软件技术说明文档

一、 软件名称

数字电路计算软件

二、 开发环境

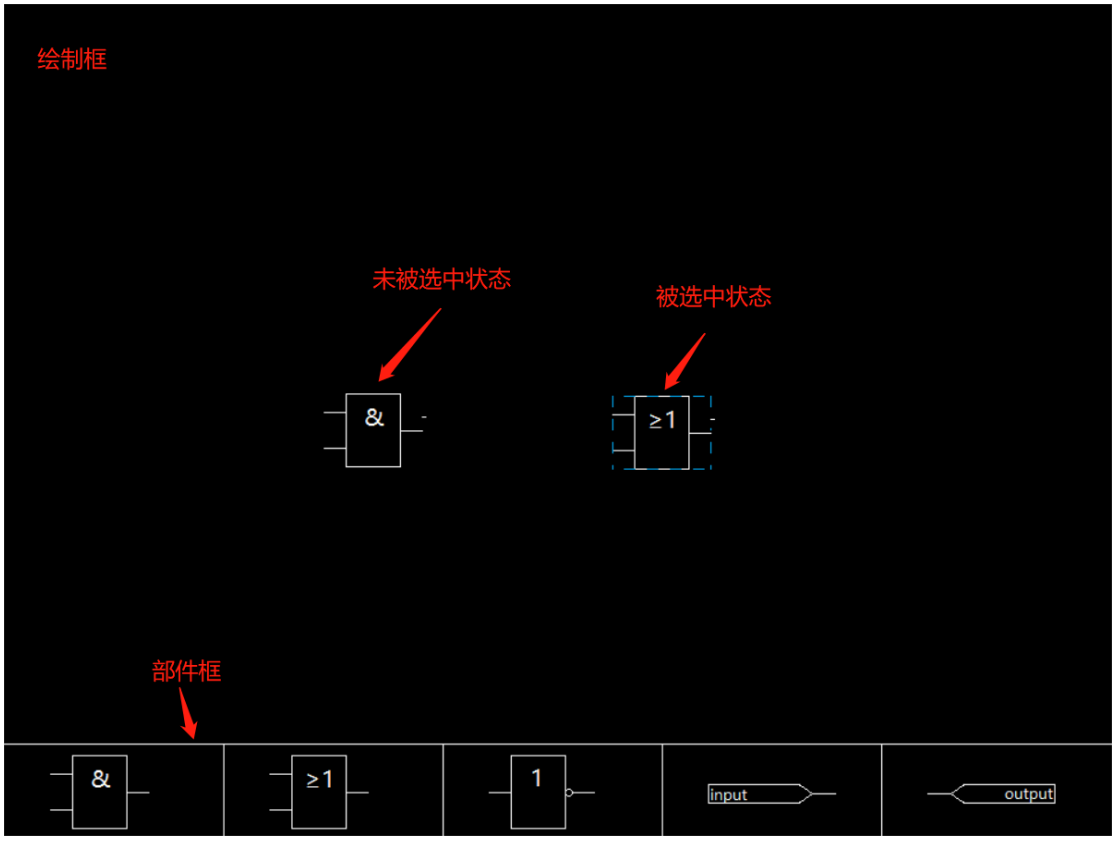
Microsoft Visual Studio 2019

EasyX Graphics Library

三、 功能设计

1. 系统中，用不同符号表示与、或、非门、输入端和输出端，每个部件有输入引脚、输出引脚
2. 整个界面分为组件框和绘制框，用户可以在组件框中将数字逻辑电路部件用鼠标拖拽到绘制框中，以新建一个部件
3. 用户可以在绘制框上，通过鼠标拖动，随意更改数字逻辑电路部件的位置
4. 用户可以随意添加和删除任意一个部件
5. 用户可以在系统中，把各输入端、与门、或门、非门、输出端用连线进行引脚的连接
6. 用户可在系统中设置每个输入端的 0 或 1 的输入值
7. 系统检查用户的连接有无错误，若连接无误则计算并显示该部件的输出值
8. 系统计算并显示输出端的信号结果（0 或 1）

四、 界面设计

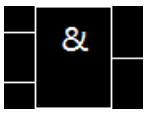


输入端

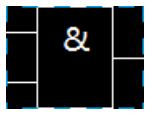
被选中的输入端

输出端

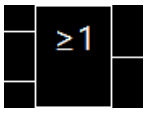
被选中的输出端



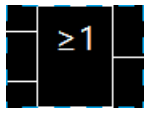
与门



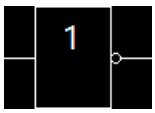
被选中的与门



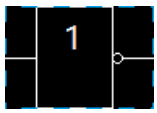
或门



被选中的或门

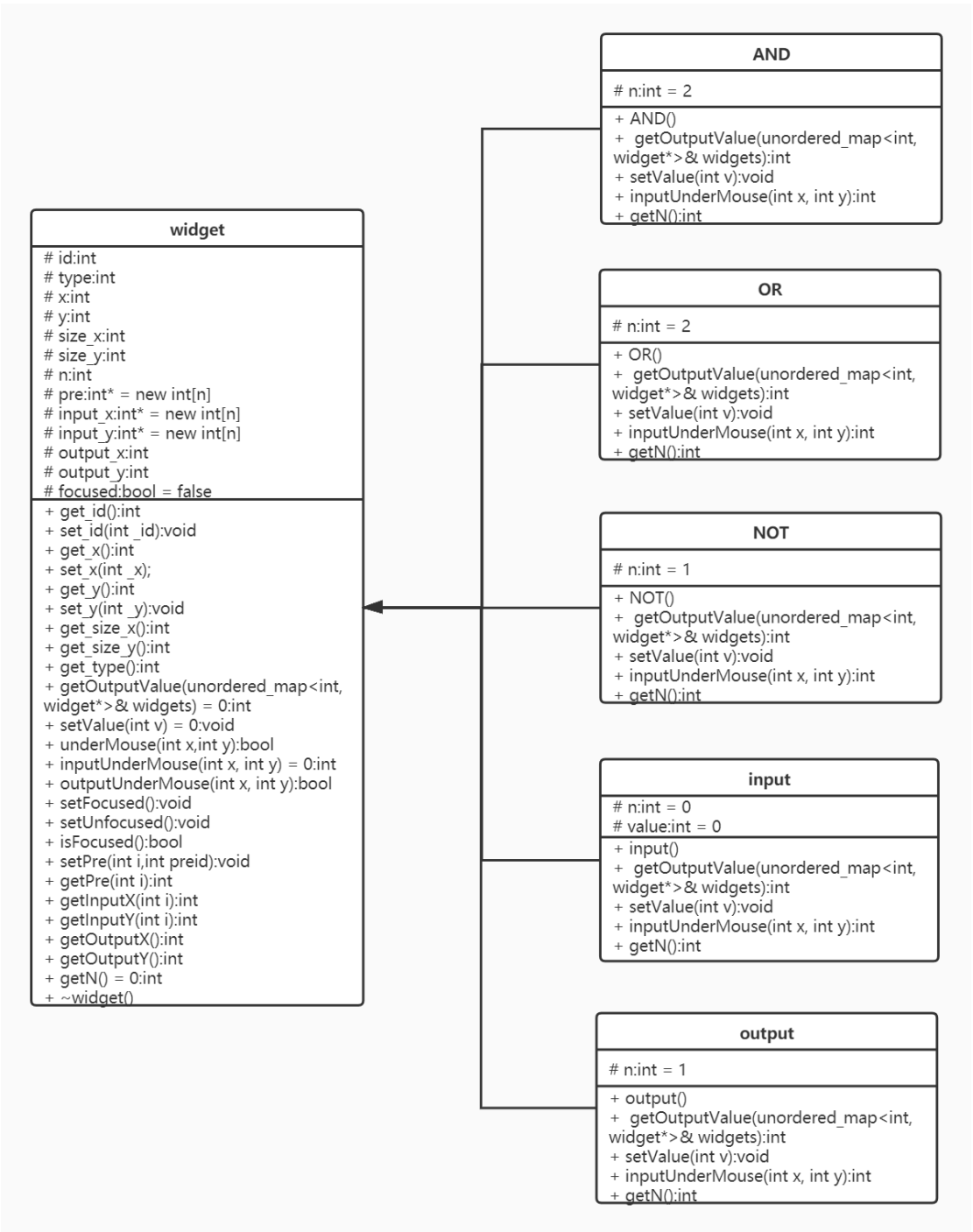


非门



被选中的非门

五、 类的设计



widget 类

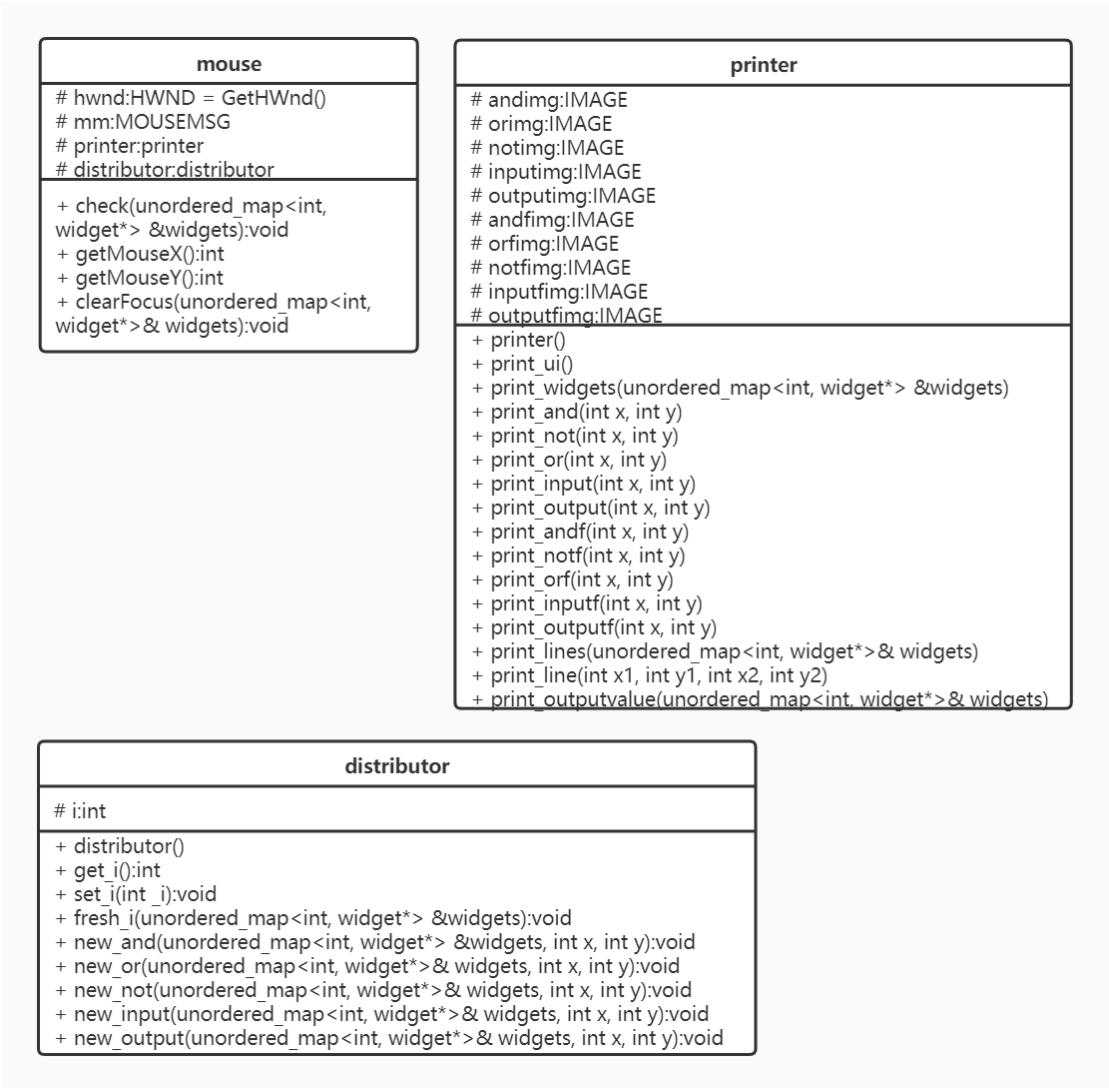
```
int id;//编号
int type;//组件类型 1:与门 2:或门 3:非门 4:输入端 5:输出端
int x;//位置横坐标
int y;//位置纵坐标
int size_x;//宽度
int size_y;//高度
int n;//输入管脚个数
int* pre = new int[n];//存放与输入端相连的组件的id
int* input_x = new int[n];//输入管脚在组件中的横坐标
int* input_y = new int[n];//输入管脚在组件中的纵坐标
int output_x;//输出管脚在组件中的横坐标
int output_y;//输出管脚在组件中的纵坐标
bool focused = false;//被选中的标记

int get_id();
void set_id(int _id);
int get_x();
void set_x(int _x);
int get_y();
void set_y(int _y);
int get_size_x();
int get_size_y();
int get_type();
virtual int getOutputValue(unordered_map<int, widget*>& widgets) = 0;//获取输出值
virtual void setValue(int v) = 0;//设置输入端的值
bool underMouse(int x,int y);//是否在鼠标下
virtual int inputUnderMouse(int x, int y) = 0;//输入管脚是否在鼠标下
bool outputUnderMouse(int x, int y);//输出管脚是否在鼠标下
void setFocused();//设为被选中
void setUnfocused();//设为未被选中
bool isFocused();//是否为选中状态
void setPre(int i,int preid);//记录与第i个输入管脚相连的组件的编号
int getPre(int i);//获取与第i个输入管脚相连的组件的编号
int getInputX(int i);//获取第i个输入管脚在窗口中的横坐标
int getInputY(int i);//获取第i个输入管脚在窗口中的纵坐标
int getOutputX();//获取输出管脚在窗口中的横坐标
int getOutputY();//获取输入管脚在窗口中的纵坐标
virtual int getN() = 0;
~widget();
```

widget 类为一个抽象类，此类是 AND 类、OR 类、NOT 类、input 类、output 类的父类，此类保存了部件的长、宽、位置、相连部件编号等属性，拥有计算输出值、检查被鼠标点击等函数。

AND 类、OR 类、NOT 类、input 类、output 类

继承了 widget 类并实现了 widget 类的抽象函数，分别表示与门、或门、非门、输入端、输出端。



mouse 类

```
class mouse//鼠标事件处理器
{
protected:
    HWND hwnd = GetHWnd();
    MOUSEMSG mm;
    printer printer;
    distributor distributor;//分配器，负责创建组件和分配编号
public:
    void check(unordered_map<int, widget*> &widgets);//检查鼠标事件
    int getMouseX();//获取鼠标在窗口中的横坐标
    int getMouseY();//获取鼠标在窗口中的纵坐标
    void clearFocus(unordered_map<int, widget*>& widgets);//清除对所有组件的选中
};
```

拥有获取鼠标在窗口中位置的函数 getMouseX()和 getMouseY()、检测并处理各项鼠标事件的函数 check()。

printer 类

```
class printer//绘图器
{
protected:
    IMAGE andimg, orimg, notimg, inputimg, outputimg, andfimg, orfimg, notfimg, inputfimg, outputfimg;
public:
    printer();
    void print_ui();//绘制组件框
    void print_widgets(unordered_map<int, widget*> &widgets);//绘制组件
    void print_and(int x, int y);//绘制与门
    void print_not(int x, int y);//绘制非门
    void print_or(int x, int y);//绘制或门
    void print_input(int x, int y);//绘制输入端
    void print_output(int x, int y);//绘制输出端
    void print_andf(int x, int y);//绘制被选中状态的与门
    void print_notf(int x, int y);//绘制被选中状态的非门
    void print_orf(int x, int y);//绘制被选中状态的或门
    void print_inputf(int x, int y);//绘制被选中状态的输入端
    void print_outputf(int x, int y);//绘制被选中状态的输出端
    void print_lines(unordered_map<int, widget*>& widgets);//绘制所有连线
    void print_line(int x1, int y1, int x2, int y2);//绘制两个组件间的一条连线
    void print_outputvalue(unordered_map<int, widget*>& widgets);//绘制每个组件的输出值
};
```

此类的对象负责进行各项绘图操作，构造函数负责读取文件中的图像。print_ui()用于绘制 ui 界面和组件框，print_widgets()用于绘制绘图框中的所有部件，print_lines()用于绘制组件间的连线。

distributor 类

```
class distributor//分配器
{
private:
    int i;
public:
    distributor();
    int get_i();
    void set_i(int _i);
    void fresh_i(unordered_map<int, widget*> &widgets);//刷新i，将i更新为比i大且未被使用的最小i
    void new_and(unordered_map<int, widget*> &widgets, int x, int y);//创建与门
    void new_or(unordered_map<int, widget*>& widgets, int x, int y);//创建或门
    void new_not(unordered_map<int, widget*>& widgets, int x, int y);//创建非门
    void new_input(unordered_map<int, widget*>& widgets, int x, int y);//创建输入端
    void new_output(unordered_map<int, widget*>& widgets, int x, int y);//创建输出端
};
```

distributor 类的对象负责生成新的 widget 类对象并保存到 C++标准库提供的 unordered_map 的一个实例 widgets 中，其中键为部件的编号，值为 widget 类

的对象的引用。此外，该类还用于实现对 id 编号的更新和回收。

六、 功能实现

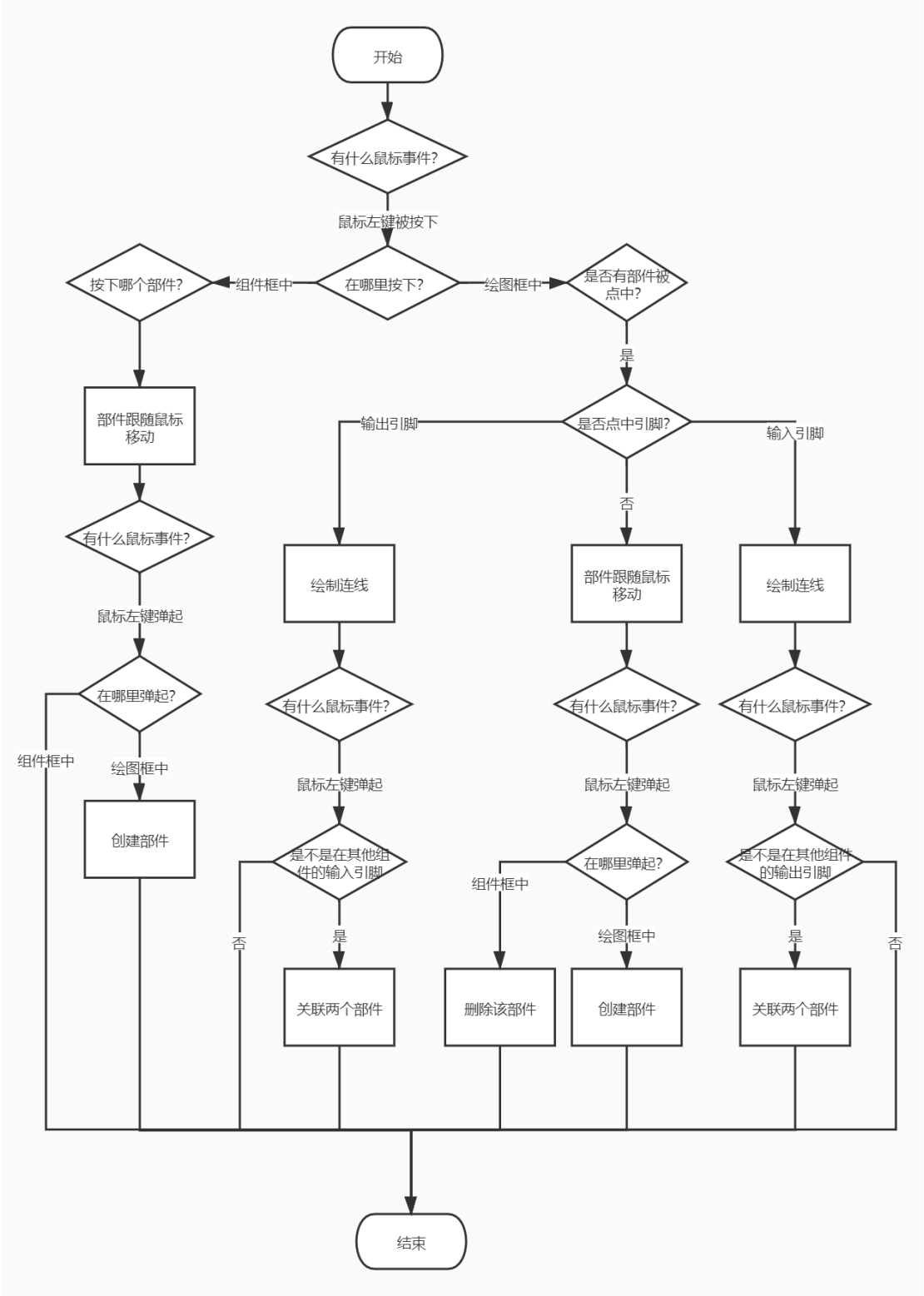
1. 使用 C++的 easyX 库实现绘图和鼠标事件监测功能，数字电路计算等核心功能均仅使用 C++实现。
2. 图像的显示

   使用 Windows 画图工具绘制与或非门、输入端、输出端及其被选中状态的图像，并在绘图器 printer 创建时，使用 easyX 提供的 loadimage()函数从文件中读入图像，保存在 printer 对象的 10 个 IMAGE 对象中，并在绘图器 printer 调用 print_ui()和 print_widgets()函数绘图时调用 easyX 库中的 putimage()函数直接使用 IMAGE 对象绘制图像。

   绘图器 printer 的 print_lines()函数会遍历所有部件，若该部件的输入引脚与其他部件的输出引脚相连，则绘制一条连线，为了使得连线符合数字电路绘图要求，编写了 print_line()函数进行美化。

   此外，绘图器 printer 的 print_outputvalue ()函数会遍历所有部件，若该部件的输出引脚有正确的输出值（即 0 或 1），则在输出引脚上方显示该值，否则显示“-”号。
3. 部件的拖动和连线

   部件的拖动和连线需要监测鼠标事件，所有鼠标事件均由 mouse 对象的 check()调用 easyX 的库函数来检测，并由 check()来处理，具体流程图如下：



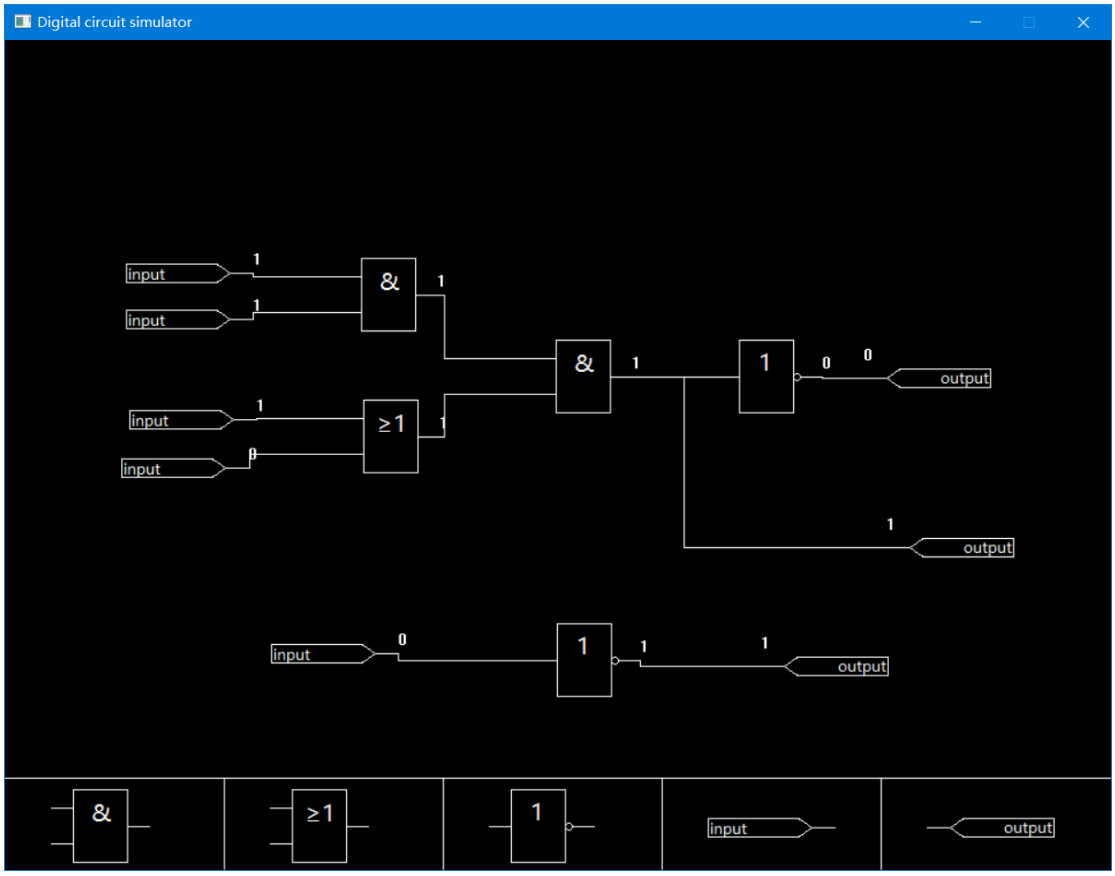
部件的拖动、连线和输入端数值的修改均由该函数控制。

4. widget 类对象的创建、储存和删除

所有 widget 类对象都保存在 C++标准库提供的 unordered_map 的一个实

例 widgets 中，其中键为部件的编号 id，值为 widget 类的对象的引用。当鼠标事件处理器 mouse 调用分配器 contributor 的创建部件的函数时，分配器 contributor 会为新对象分配内存空间和 id，并将该新对象的引用储存到 widgets 当中，随后会立即调用 fresh_i()函数以保证分配器的成员对象 i 为未被使用的最小 id 值。当一个 widget 类对象被删除后，占用的 id 被释放，这时鼠标事件处理器 mouse 会调用分配器 contributor 的 set_i()以保证分配器的成员对象 i 为未被使用的最小 id 值。

七、 测试



经过测试，所有需求均成功实现。

八、 不足与改进方向

1、 绘制框的大小有限，无法绘制复杂的电路图，后续可以扩大可操作的范围并

添加缩放功能，使得软件能绘制更加复杂的电路图。

2、 目前只支持一次选择一个部件，然后对其进行移动，而不能进行部件的多选。后续可加入拖拽鼠标进行部件多选，使用键盘按键进行复制、粘贴、删除的功能。

3、 CPU 占用率高，性能有待优化。本软件需要循环进行鼠标事件的检测，在软件运行时 CPU 占用率较高，有待优化。