Easy Circuit

-An easy language to draw your circuit and figure it out

PLT proposal

Lei Zhang |LZ2343

Xingyue Zhou |XZ2299

Liming Zhang |LZ2342

Yingnan Li|YL2884

Wei Zhang |WZ2254

1. Motivation

Circuit design plays an important role in many areas. However, some circuit operations, such as computing equivalent resistance, or transforming a circuit, though intuitional in some ways, are still hard to formulary expressed in mathematic or program. Famous electronic design automation (EDA) and simulation program, such as Multisim[1], can help engineers to solve these problems. However, for common students, these developing environments seem to be heavy and expensive. Besides, even for people who are experts in using such simulation software, these applications sometimes still lacks of controllability and accuracy when dragging an element or drawing a wire on the screen. For these reasons, we want to design a slight, succinct, accurate and easy-to-control programming language, called Easy Circuit, to facilitate people in designing, controlling and analyzing circuits. By using our language, people who as long as possess basic knowledge of programming and circuit, can design a circuit from scratch, and then model it into a component with a function defining its properties. Such modularized components can be reused in a more complicated electronic system.

2. Description

Easy Circuit is a special language for people to design circuit in several aspects as followed:

- In our Easy Circuit language, user can conveniently define their own electronic components, such as combination of certain resistor and capacitor, by using the data type and operator in our language. Easy Circuit language can implement not only the basic operators, but also particular operators especially for circuit, such as series connection, parallel connection, and reciprocal.
- The models of some circuits may be frequently-used, thus the designer may want to save them as a modules or templates. The user may define a function to describe the property of such template, where the variant electronic components corresponds to the parameters in the function, and the function maps these variants to the output according to the user's definition of the function.
- Our language can be used to visualize a circuit, which means that the user can program a circuit by flow, and automatically generate an equivalent circuit diagram.
- The development environment of our Easy Circuit language would have a graphic user interface, thus when designing the circuit, user may choose to design their circuit not by coding, but by several clicks on the graphic. After the sketch of the circuit has been made, user can use the graphic interface to assign the value and denote input parameters and output variants, as well as the function between them.

3. Objectives

We want to design a language especially for circuit designer, who may use our language to define and calculate their circuit model. The language would have the following features:

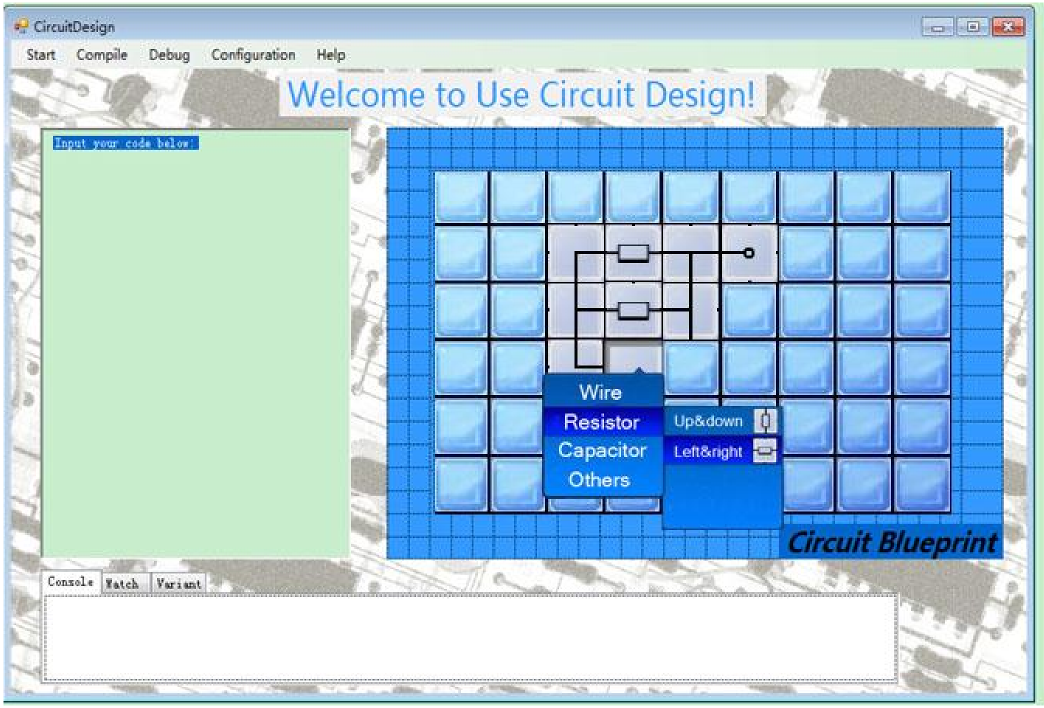
- **Easy to learn:** The style is similar to C language, and contains basic primitive types and

[1] http://en.wikipedia.org/wiki/Multisim

operators, such as int, float, long, double, boolean, +, -, *, /, etc.

- **Customized operators and data structures:** Our language contains operators and types especially for circuit design and calculation, such as series connection, parallel connection, and reciprocal; resistance, capacity, inductance, switch, power, wire, etc.
- **Functional:** User can use our language to model a circuit, i.e., to define a function to describe the properties of certain circuit. In these functions, the parameters are the adjustable component in the corresponding circuit.
- **Graphic user interface:** In the developing environment, user may use either codes or graphic interface or the combination of both, to program design the circuit.

  An anticipated interface may have the following look[2]:



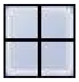
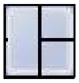
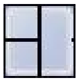
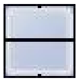
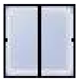
4. syntax:

4.1 definitions

box: is the keyword for users to choose their circuit diagrams. It is a data type to represent how nodes connect with each other inside the box. Our language designs prescriptive elements as followed.

box A = (char R or Rq or W or C or V, a four-binary-bits array)

“A four-binary-bits array”, namely (0/1, 0/1, 0/1, 0/1), shows the connection situation of the four nodes in the box: value 1 indicates connected while value 0 unconnected. The beginning node is in the north of the box and others are in a clockwise direction. See following examples.

(W, 1, 1, 1, 1) (W, 1, 1, 1, 0) (W, 1, 0, 1, 1) (W, 0, 1, 0, 1) (W, 1, 0, 1, 0)

Char “W” means the nodes of binary value 1 are connected by wire.

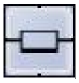
[2] Background picture from http://office.microsoft.com

Char “R” represents that our language users input the value of resistance in the box.

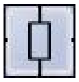
Char “Rq” indicates that our language users do not input the specific value of resistance. They design that there exists some resistance in the box and let their customers determine the value. Customers can click the resistor picture and input their value.

Char “C” shows a capacity in the box.

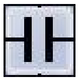
Char “V” means the box where we apply a voltage.



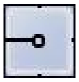
(R/Rq, 0, 1, 0, 1)



(R/Rq, 1, 0, 1, 0)



(C, 1, 0, 0, 0)



(V, 0, 0, 0, 1)

For example, box A = (Rq, 1, 0, 1, 0). This means that the north node and the south node are connected and there is some resistance in the box whose value is decided by the customers.

Header file: allows programmers to separate certain elements of a program's source code into reusable files. Header files contain forward declarations of classes, subroutines, variables, and other identifiers.

4.2 Functions

function	description
set(x,y)	Provide a box frame with length x and width y. Then, we have x*y boxes in the frame. Each box in the frame is initialized as an “empty” box, namely (W,0,0,0,0).
print(A, x, y)	Display the circuit diagram of box A with coordinate(x, y) in the graph window.
value(A,v)	Language users input a resistance value into box A=(R, 0/1, 0/1, 0/1, 0/1). There is a return value after functioning.
Inputvalue(A)	Customers input a resistance value into box A=(Rq, 0/1, 0/1, 0/1, 0/1) of the circuit diagram designed by language users. There is a return value after functioning.

4.3 Data types

Our language also contains basic data types as followed: int, float, double, char, Boolean.

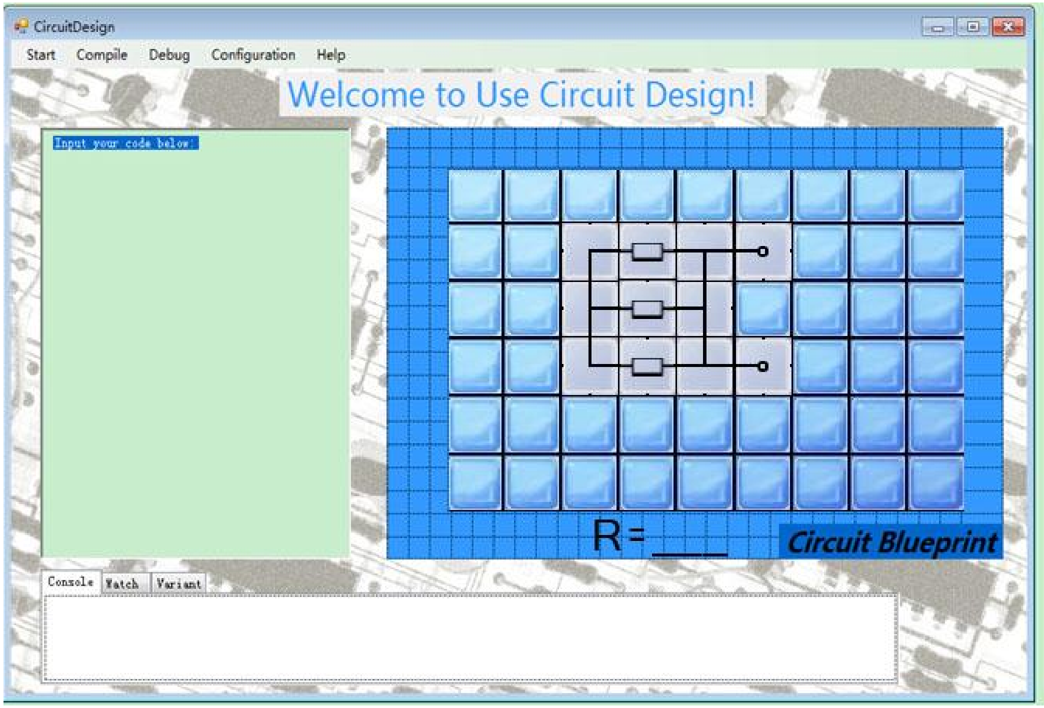
4.4 Operators:

a+b	addition	a!=b	not equal
a-b	subtraction	a>b	more than
a*b	multiplication	a<b	less than
a/b	division	a>=b	more than or equal to
|>a	reciprocal of value a	a<=b	less than or equal to
a==b	equals	// or /*…*/	comments

4.5 Declaration, Expression and Statement:

A declaration, an expression or a statement will be similar to those in java or C language.

5. Sample Code



```
# include <stdpackage. h>  /* stdpackage.h is defined as one of several kinds of header files. It
contains some basic functions and data interfaces, aiming to store the declarations of the
programs*/
void main
{
set ( 9, 6 ); /*to initiate a frame where the programmer can initiate any components he wants*/
//box A= new box (R, 0, 1, 0,1);/*started with// indicates that this statement or declaration or
expression is generated automatically, triggered by the action of choosing components
graphically.* /
//print(A, 2, 4);
//int a=value(A,5); /*For example, these three statements are generated due to the right-click at
coordinate(2,4), the choice of one resistance and the value input */
//box B= new box (R, 0, 1, 0,1);
//print(B, 3, 4);
//int b=value(B,10);
//box C= new box (Rq, 0, 1, 0,1);
//print(C, 4, 4);
//double v;
//int c=inputvalue(C);
// box D= new box (W, 0, 1, 1, 0);
// print(D, 2,3);
// box E= new box (W, 1, 1, 1, 0);
//print(E, 3,3);
// box F= new box (W, 1, 1, 0, 0);
//print(F,4,3);
// box G= new box (W,0, 1,1, 1);
//print(G,2,5);
// box H= new box (W, 1, 0, 1, 1);
//print(H,3,5);
//box I= new box (W, 1, 1, 0, 1);
//print(I,4,5);
//box J= new box (V, 0, 0, 0, 1);
//print(J,3,6);
//box K= new box (V, 0, 0, 0, 1);
//print(k,4,6);
int result;
result=|>(|>a+|>b+|>c);
printresult("%d", &result);
}
```