



(12) 发明专利申请

(10) 申请公布号 CN 103677763 A

(43) 申请公布日 2014. 03. 26

(21) 申请号 201210316959. 7

(22) 申请日 2012. 08. 30

(71) 申请人 中国科学院软件研究所  
地址 100190 北京市海淀区中关村南四街 4 号

(72) 发明人 李明树 乔颖 翁彦 梁国政  
王朝辉

(74) 专利代理机构 北京君尚知识产权代理事务  
所(普通合伙) 11200  
代理人 余长江

(51) Int. Cl.  
G06F 9/44(2006. 01)

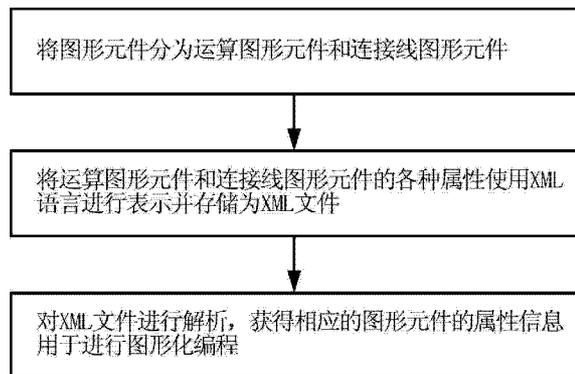
权利要求书1页 说明书9页 附图2页

(54) 发明名称

一种图形化编程的源文件存储及解析方法

(57) 摘要

本发明提供一种图形化编程的源文件存储及解析方法,其步骤包括:将图形化编程所需的图形元素分为运算图形元素和连接线图形元素;将所述运算图形元素和所述连接线图形元素的各种属性使用 XML 语言进行表示,将每种图形元素各自存储为一个 XML 文件;对所述 XML 文件进行解析并获得相应的图形元素的属性信息,用于进行图形化编程。本发明在图形化编程的源文件存储及转换的过程中使用 XML 技术,可以方便地实现图形化程序与 XML 描述之间的相互转换,使编程工作更高效、更方便,并缩短产品开发周期。



1. 一种图形化编程的源文件存储及解析方法,其步骤包括:
  - 1) 将图形化编程所需的图形元件分为运算图形元件和连接线图形元件;
  - 2) 将所述运算图形元件和所述连接线图形元件的各种属性使用 XML 语言进行表示,将每种图形元件各自存储为一个 XML 文件;
  - 3) 对所述 XML 文件进行解析并获得相应的图形元件的属性信息,用于进行图形化编程。
2. 如权利要求 1 所述的方法,其特征在于:所述运算图形元件包括算术运算图形元件和逻辑运算图形元件。
3. 如权利要求 1 所述的方法,其特征在于:所述运算图形元件的属性包括:输入端口属性、输出端口属性以及图形显示属性;所述图形显示属性包括:名称、高度、宽度和颜色。
4. 如权利要求 1 所述的方法,其特征在于:所述连接线图形元件的属性包括:名称属性、起始端点属性、终点端点属性、拐点位置坐标属性。
5. 如权利要求 1 所述的方法,其特征在于:使用开发工具 Altova XMLSpy 或者 Visual Studio 创建所述 XML 文件。
6. 如权利要求 1 所述的方法,其特征在于:使用 C、C++ 或 C# 语言实现所述解析,将图形元件属性解析到自定义的各种变量之中,用于后续编程。

## 一种图形化编程的源文件存储及解析方法

### 技术领域

[0001] 本发明属于计算机图形存储显示技术领域,具体涉及一种图形化编程的源文件存储及解析方法。

### 背景技术

[0002] 随着信息时代的飞速发展,社会对软件的需求量快速增长,因此制作软件的效率也需要不断提高。对于软件制作人员来说,使用高效的编程技术能够最大限度地节省人力、物力,降低成本,缩短产品开发周期,抢占市场先机。

[0003] 软件编程技术发展迅速,其重要组成部分—软件开发语言,由古老的机器码,经历了 BASIC、C、C++ 等,发展为高级语言;软件的开发环境也由文本式的代码编写,发展为可视化的多种工具集成的开发环境。作为编程技术的一种,图形化编程技术正走在前沿,是集可视化、对象化、组件化等思想为一体的集成开发环境。运用图形化的编程工具,编程人员可以像制作流程图一样,通过简单地添加和拖动几个图形化编程元素,连接必要的输入输出端,便可生成所需要的程序。图形化编程技术是软件编程技术发展过程中的一个飞跃,引领着新一代编程技术的发展。现有图形化编程技术中,所需图形元件一般为图片格式(如 png、jpg 文件等),每个图片文件代表具有不同属性的图形元件,这种开发模式工作量大、周期较长、功能有限、后期维护困难。

[0004] XML 语言因其具备的多种特性,正得到越来越广泛的应用。XML 语言与图形化编程技术的融合能体现出各自的优点,从而发挥出更大的作用。如果能将 XML 语言运用于图形化编程的源文件存储及转换,可使开发人员从繁复的代码中解脱出来,使编程工作更高效、更方便。

### 发明内容

[0005] 本发明的目的是提供一种图形化编程的源文件存储及解析方法,在图形化编程的源文件存储及解析(转换)的过程中使用 XML 技术,可以方便地实现图形化程序与 XML 描述之间的相互转换,使编程工作更高效、更方便,并缩短产品开发周期。

[0006] 本发明将图形化编程的所需图形元件的各种属性,如图形元件名称、宽度、高度、颜色、输入端口个数、输出端口个数、输入类型、输出类型等,以 XML 文件的形式存储;在图形化编程平台使用图形元件编程时,将存储该图形元件的 XML 文件进行解析,获得该图形元件的属性信息。具体来说,本发明采用如下技术方案:

[0007] 一种图形化编程的源文件存储及解析方法,其步骤包括:

[0008] 1) 将图形化编程所需的图形元件分为运算图形元件和连接线图形元件;

[0009] 2) 将所述运算图形元件和所述连接线图形元件的各种属性使用 XML 语言进行表示,将每种图形元件各自存储为一个 XML 文件;

[0010] 3) 对所述 XML 文件进行解析并获得相应的图形元件的属性信息,用于进行图形化编程。

[0011] 进一步地,所述运算图形元件包括算术运算图形元件和逻辑运算图形元件。

[0012] 进一步地,所述运算图形元件的属性包括:输入端口属性、输出端口属性以及图形显示属性;所述图形显示属性包括:名称、高度、宽度和颜色。

[0013] 进一步地,所述连接线图形元件的属性包括:名称属性、起始端点属性、终点端点属性、拐点位置坐标属性。

[0014] 进一步地,使用 C、C++ 或 C# 语言实现所述解析,将图形元件属性解析到自定义的各种变量之中,用于后续编程。

[0015] 在图形化程序中,图形化编程元素表现能力突出,远远优于代码;图形包含比字母代码更多的信息,它独特的表现方式能使编程人员在看到它的瞬间大致了解其作用。图形化的程序便于从整体上看清程序结构,使编程人员保持思路清晰,了解编程的进展情况。因为 XML 文档不单单是文本型的,它所存储的数据是结构化的,具有独特的自描述性,通过解析器可以得到结构化的数据,既能实现图形化编程元素的 XML 表示,又能将其作为编程技术中的通用语言,解决异构系统的编程,增强程序的复用性。通过本发明方法,借助 XML 的自描述性,可以方便地实现图形化程序与 XML 描述之间的相互转换,使编程工作更高效、更方便,并缩短产品开发周期。

#### 附图说明

[0016] 图 1 为本发明实施例的图形化编程的源文件存储及解析方法的步骤流程图。

[0017] 图 2 为本发明实施例的各种图形元件示意图。

#### 具体实施方式

[0018] 下面通过具体实施例,并配合附图,对本发明做详细的说明。

[0019] 图 1 是本实施例的图形化编程的源文件存储及解析方法的步骤流程图。对其具体说明如下:

[0020] 1) 对图形化编程所需的图形元件进行分类。本发明将图形元件分为两类:

[0021] 第一类是运算图形元件,包括算术运算和逻辑运算图形元件。算术运算有加法、减法、乘法、除法、取余数、取反数、取绝对值;逻辑运算有取反运算、逻辑与运算、逻辑或运算。该图形元件表示某种算术运算或逻辑运算,它具有可以设定数量和类型的输入端口属性和输出端口属性,及图形显示的属性如名称、高度、宽度、颜色。如表 1 所示。

[0022] 表 1. 算术运算和逻辑运算图形元件属性表

[0023]

元素名称	属性名称	类型
功能标示 (func)	功能名称 (funcname)	字符串
	名称 (name)	字符串
形状标示 (rect)	高度 (height)	数字
	宽度 (width)	数字
	颜色 (color)	字符串
输入标示 (input)	输入端口个数 (num)	数字
输出标示 (output)	输出端口个数 (num)	数字
端口标示 (value)	名称 (name)	字符串
	类型 (type)	字符串

[0024] 第二类是各个图形元件端口之间的连接线图形元件。即该图形元件用于连接各个图形元件的端口,表示各个图形元件端口的输入输出关系。它具有名称属性、起始端点属性、终点端点属性、拐点位置坐标属性。如表 2 所示。

[0025] 表 2. 连接线图形元件属性表

[0026]

元素名称	属性名称	类型
功能标示 (func)	功能名称 (funcname)	字符串
	名称 (name)	字符串
形状标示 (line)	宽度 (height)	数字
	颜色 (color)	字符串
起始端点 (start)	输入变量名称 (name)	字符串
	输入变量类型 (type)	数字
	坐标 (point)	数字
起始端点 (end)	输出变量名称 (name)	字符串
	输出变量类型 (type)	数字
	坐标 (point)	数字
拐点集合 (points)	单个拐点 (point)	数字

[0027] 2) 用 XML 技术对图形化编程的源文件进行存储。

[0028] 将某种图形元件的各种属性使用 XML 语言进行表示,每种图形元件各自存储为一个 XML 文件。可使用开发工具(如 Altova XMLSpy、Visual Studio 2010)根据图形元件属性创建 XML 文件。对于各图形元件的 XML 语言表示形式见下面各示例,其中标记“func”表示此图形元件完成的功能,用“funcname”的值来区分完成何种功能;标记“rect”表示图形元件的形状特征,“height”表示高度值,“width”表示宽度值,“color”表示颜色值;标记“input”表示输入,“num”为输入个数;标记“output”表示输入,“num”为输入个数;标记“value”表示输入或输入变量,“name”为变量名称,“type”表示变量类型。现举例说明如下:

[0029] 2-1) 算术运算的加法图形元件,如图 2(a) 所示。

[0030]

```
<? xml version= 1.0 encoding= utf-8 ? > //声明xml文件版本、编码格式
<func funcname = "adder" > //图形元件功能
  <rect height="20" width="40" color="gray"> //图形元件高度、宽度、颜色
    <input num="2"> //输入变量个数
      <value name="value1" type="int"/> //输入变量名称、变量类型
      <value name="value2" type="int"/> //输入变量名称、变量类型
    </input>
    <output num="1"> //输出变量个数
      <value name="value1" type="int"/> //输出变量名称、变量类型
    </output>
  </rect>
</func>
```

[0031] 2-2) 算术运算的减法图形元件,如图 2(b) 所示。

[0032]

```
<? xml version= 1.0 encoding= utf-8 ? > //声明xml文件版本、编码格式
<func funcname = "suber" > //图形元件功能
  <rect height="20" width="40" color="gray"> //图形元件高度、宽度、颜色
    <input num="2"> //输入变量个数
      <value name="value1" type="int"/> //输入变量名称、变量类型
      <value name="value2" type="int"/> //输入变量名称、变量类型
    </input>
    <output num="1"> //输出变量个数
      <value name="value1" type="int"/> //输出变量名称、变量类型
    </output>
  </rect>
</func>
```

[0033] 2-3) 算术运算的乘法图形元件,如图 2(c) 所示。

[0034]

```
<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "muler" > //图形元件功能
  <rect height="20" width="40" color="gray"> //图形元件高度、宽度、颜色
    <input num="2"> //输入变量个数
      <value name="value1" type="int"/> //输入变量名称、变量类型
      <value name="value2" type="int"/> //输入变量名称、变量类型
    </input>
    <output num="1"> //输出变量个数
      <value name="value1" type="int"/> //输出变量名称、变量类型
    </output>
  </rect>
</func>
```

[0036] 2-4) 算术运算的除法图形元件,如图 2(d) 所示。

[0037]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "diver" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="2">//输入变量个数
      <value name="value1" type="int"/>//输入变量名称、变量类型
      <value name="value2" type="int"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="int"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0038] 2-5) 算术运算的取余数图形元件, 如图 2(e) 所示。

[0039]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "memer" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="2">//输入变量个数
      <value name="value1" type="int"/>//输入变量名称、变量类型
      <value name="value2" type="int"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="int"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0040] 2-6) 算术运算的取反数图形元件, 如图 2(f) 所示。

[0041]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "inver" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="1">//输入变量个数
      <value name="value1" type="int"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="int"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0042] 2-7) 算术运算的取绝对值图形元件, 如图 2(g) 所示。

[0043]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "abser" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="1">//输入变量个数
      <value name="value1" type="int"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="int"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0044] 2-8) 逻辑运算的取反图形元件,如图 2(h) 所示。

[0045]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "anter" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="1">//输入变量个数
      <value name="value1" type="bool"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="bool"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0046] 2-9) 逻辑运算的或图形元件,如图 2(i) 所示。

[0047]

```

<? xml version= 1.0 encoding= utf-8 ? >
<func funcname = "orer" >//图形元件功能
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色
    <input num="2">//输入变量个数
      <value name="value1" type="bool"/>//输入变量名称、变量类型
      <value name="value1" type="bool"/>//输入变量名称、变量类型
    </input>
    <output num="1">//输出变量个数
      <value name="value1" type="bool"/>//输出变量名称、变量类型
    </output>
  </rect>
</func>

```

[0048] 2-10) 逻辑运算的与图形元件,如图 2(i) 所示。

[0049]

```
<? xml version= 1.0 encoding= utf-8 ? >  
<func funcname = "lander" >//图形元件功能  
  <rect height="20" width="40" color="gray">//图形元件高度、宽度、颜色  
    <input num="2">//输入变量个数  
      <value name="value1" type="bool"/>//输入变量名称、变量类型  
      <value name="value1" type="bool"/>//输入变量名称、变量类型  
    </input>  
  <output num="1">//输出变量个数
```

[0050]

```
      <value name="value1" type="bool"/>  
    </output>  
  </rect>  
</func>
```

[0051] 3) 在使用图形元件编程时,对所述 XML 文件进行解析,获得图形元件的属性信息。

[0052] 转换时,使用 XML 文件解析模块,该模块使用 C 或 C++ 或 C# 语言编写,用于解析 XML 文件,将图形元件属性解析到自定义的各种变量之中。

[0053] 下面提供解析过程的示例 C# 代码,此代码解析了代表“Adder”图形元件的 XML 文件:

[0054]

```
using System; //代码所需的C#库命名空间
using System.Collections.Generic; //代码所需的C#库命名空间
using System.Linq; //代码所需的C#库命名空间
using System.Text; //代码所需的C#库命名空间
using System.Windows; //代码所需的C#库命名空间
using System.Windows.Controls; //代码所需的C#库命名空间
using System.Xml; //代码所需的C#库命名空间
namespace ParseXML
{
    public class ParseAdder
    {
        public List<Variable> InputVarList = new List<Variable>(); //输入变量
        public List<Variable> OutputVarList = new List<Variable>(); //输出变量
        public Color AdderColor = new Color(); //图形元件颜色
        public int Width = 0; //图形元件宽度
        public int Height = 0; //图形元件高度

        public ParseAdder ()
        {
            string XmlFileName = //存储图形元件xml文件的路径
                System.IO.Path.GetDirectoryName(Application.ResourceAssembly.Location).ToString();
            ParseXmlFile(XmlFileName); //解析xml文件的函数
        }

        public void ParseXmlFile(string XmlFileName) //定义解析xml文件函数
        {
            XmlDocument AdderDoc = new XmlDocument(); //装载xml文件内容的变量
            AdderDoc.Load(XmlFileName); //装载xml文件
            XmlNode RootNode = AdderDoc.SelectSingleNode("rect"); //读取xml的rect节点
            this.Width = int.Parse(RootNode.Attributes[0].Value); //用变量存储图形元件宽度
            this.Height = int.Parse(RootNode.Attributes[1].Value); //用变量存储图形元件高度

            this.Color = RootNode.Attributes[2].Value; //用变量存储图形元件高颜色
            XmlNodeList NodeList = RootNode.ChildNodes; //读取xml文件的节点
            XmlNode InputNode = NodeList[0]; //读取input节点
        }
    }
}
```

[0055]

```
foreach (XmlNode Input in InputNode) //遍历input节点
{
    Variable Varia = new Variable();//定义一个输入变量
    Varia.VarName = Input.Attributes[0].Value;//存储变量名称
    Varia.VarType = (Type)Input.Attributes[0].Value;//存储变量类型
    this.InputVarList.Add(Varia);//将变量存储到输入变量集合
}
XmlNode OutputNode = NodeList[1]; //读取output节点
foreach (XmlNode Output in InputNode) //遍历output节点
{
    Variable Varia = new Variable();//定义一个输出变量
    Varia.VarName = Output.Attributes[0].Value; //存储变量名称
    Varia.VarType = (Type)Output.Attributes[0].Value; //存储变量类型
    this.OutputVarList.Add(Varia); //将变量存储到输出变量集合
}
}
}

public class Variable //代表变量的类
{
    private string VarNameValue = string.Empty; //变量名称
    private Type VarTypeValue = null; //变量类型

    public string VarName //与变量名称相关联的属性
    {
        get
        {return this.VarNameValue;}//获得变量名称
        set
        {this.VarNameValue = value;}//设置变量名称
    }

    public Type VarType //与变量类型相关联的属性
    {
        get
        {return this.VarTypeValue;}//获得变量类型
        set
        {this.VarTypeValue = value;}//设置变量类型
    }
}
}
```

[0056] 以上实施例仅用以说明本发明的技术方案而非对其进行限制,本领域的普通技术人员可以对本发明的技术方案进行修改或者等同替换,而不脱离本发明的精神和范围,本发明的保护范围应以权利要求所述为准。

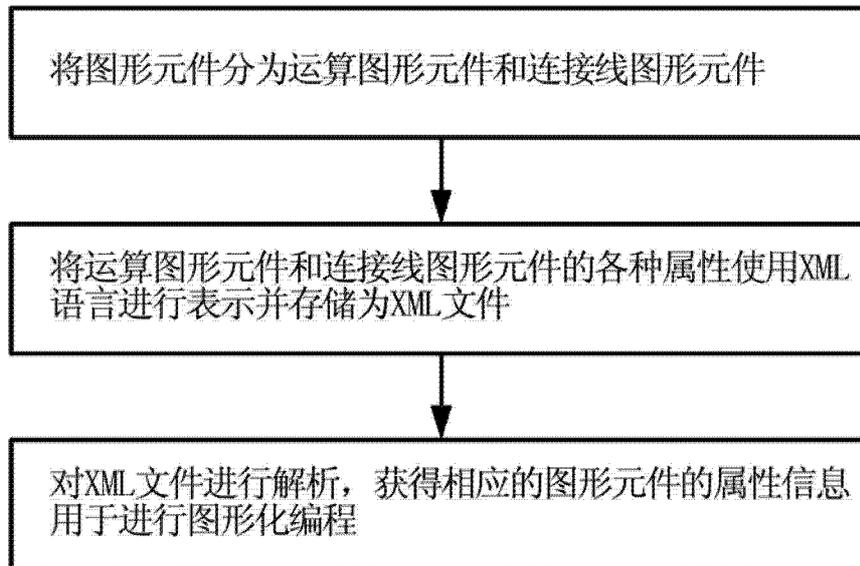


图 1

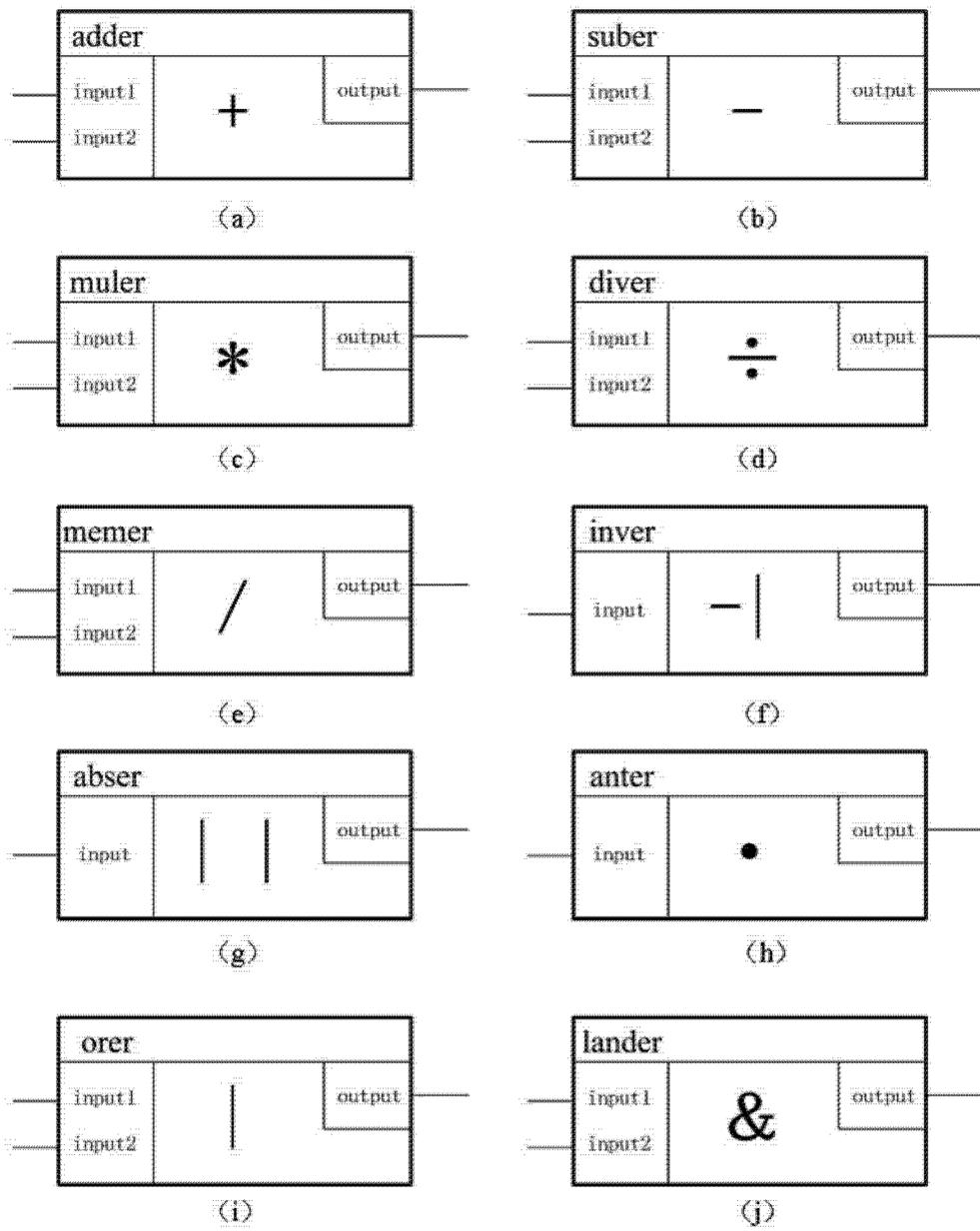


图 2