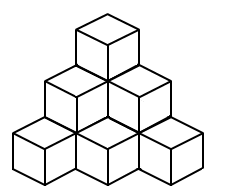
**数字IC部分（110分满分）**

1．**单选题**　|5分

把若干个大小相同的水立方摆成如图形状，从上向下数，摆1层有1个立方体，摆2层共有4个立方体，摆3层共有10个立方体，问摆7层共有多少个立方体：



|  |  |  |
| --- | --- | --- |
| **A、** | 60 |  |

|  |  |  |
| --- | --- | --- |
| **B、** | 64 |  |

|  |  |  |
| --- | --- | --- |
| **C、** | 80 |  |

|  |  |  |
| --- | --- | --- |
| **D、** | 84 |  |

2．**问答题**　|5分

已知有个10个药瓶（9个标准瓶和1个有问题瓶），里面装满了足够多的药片。其中一个有问题瓶子的药片的重量要比其他标准瓶子的重量要少一点（标准药瓶里面的药片重量是已知的，有问题瓶子里的药片每片都比正常药片少，并且少的重量是已知的，偏差不大）。 有个秤可称量重量，但是只能称重一次。 请设计一种方法，快速的把有问题的瓶子找到。

1. 画出2输入与非门的CMOS电路图（2分）和对应版图（3分）
2. 列出题1的真值表，注意输入包括0/1/x/z四种状态，并简要解释x z组合输出值的原因（5分）
3. 简要描述ASIC代码实现中 reset 的复位和释放的方式，并介绍这么做的原因（5分）
4. FIFO设计基础，根据FIFO的特性，设计地址位宽为4bit及深度16的FIFO，回答以下问题（共20分）：
   1. 指出普通二进制编码跟格雷码的区别，并指出哪个更适合FIFO编码(5分)
   2. 描述是否需要对gray code同步加SDC约束，若需要请描述如何加约束(5分)
   3. 请列出该深度16的 FIFO格雷码编码的值 (5分)：
   4. 如果FIFO深度由16调整为12，请列出深度12的FIFO的格雷码编码的值（5分）
5. 程序改错题：（15分）

module test (

input clk,

input [3:0] addr,

input ren,

input rst\_n,

input [7:0] wdata,

input wen,

output [7:0] data\_out,

)

reg [7:0] mem\_name [7:0];

always @ (posedge clk or negedge rst\_n)begin

if (rst\_n)

mem\_name = 0;

else if (wen)

mem\_name[addr] = wdata;

else

;

end

always @ (ren)

if (ren)

data\_out <= mem\_name[addr];

endmodule

1. 将下面C函数改写成Verilog/VHDL代码（20分）

int seqsum(int a, int b) {

    int y=0;

    for(int i=a; i<=b; i+=2)

        y=y+i;

return y;

}

1. 在实际工作中，我们有大量的规格书阅读，请阅读以下素材完成问题回答（30分）

简介：在RTL设计完成后，我们都会进行仿真验证功能，那么并行的RTL描述的硬件电路如何在串行调度的仿真器里面运行，请阅读以下手册完成问题作答：

-------------------------------------------------------------------------------------------------------------------------------------

**Execution of a model**

The balance of the clauses of this standard describe the behavior of each of the elements of the language. This clause gives an overview of the interactions between these elements, especially with respect to the scheduling and execution of events.

The elements that make up the Verilog HDL can be used to describe the behavior, at varying levels of abstraction, of electronic hardware. An HDL has to be a parallel programming language. The execution of certain language constructs is defined by parallel execution of blocks or processes. It is important to understand what execution order is guaranteed to the user and what execution order is indeterminate.

Although the Verilog HDL is used for more than simulation, the semantics of the language are defined for simulation, and everything else is abstracted from this base definition.

**Event simulation**

The Verilog HDL is defined in terms of a discrete event execution model. The discrete event simulation is described in more detail in this subclause to provide a context to describe the meaning and valid interpretation of Verilog HDL constructs. These resulting definitions provide the standard Verilog reference model for simulation, which all compliant simulators shall implement. However, there is a great deal of choice in the definitions that follow, and differences in some details of execution are to be expected between different simulators. In addition, Verilog HDL simulators are free to use different algorithms from those described in this clause, provided the user-visible effect is consistent with the reference model.

A design consists of connected threads of execution or processes. Processes are objects that can be evaluated, that may have state, and that can respond to changes on their inputs to produce outputs. Processes include primitives, modules, initial and always procedural blocks, continuous assignments, asynchronous tasks, and procedural assignment statements.

Every change in value of a net or variable in the circuit being simulated, as well as the named event, is considered **an *update event*.**

Processes are sensitive to update events. When an update event is executed, all the processes that are sensitive to that event are evaluated in an arbitrary order. The evaluation of a process is also an event, known as **an *evaluation event*.**

In addition to events, another key aspect of a simulator is time. The term ***simulation time*** is used to refer to the time value maintained by the simulator to model the actual time it would take for the circuit being simulated. The term ***time***is used interchangeably with simulation time in this clause.

Events can occur at different times. In order to keep track of the events and to make sure they are processed

in the correct order, the events are kept on an ***event queue***, ordered by simulation time. Putting an event on

the queue is called ***scheduling an event*.**

**The stratified event queue**

The Verilog event queue is logically segmented into five different regions. Events are added to any of the five regions, but are only removed from the active region.

a) ***Active events*** occur at the current simulation time and can be processed in any order.

b) ***Inactive* events** occur at the current simulation time, but shall be processed after all the active events are processed.

c) ***Nonblocking assign update* events** have been evaluated during some previous simulation time, but shall be assigned at this simulation time after all the active and inactive events are processed.

d) ***Monito****r* **events** shall be processed after all the active, inactive, and nonblocking assign update events are processed.

e) ***Future* events** occur at some future simulation time. Future events are divided into ***future inactive events***and ***future nonblocking assignment update events*.**

The processing of all the active events is called a *simulation cycle*.

The freedom to choose any active event for immediate processing is an essential source of nondeterminism in the Verilog HDL.

An *explicit zero delay* (#0) requires that the process be suspended and added as an inactive event for the current time so that the process is resumed in the next simulation cycle in the current time.

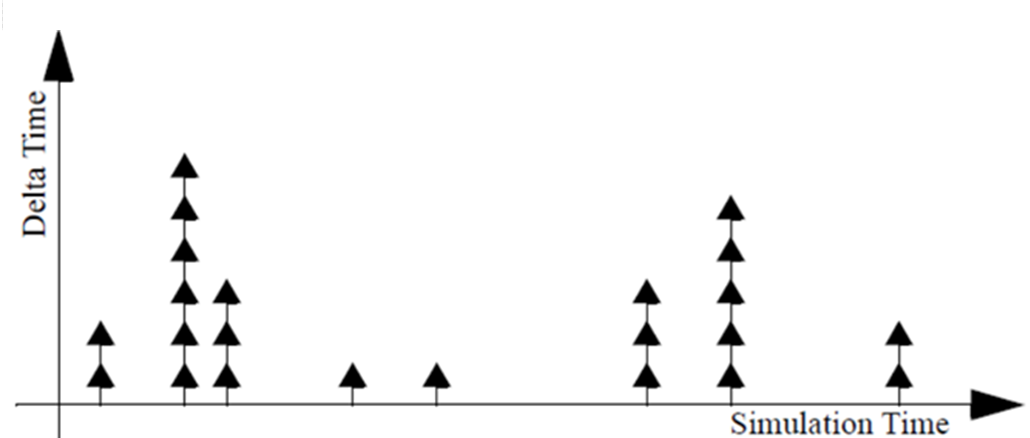
A nonblocking assignment creates a nonblocking assign update event, scheduled for a current or later simulation time.

The **$monitor** and **$strobe** system tasks create monitor events for their arguments. These events are continuously re-enabled in every successive time step. The monitor events are unique in that they cannot create any other events.

The callback procedures scheduled with PLI routines such as vpi\_register\_cb (cbReadWriteSynch) shall be treated as inactive events.

-------------------------------------------------------------------------------------------------------------------------------------

**问题1：**请结合以上SPEC和下图，回答以下问题（15分）



1. 每个黑色箭头表示的意义是什么 (2分)
2. 部分时刻点箭头数量有多有少，请描述原因 (3分)
3. 结合delta cycle请描述仿真时间如何推进的 (5分)
4. 4.请描述scheduler在仿真中扮演的作用以及规定scheduler的意义是什么 (5分)

**问题2：**根据以上描述完成Verilog simulation reference model填空，把对应A~n填入while代码内各X1~x11处（15分）：

A:there are nonblocking assign update events

B:there are events

C:no active events

D:there are monitor events

E:there are inactive events

g:update the modified object; add evaluation events for sensitive processes to event queue;

h:activate all monitor events

i:activate all inactive events

m:advance T to the next event time; activate all inactive events for time T;

k:activate all nonblocking assign update events

n:evaluate the process; add update events to the event queue;

while (X1) {

if (X2) {

if (X3) {

x6

} else if (X4) {

x7

} else if (X5) {

x8

} else {

x9

}

}

E = any active event;

if (E is an update event) {

x10

} else { /\* shall be an evaluation event \*/

x11

}

}

Notes：In all the examples that follow, **T** refers to the current simulation time, and all events are held in the event queue, ordered by simulation time.

**请直接在答题纸上在X（）内填入认为正确的A~E ，x（）内填入认为正确的g~n：**

X1 （）

X3 （）

X4 （）

X5 （）

x6 （）

x7 （）

x8 （）

x9 （）

x10 （）

x11 （）