专业从事2G，3G，4G移动互联网终端核心技术的研发与应用，提供3G/4G移动终端芯片及解决方案。

网址：www.leadcoretech.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612360341.jpg)

8、瑞芯微电子股份有限公司

简介： 瑞芯微电子是中国最具创新和务实的集成电路设计公司，为智能手机、平板电脑、通讯平板，电视机顶盒、车载导航、IoT物联网和多媒体音视频产品提供专业芯片解决方案，获得八届中国芯评选最高荣誉，在以技术创新推动个人便携终端产品向前发展方面扮演着重要的角色。

网址：www.rock-chips.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612392832.jpg)

9、敦泰科技（深圳）有限公司

简介：敦泰科技(FocalTech)专注于电容式触摸屏控制芯片的设计研发、制造及销售，是一家具有全球影响力的华人IC设计公司。FocalTech具备雄厚的技术实力且注重知识产权的积累和保护, 拥有200余项关于电容式触控技术的专利。

网址：cache.baiducontent.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612558655.jpg)

10、上海华虹集成电路有限责任公司

简介：上海华虹集成电路有限责任公司是专业的智能卡与信息安全芯片解决方案供应商，是中国“909 工程”的重要IC设计公司。华虹智能卡技术经历了四代发展，呈现出多应用、多界面、高安全性、自主可控、高通信速率、高处理能力、先进工艺、低成本等优势和特点。

网址：www.shhic.com

[](http://image.esm-cn.com/images/attachments/201511/original/9950297956_TIME_1448612580804.jpg)

11、格科微电子（上海）有限公司

简介：格科微电子（上海）有限公司位于中国集成电路设计和制造中心上海浦东张江高科技园区，由多名硅谷技术专家于 2003 年 9 月创立，专注CMOS图像传感器的设计开发和销售，产品主要应用于拍照手机、数码相机、 PC camera 、监视摄像系统以及玩具产品等。

网址：www.sh.xinhuanet.com

 12、珠海全志科技股份有限公司

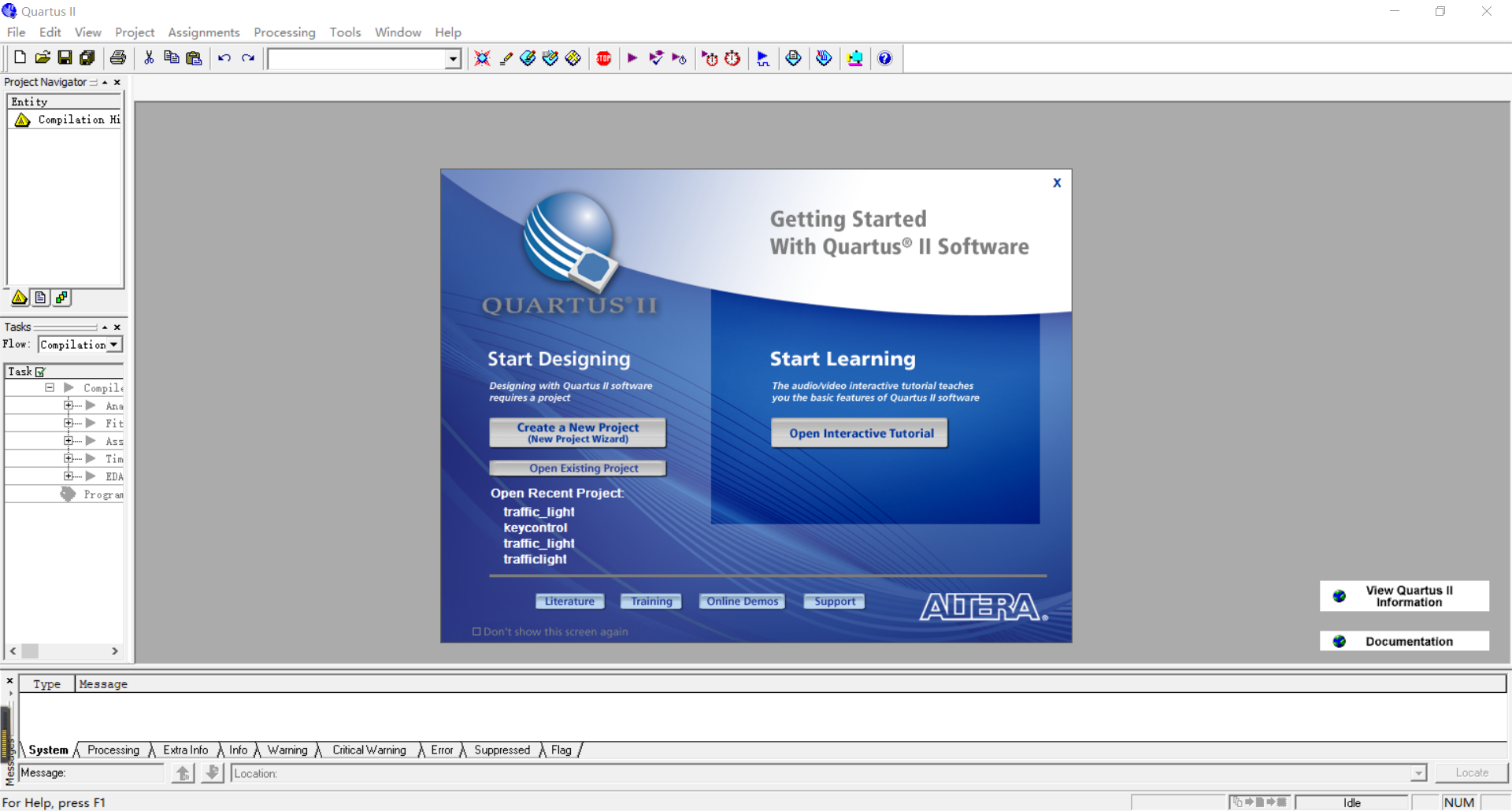
简介：珠海全志科技股份有限公司是领先的智能应用处理器SoC和智能模拟芯片设计厂商。公司主要产品为多核智能终端应用处理器、智能电源管理芯片等。

网址：www.allwinnertech.com

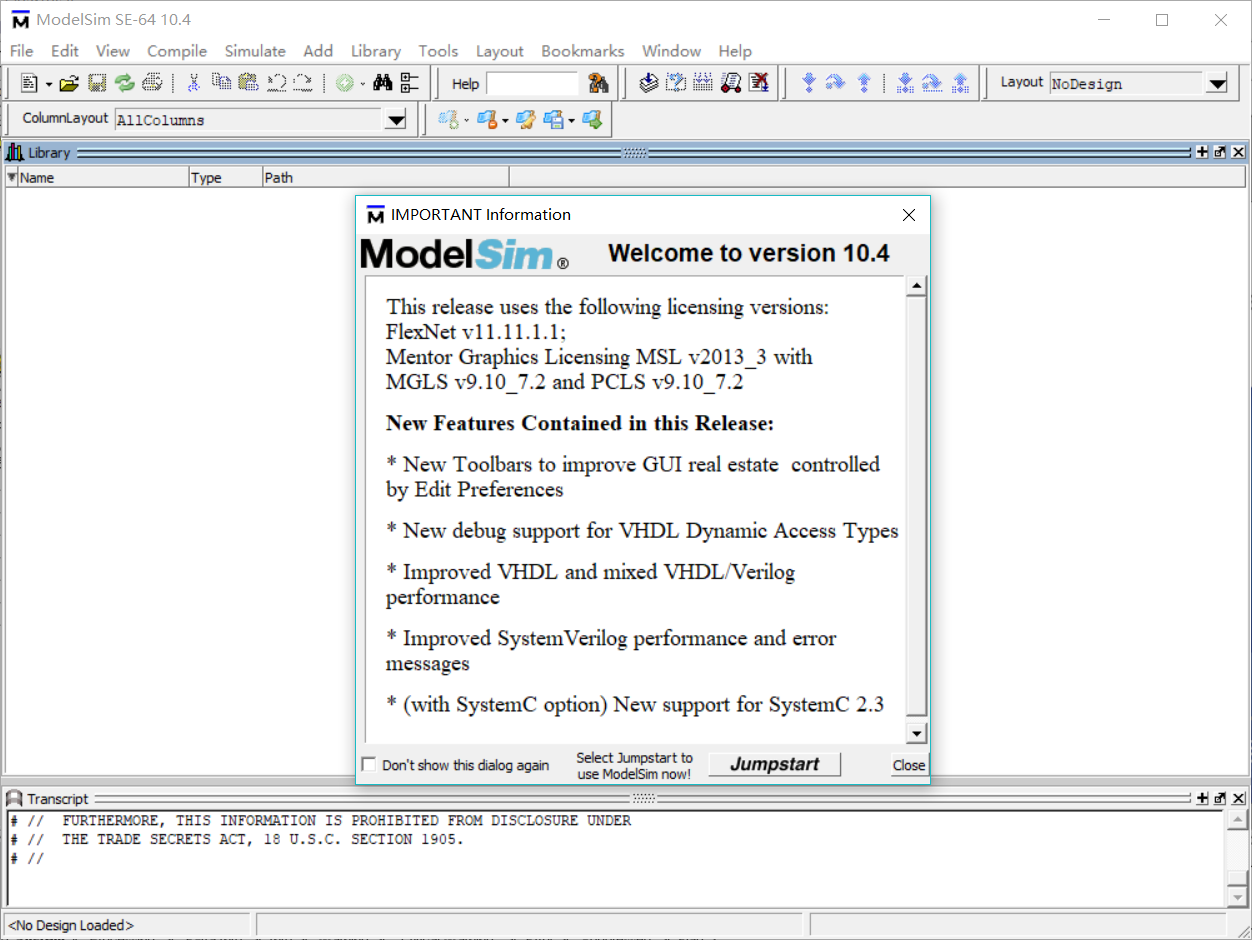
**题目2：安装软件**

软件界面截图

Quartus II



ModelSim



## hw2

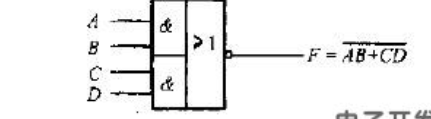
1、什么是层次化设计

2、为今天设计的AOI门写一个testbench，并用modelsim验证。

答：

1、层次电路原理图的设计理念是将实际的总体电路进行模块划分，划分的原则是每个电路模块都应该有明确的功能特征和相对独立的结构，而且，还要有简单、统一的接口，便于模块彼此之间的连接。

2、



**aoi.vhdl代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** aoi **is**

**port(**a**:in** std\_logic**;**

b**:in** std\_logic**;**

c**:in** std\_logic**;**

d**:in** std\_logic**;**

f**:out** std\_logic

**);**

**end** **entity;**

**architecture** behav **of** aoi **is**

**begin**

f**<=**not**((**a and b**)**or**(**c and d**));**

**end** behav**;**

**aoi.vht代码**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** aoi\_tb **IS**

**END** aoi\_tb**;**

**ARCHITECTURE** aoi\_arch **OF** aoi\_tb **IS**

-- constants

-- signals

**SIGNAL** a **:** STD\_LOGIC**;**

**SIGNAL** b **:** STD\_LOGIC**;**

**SIGNAL** c **:** STD\_LOGIC**;**

**SIGNAL** d **:** STD\_LOGIC**;**

**SIGNAL** f **:** STD\_LOGIC**;**

**COMPONENT** aoi

**PORT** **(**

a **:** **IN** STD\_LOGIC**;**

b **:** **IN** STD\_LOGIC**;**

c **:** **IN** STD\_LOGIC**;**

d **:** **IN** STD\_LOGIC**;**

f **:** **OUT** STD\_LOGIC

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** aoi

**PORT** **MAP** **(**

-- list connections between master ports and signals

a **=>** a**,**

b **=>** b**,**

c **=>** c**,**

d **=>** d**,**

f **=>** f

**);**

always **:** **PROCESS**

**BEGIN**

-- code executes for every event on sensitivity list

a**<=**'0'**;**

b**<=**'0'**;**

c**<=**'0'**;**

d**<=**'0'**;**

**for** i **in** 0 **to** 1 **loop**

**for** j **in** 0 **to** 1 **loop**

**for** k **in** 0 **to** 1 **loop**

**for** m **in** 0 **to** 1 **loop**

**wait** **for** 10 ns**;**

a**<=**not a**;**

**end** **loop;**

b**<=**not b**;**

**end** **loop;**

c**<=**not c**;**

**end** **loop;**

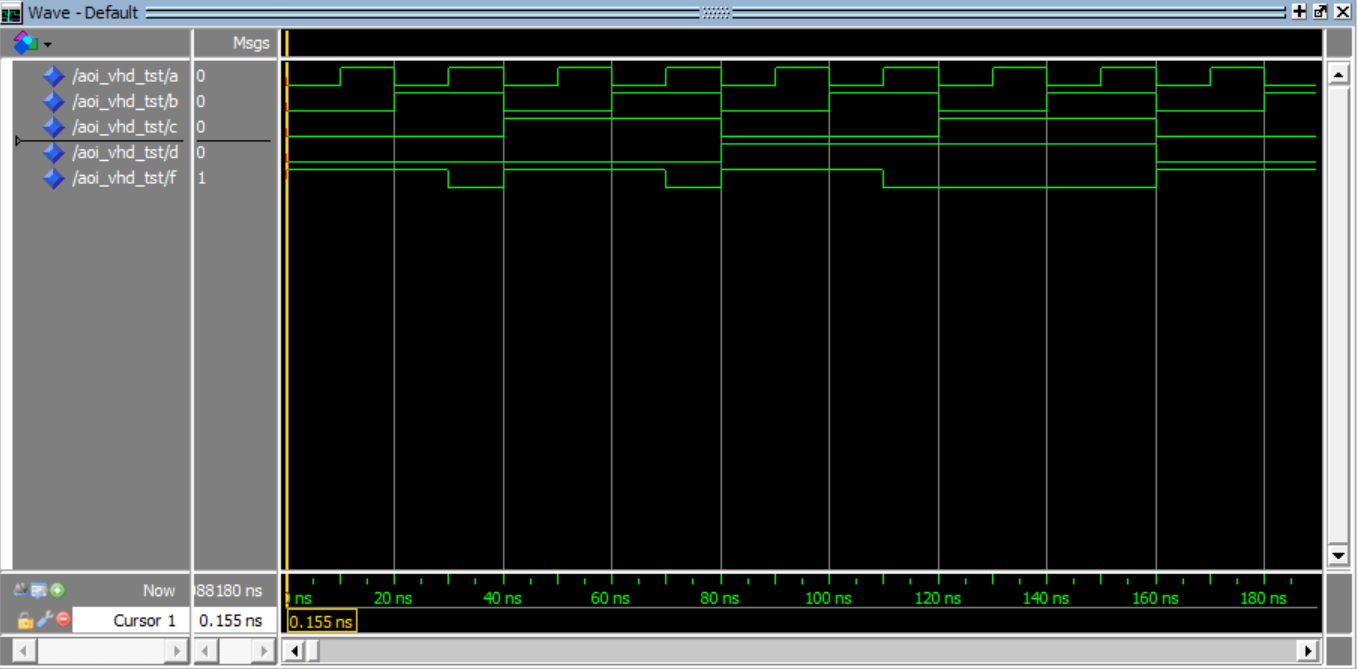
d**<=**not d**;**

**end** **loop;**

**END** **PROCESS** always**;**

**END** aoi\_arch**;**

**ModelSim仿真图**



## hw3

### 1、

Encoders are used to encode discrete data into a coded form and decoders are used to convert it back into its original undecoded form. An encoder that has 2(or less) input lines encodes input data to provide n encoded output lines. The truth table for an 8-3 binary encoder(8 inputs and 3 outputs) is shown in followed Table. It is assumed that only one input has a value of 1 at any given, otherwise the output has some undefined value and the circuit is meaningless. Please write the Verilog model of the given encoder and write the testbench to verify your model.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| inputs | | | | | | | | outputs | | |
| A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | Y2 | Y1 | Y0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** encoder83 **is**

**port(**a**:in** std\_logic\_vector**(**7 **downto** 0**);**

y**:out** std\_logic\_vector**(**2 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** encoder83 **is**

**begin**

**process(**a**)**

**begin**

**case** a **is**

**when** "00000001"**=>**y**<=**"000"**;**

**when** "00000010"**=>**y**<=**"001"**;**

**when** "00000100"**=>**y**<=**"010"**;**

**when** "00001000"**=>**y**<=**"011"**;**

**when** "00010000"**=>**y**<=**"100"**;**

**when** "00100000"**=>**y**<=**"101"**;**

**when** "01000000"**=>**y**<=**"110"**;**

**when** "10000000"**=>**y**<=**"111"**;**

**when** **others=>**y**<=**"XXX"**;**

**end** **case;**

**end** **process;**

**end** behav**;**

**Testbench代码**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** encoder83\_vhd\_tst **IS**

**END** encoder83\_vhd\_tst**;**

**ARCHITECTURE** encoder83\_arch **OF** encoder83\_vhd\_tst **IS**

-- constants

-- signals

**SIGNAL** a **:** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**);**

**SIGNAL** y **:** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

**COMPONENT** encoder83

**PORT** **(**

a **:** **IN** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**);**

y **:** **OUT** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**)**

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** encoder83

**PORT** **MAP** **(**

-- list connections between master ports and signals

a **=>** a**,**

y **=>** y

**);**

always **:** **PROCESS**

-- optional sensitivity list

-- ( )

-- variable declarations

**BEGIN**

-- code executes for every event on sensitivity list

a**<=**"00000001"**;**

**wait** **for** 10 ns**;**

a**<=**"00000010"**;**

**wait** **for** 10 ns**;**

a**<=**"00000100"**;**

**wait** **for** 10 ns**;**

a**<=**"00001000"**;**

**wait** **for** 10 ns**;**

a**<=**"00010000"**;**

**wait** **for** 10 ns**;**

a**<=**"00100000"**;**

**wait** **for** 10 ns**;**

a**<=**"01000000"**;**

**wait** **for** 10 ns**;**

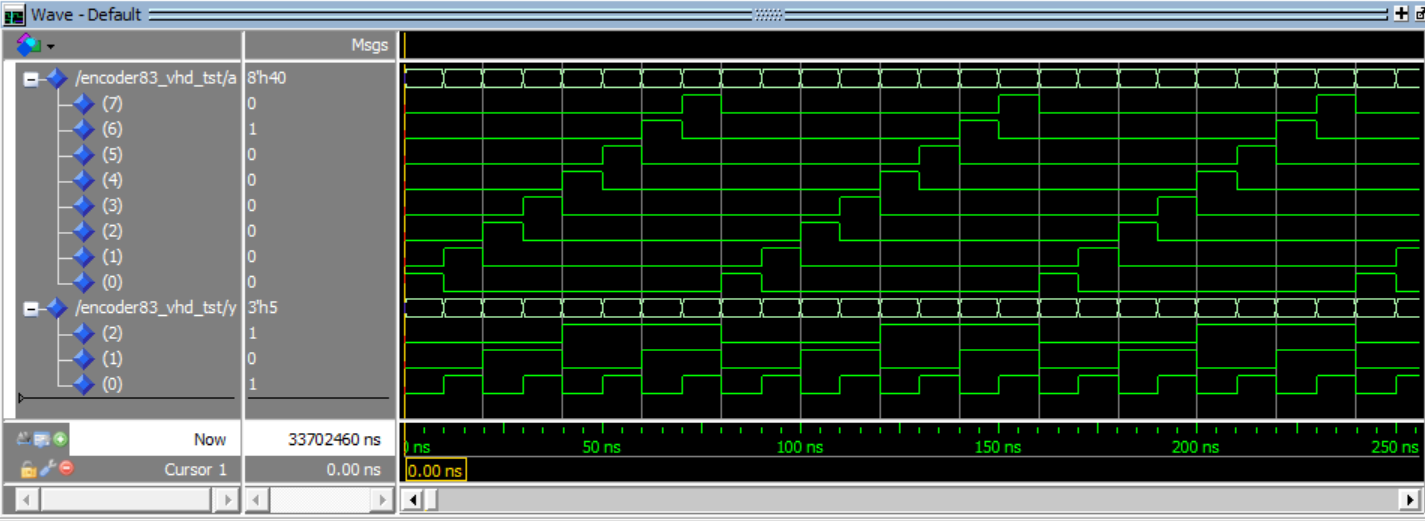
a**<=**"10000000"**;**

**wait** **for** 10 ns**;**

**END** **PROCESS** always**;**

**END** encoder83\_arch**;**

**运行截图**



### 2、

The truth table for a priority 8-3 binary encoder(8 inputs and 3 outputs) is shown in followed Table. Please write the verilog model of the given encoder and write the testbench to verify your model.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| inputs | | | | | | | | outputs | | | |
| A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | Y2 | Y1 | Y0 | VALID |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | X | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | X | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | X | X | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | X | X | X | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | 1 | X | X | X | X | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | X | X | X | X | X | 1 | 0 | 1 | 1 |
| 0 | 1 | X | X | X | X | X | X | 1 | 1 | 0 | 1 |
| 1 | X | X | X | X | X | X | X | 1 | 1 | 1 | 1 |

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** binaryencoder83 **is**

**port(**a**:in** std\_logic\_vector**(**7 **downto** 0**);**

y**:out** std\_logic\_vector**(**2 **downto** 0**);**

valid**:out** std\_logic**);**

**end** **entity;**

**architecture** behav **of** binaryencoder83 **is**

**begin**

**process(**a**)**

**begin**

**if** **(**a**(**7**)=**'1'**)then** y**<=**"111"**;**valid**<=**'0'**;**

**elsif(**a**(**6**)=**'1'**)then** y**<=**"110"**;**valid**<=**'1'**;**

**elsif(**a**(**5**)=**'1'**)then** y**<=**"101"**;**valid**<=**'1'**;**

**elsif(**a**(**4**)=**'1'**)then** y**<=**"100"**;**valid**<=**'1'**;**

**elsif(**a**(**3**)=**'1'**)then** y**<=**"011"**;**valid**<=**'1'**;**

**elsif(**a**(**2**)=**'1'**)then** y**<=**"010"**;**valid**<=**'1'**;**

**elsif(**a**(**1**)=**'1'**)then** y**<=**"001"**;**valid**<=**'1'**;**

**elsif(**a**(**0**)=**'1'**)then** y**<=**"000"**;**valid**<=**'1'**;**

**else** y**<=**"XXX"**;**valid**<=**'0'**;**

**end** **if;**

**end** **process;**

**end** behav**;**

**Testbench代码**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** binaryencoder83\_vhd\_tst **IS**

**END** binaryencoder83\_vhd\_tst**;**

**ARCHITECTURE** binaryencoder83\_arch **OF** binaryencoder83\_vhd\_tst **IS**

-- constants

-- signals

**SIGNAL** a **:** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**);**

**SIGNAL** valid **:** STD\_LOGIC**;**

**SIGNAL** y **:** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

**COMPONENT** binaryencoder83

**PORT** **(**

a **:** **IN** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**);**

valid **:** **OUT** STD\_LOGIC**;**

y **:** **OUT** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**)**

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** binaryencoder83

**PORT** **MAP** **(**

-- list connections between master ports and signals

a **=>** a**,**

valid **=>** valid**,**

y **=>** y

**);**

always **:** **PROCESS**

**BEGIN**

-- code executes for every event on sensitivity list

a**<=**"00000000"**;**

**wait** **for** 10 ns**;**

a**<=**"00000001"**;**

**wait** **for** 10 ns**;**

a**<=**"0000001X"**;**

**wait** **for** 10 ns**;**

a**<=**"000001XX"**;**

**wait** **for** 10 ns**;**

a**<=**"00001XXX"**;**

**wait** **for** 10 ns**;**

a**<=**"0001XXXX"**;**

**wait** **for** 10 ns**;**

a**<=**"001XXXXX"**;**

**wait** **for** 10 ns**;**

a**<=**"01XXXXXX"**;**

**wait** **for** 10 ns**;**

a**<=**"1XXXXXXX"**;**

**wait** **for** 10 ns**;**

**WAIT;**

**END** **PROCESS** always**;**

**END** binaryencoder83\_arch**;**

**运行截图：**



### 3、

The truth table for a priority 3-8 binary decoder(3 inputs and 8 outputs) is shown in followed Table. Please write the verilog model of the given encoder and write the testbench to verify your model.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| outputs | | | | | | | | inputs | | |
| Y7 | Y6 | Y5 | Y4 | Y3 | Y2 | Y1 | Y0 | A2 | A1 | A0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** decoder38 **is**

**port(**a**:in** std\_logic\_vector**(**2 **downto** 0**);**

y**:out** std\_logic\_vector**(**7 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** decoder38 **is**

**begin**

**process(**a**)**

**begin**

**if** **(**a**=**"000"**)then** y**<=**"00000001"**;**

**elsif(**a**=**"001"**)then** y**<=**"00000010"**;**

**elsif(**a**=**"010"**)then** y**<=**"00000100"**;**

**elsif(**a**=**"011"**)then** y**<=**"00001000"**;**

**elsif(**a**=**"100"**)then** y**<=**"00010000"**;**

**elsif(**a**=**"101"**)then** y**<=**"00100000"**;**

**elsif(**a**=**"110"**)then** y**<=**"01000000"**;**

**elsif(**a**=**"111"**)then** y**<=**"10000000"**;**

**else** y**<=**"XXXXXXXX"**;**

**end** **if;**

**end** **process;**

**end** behav**;**

**Testbench代码**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** decoder38\_vhd\_tst **IS**

**END** decoder38\_vhd\_tst**;**

**ARCHITECTURE** decoder38\_arch **OF** decoder38\_vhd\_tst **IS**

-- constants

-- signals

**SIGNAL** a **:** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

**SIGNAL** y **:** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**);**

**COMPONENT** decoder38

**PORT** **(**

a **:** **IN** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

y **:** **OUT** STD\_LOGIC\_VECTOR**(**7 **DOWNTO** 0**)**

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** decoder38

**PORT** **MAP** **(**

-- list connections between master ports and signals

a **=>** a**,**

y **=>** y

**);**

always **:** **PROCESS**

**BEGIN**

-- code executes for every event on sensitivity list

a**<=**"000"**;**

**wait** **for** 10 ns**;**

a**<=**"001"**;**

**wait** **for** 10 ns**;**

a**<=**"010"**;**

**wait** **for** 10 ns**;**

a**<=**"011"**;**

**wait** **for** 10 ns**;**

a**<=**"100"**;**

**wait** **for** 10 ns**;**

a**<=**"101"**;**

**wait** **for** 10 ns**;**

a**<=**"110"**;**

**wait** **for** 10 ns**;**

a**<=**"111"**;**

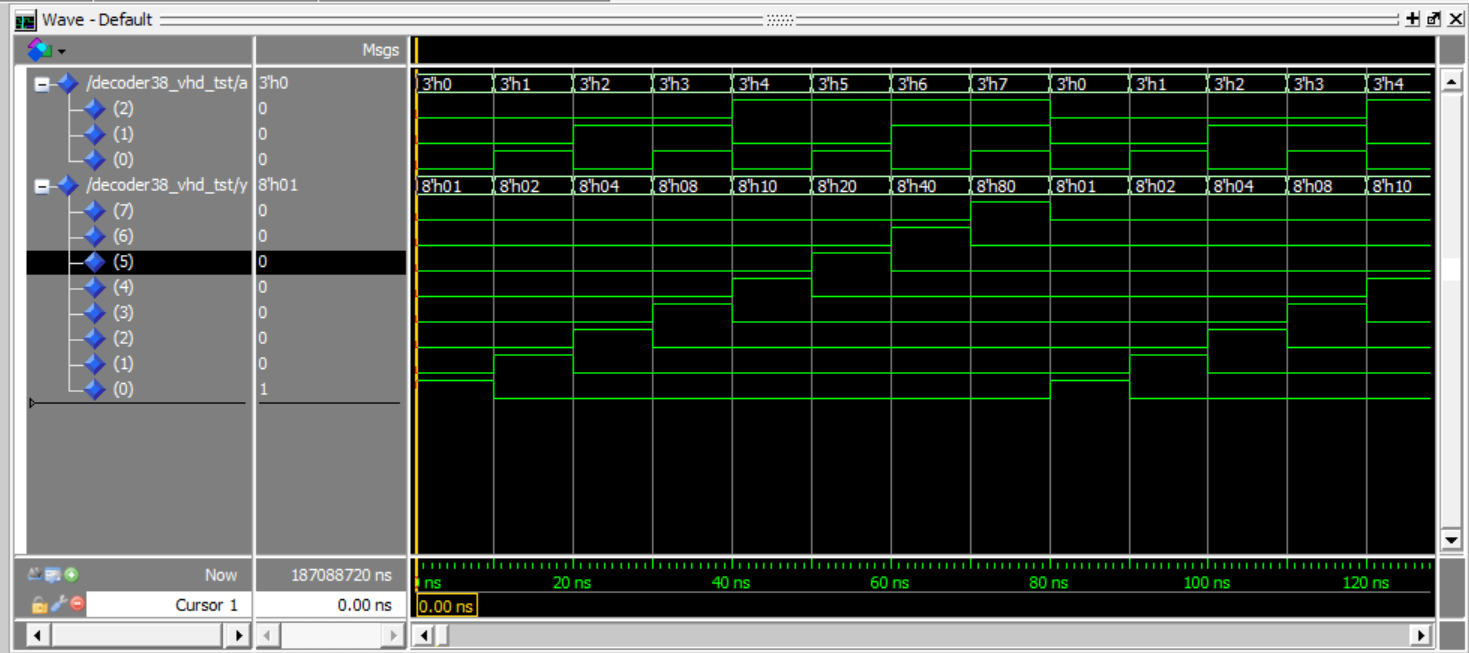
**wait** **for** 10 ns**;**

**WAIT;**

**END** **PROCESS** always**;**

**END** decoder38\_arch**;**

**运行截图**



### 4、

The truth table for a 3-6 binary decoder(3 inputs and 8 outputs) with enable is shown in followed Table. Please write the verilog model of the given encoder and write the testbench to verify your model.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| inputs | | | | | | | outputs | | | |
| Y5 | Y4 | Y3 | Y2 | Y1 | Y0 | A2 | | A1 | A0 | En |
| 0 | 0 | 0 | 0 | 0 | 0 | X | | X | X | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | | 1 | 0 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 0 | 1 | | 1 | 1 | 1 |

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** decoder36 **is**

**port(**en**:in** std\_logic**;**

a**:in** std\_logic\_vector**(**2 **downto** 0**);**

y**:out** std\_logic\_vector**(**5 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** decoder36 **is**

**begin**

**process(**a**,**en**)**

**begin**

**if(**en**=**'1'**)then**

**case** a **is**

**when** "000"**=>**y**<=**"000001"**;**

**when** "001"**=>**y**<=**"000010"**;**

**when** "010"**=>**y**<=**"000100"**;**

**when** "011"**=>**y**<=**"001000"**;**

**when** "100"**=>**y**<=**"010000"**;**

**when** "101"**=>**y**<=**"100000"**;**

**when** "110"**=>**y**<=**"000000"**;**

**when** "111"**=>**y**<=**"000000"**;**

**when** **others=>**y**<=**"000000"**;**

**end** **case;**

**else** y**<=**"XXXXXX"**;**

**end** **if;**

**end** **process;**

**end** behav**;**

**Testbench代码**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** decoder36\_vhd\_tst **IS**

**END** decoder36\_vhd\_tst**;**

**ARCHITECTURE** decoder36\_arch **OF** decoder36\_vhd\_tst **IS**

-- constants

-- signals

**SIGNAL** a **:** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

**SIGNAL** en **:** STD\_LOGIC**;**

**SIGNAL** y **:** STD\_LOGIC\_VECTOR**(**5 **DOWNTO** 0**);**

**COMPONENT** decoder36

**PORT** **(**

a **:** **IN** STD\_LOGIC\_VECTOR**(**2 **DOWNTO** 0**);**

en **:** **IN** STD\_LOGIC**;**

y **:** **OUT** STD\_LOGIC\_VECTOR**(**5 **DOWNTO** 0**)**

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** decoder36

**PORT** **MAP** **(**

-- list connections between master ports and signals

a **=>** a**,**

en **=>** en**,**

y **=>** y

**);**

always **:** **PROCESS**

**BEGIN**

-- code executes for every event on sensitivity list

a**<=**"XXX"**;**

**wait** **for** 10 ns**;**

a**<=**"000"**;**

**wait** **for** 10 ns**;**

a**<=**"001"**;**

**wait** **for** 10 ns**;**

a**<=**"010"**;**

**wait** **for** 10 ns**;**

a**<=**"011"**;**

**wait** **for** 10 ns**;**

a**<=**"100"**;**

**wait** **for** 10 ns**;**

a**<=**"101"**;**

**wait** **for** 10 ns**;**

a**<=**"110"**;**

**wait** **for** 10 ns**;**

a**<=**"111"**;**

**wait** **for** 10 ns**;**

**WAIT;**

**END** **PROCESS** always**;**

**END** decoder36\_arch**;**

**运行截图**



## hw4

clock division

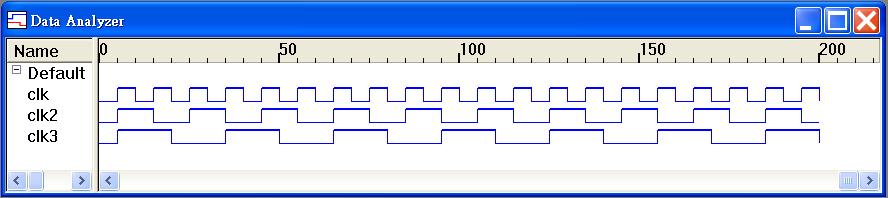
操作说明：

input: clock(clk)。

output: clock频率分别 除2(clk2) 及除3(clk3)。

练习目标：

熟悉dataflow modeling, behavioral modeling, timing control



testbench：

module Clock\_Divide\_tb;

reg clk;

wire clk2,clk3;

Clock\_Divide m1(clk,clk2,clk3);

initial clk=0;

always #5 clk=~clk;

initial #200 $finish;

endmodule

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** dividef **is**

**generic(**m**:**integer**:=**2**);**

**port(**clk**:in** std\_logic**;**

clk2**:out** std\_logic**;**

clk3**:out** std\_logic**);**

**end** **entity;**

**architecture** behav **of** dividef **is**

**signal** division2**:**std\_logic**:=**'0'**;**

**signal** division4**:**std\_logic**:=**'0'**;**

**signal** temp1**,**temp2**:**integer **range** 0 **to** 3**:=**0**;**

**signal** cnt**:**integer **range** 0 **to** m**-**1**;**

**begin**

--count

count**:process(**clk**)**

**begin**

**if** **(**clk'**event** and clk**=**'1'**)** **then**

**if(**cnt**<**m**-**1**)** **then**

cnt**<=** cnt**+**1**;**

**else**

cnt **<=** 0**;**

**end** **if;**

**end** **if;**

**end** **process** count**;**

--control\_clk2

control\_clk2**:process(**cnt**)**

**begin**

**if(**cnt**<**m**/**2**)then**

clk2**<=**'0'**;**

**else**

clk2**<=**'1'**;**

**end** **if;**

**end** **process** control\_clk2**;**

--count rising\_edge

p1**:process(**clk**)**

**begin**

**if** **rising\_edge(**clk**)** **then**

**if** temp1**=**0 **then**

division2**<=**'1'**;**

**elsif** temp1**=**1 **then**

division2**<=**'0'**;**

**end** **if;**

temp1**<=**temp1**+**1**;**

**if** temp1**=**2 **then**

temp1**<=**0**;**

**end** **if;**

**end** **if;**

**end** **process** p1**;**

--count falling\_edge

p2**:process(**clk**)**

**begin**

**if** clk'**event** and clk**=**'0' **then**

**if** temp2**=**0 **then**

division4**<=**'1'**;**

**elsif** temp2**=**1 **then**

division4**<=**'0'**;**

**end** **if;**

temp2**<=**temp2**+**1**;**

**if** temp2**=**2 **then**

temp2**<=**0**;**

**end** **if;**

**end** **if;**

**end** **process** p2**;**

--control\_clk3

control\_clk3**:process(**division2**,**division4**)**

**begin**

clk3**<=**division2 or division4**;**

**end** **process** control\_clk3**;**

**end** behav**;**

**testbench代码：**

**LIBRARY** ieee**;**

**USE** ieee**.**std\_logic\_1164**.all;**

**ENTITY** dividef\_vhd\_tst **IS**

**END** dividef\_vhd\_tst**;**

**ARCHITECTURE** dividef\_arch **OF** dividef\_vhd\_tst **IS**

**SIGNAL** clk **:** STD\_LOGIC**;**

**SIGNAL** clk2 **:** STD\_LOGIC**;**

**SIGNAL** clk3 **:** STD\_LOGIC**;**

**COMPONENT** dividef

**PORT** **(**

clk **:** **IN** STD\_LOGIC**;**

clk2 **:** **OUT** STD\_LOGIC**;**

clk3 **:** **OUT** STD\_LOGIC

**);**

**END** **COMPONENT;**

**BEGIN**

i1 **:** dividef

**PORT** **MAP** **(**

clk **=>** clk**,**

clk2 **=>** clk2**,**

clk3 **=>** clk3

**);**

always **:** **PROCESS**

**BEGIN**

**constant** clk\_period**:**time**:=**10 ns**;**

**begin**

clk**<=**'0'**;**

**wait** **for** clk\_period**/**2**;**

clk**<=**'1'**;**

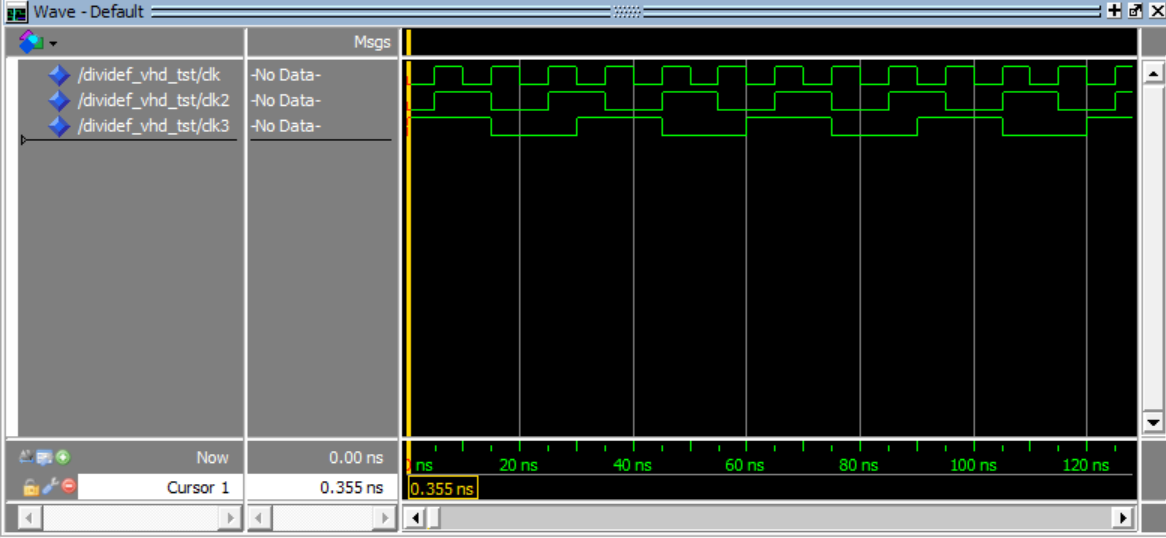
**wait** **for** clk\_period**/**2**;**

**WAIT;**

**END** **PROCESS** always**;**

**END** dividef\_arch**;**

**运行截图：**

****

## hw5

FSM. Write a behavioral Verilog module

description for a finite state machine with the

function of the state transition diagram shown to the

right.

The “reset” inputs forces the machine into S0 at the

rising-edge of the clock , and takes priority over IN.

**module FSM(clk, reset, IN, OUT);**

**input clk, reset, IN;**

**output OUT;**



**Vhdl代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** transtate **is**

**port(**clk**:in** std\_logic**;**

rst**:in** std\_logic**;**

input1**:in** std\_logic**;**

output1**:out** std\_logic**);**

**end** **entity;**

**architecture** behav **of** transtate **is**

**type** state\_type **is(**s0**,**s1**,**s2**);**

**signal** present\_state**,**next\_state**:**state\_type**;**

**begin**

--seg

seg**:process(**clk**,**rst**)**

**begin**

**if** rst**=**'1' **then** present\_state**<=**s0**;**

**elsif(rising\_edge(**clk**))** **then** present\_state**<=**next\_state**;**

**end** **if;**

**end** **process** seg**;**

--com

com**:process(**present\_state**,**input1**)**

**begin**

**case** present\_state **is**

**when** s0**=>**

**if** input1**=**'0' **then** next\_state**<=**s0**;**output1**<=**'1'**;**

**else** next\_state**<=**s2**;**output1**<=**'0'**;**

**end** **if;**

**when** s1**=>**

**if** input1**=**'0' **then** next\_state**<=**s1**;**output1**<=**'0'**;**

**else** next\_state**<=**s0**;**output1**<=**'1'**;**

**end** **if;**

**when** s2**=>**

**if** input1**=**'0' **then** next\_state**<=**s0**;**output1**<=**'0'**;**

**else** next\_state**<=**s1**;**output1**<=**'0'**;**

**end** **if;**

**end** **case;**

**end** **process** com**;**

**end** behav**;**

## hw6

### 1、

Design a combinational logic subsystem with three inputs, I3, I2, I1, and two outputs, O1, O0, that behaves as follows. The outputs indicate the highest index of the inputs that is driven high. For example, if I3 is 0, I2 is 1, I1 is 1, then O1, O0 would be 10 (i.e., I2 is the highest input set to 1). If none of the inputs is 1, then the outputs would be 00.

a. Specify the subsystem by filling out a complete truth table for the two outputs.

b. Write a specification of this function in Verilog.

c. Find the minimized Sum of Products description using K-maps.

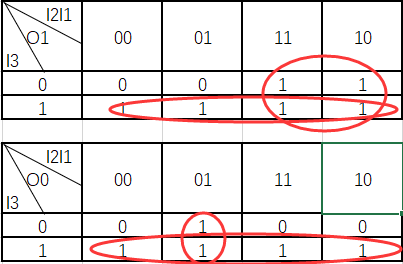
d. Implement the subsystem using 2 x 4:1 multiplexers.

**解**

**真值表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| I3 | I2 | I1 | O1 | O0 |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | X | 1 | 0 |
| 1 | X | X | 1 | 1 |

**卡诺图**



**得关系式：**

O1=I3+I2，O0=I3+I2’I1

**VHDL代码：**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** subsystem **is**

**port(**i3**,**i2**,**i1**:in** std\_logic**;**

o1**,**o0**:out** std\_logic**);**

**end** **entity;**

**architecture** behav **of** subsystem **is**

**begin**

o1**<=**i3 or i2**;**

o0**<=**i3 or **((**not i2**)**and i1**);**

**end** behav**;**

### 2、

Scientists have discovered that the Venusians use a base 16 number system. However, the digits are quite different than the ones to which we are accustomed. The first row below is 0 through 7, and the second row is 8 through F. Your task is to design a combinational logic subsystem to decode a hexadecimal digit in the range of 0 (0000) through F (1111) to drive a seven-segment display for the Martian version of the hexadecimal digits (0-7 in the top row, 8-F in the bottom row). The LED segments are numbered counter clockwise starting at the horizontal LED at the bottom (LED0). The middle LED is LED6.



a. Specify the function by filling out a complete truth table for each of the seven segment drivers.

b. Write each as a Verilog specification.

**VHDL代码**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** decoder47 **is**

**port(**input1**:in** std\_logic\_vector**(**3 **downto** 0**);**

output1**:out** std\_logic\_vector**(**6 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** decoder47 **is**

**begin**

**with** input1 **select**

output1**<=**"0000000" **when** "0000"**,**

"0100000" **when** "0001"**,**

"0110000" **when** "0010"**,**

"0111000" **when** "0011"**,**

"0111100" **when** "0100"**,**

"0111110" **when** "0101"**,**

"1111110" **when** "0110"**,**

"1111111" **when** "0111"**,**

"1111101" **when** "1000"**,**

"0111101" **when** "1001"**,**

"0111001" **when** "1010"**,**

"0110001" **when** "1011"**,**

"0100011" **when** "1100"**,**

"0000111" **when** "1101"**,**

"1000111" **when** "1110"**,**

"1100011" **when** "1111"**,**

**unaffected** **when** **others;**

**end** behav**;**

### 3、

Design a 3-bit counter that implements the following sequence: 000, 010, 100, 110, 111, 101, 011, 001, and repeat. Design the counter with a reset input that causes the counter to enter the 000 state.

a. Write a Verilog specification for this counter.

**VHDL代码：**

**library** ieee**;**

**use** ieee**.**std\_logic\_1164**.all;**

**entity** states8 **is**

**port(**clk**:in** std\_logic**;**

rst**:in** std\_logic**;**

counter**:out** std\_logic\_vector**(**2 **downto** 0**));**

**end** **entity;**

**architecture** behav **of** states8 **is**

**type** state\_type **is(**s0**,**s1**,**s2**,**s3**,**s4**,**s5**,**s6**,**s7**);**

**signal** present\_state**,**next\_state**:**state\_type**;**

**begin**

--seg

seg**:process(**clk**,**rst**)**

**begin**

**if** rst**=**'1' **then** present\_state**<=**s0**;**counter**<=**"000"**;**

**elsif(rising\_edge(**clk**))** **then** present\_state**<=**next\_state**;**

**end** **if;**

**end** **process** seg**;**

--com

com**:process(**present\_state**)**

**begin**

**case** present\_state **is**

**when** s0**=>**

next\_state**<=**s1**;**

counter**<=**"000"**;**

**when** s1**=>**

next\_state**<=**s2**;**

counter**<=**"010"**;**

**when** s2**=>**

next\_state**<=**s3**;**

counter**<=**"100"**;**

**when** s3**=>**

next\_state**<=**s4**;**

counter**<=**"110"**;**

**when** s4**=>**

next\_state**<=**s5**;**

counter**<=**"111"**;**

**when** s5**=>**

next\_state**<=**s6**;**

counter**<=**"101"**;**

**when** s6**=>**

next\_state**<=**s7**;**

counter**<=**"011"**;**

**when** s7**=>**

next\_state**<=**s0**;**

counter**<=**"001"**;**

**when** **others=>**

next\_state**<=**s0**;**

counter**<=**"000"**;**

**end** **case;**

**end** **process** com**;**

**end** behav**;**