Modeling Method for Virtual Digital Component Library

Rui XU\textsuperscript{1,a}, Xi-Min YANG\textsuperscript{2}, Wei LU\textsuperscript{3}

\textsuperscript{1}Guangxi Vocational & Technical Institute Of Industry, Nanning 530001, China

\textsuperscript{2}College of Computer Science, South-Central University for Nationalities, Wuhan 430074, China

\textsuperscript{3}Guangxi University of Science and Technology Tus College of Digit, Liuzhou, 545006, China

\textsuperscript{a}xr_120@qq.com

Keywords: Virtual Experiment Platform, Virtual Component Library, Modeling, Digital Logic, Laboratory Teaching.

Abstract. The virtual experiment platform is introduced to address some issues in the laboratory teaching of engineering courses, such as equipment wear/damage and a lack of sound teaching strategies. The key module of the virtual platform is the virtual component library with the virtual digital component modeling as core. In this paper, the virtual digital component is abstracted and its model is described by using formal description. Then, three examples, based on the 74LS138, is given to explain the idea and implementation of the virtual digital component modeling (VDCM). In addition, the modeling we proposed is verified via a model-checking tool Uppaal, in terms of three properties: reachability, safety, and liveness. The experiment result shows that the proposed VDCM approach makes the virtual experiment platform more Scalable and reusable. Especially, a new virtual component can be modeled quickly through a simple configuration by using the VDCM.

Introduction

Laboratory has provided enough space for students studying the Information Technology (IT) subjects emphasizing that the combination of theory and practice. However, some problems still exist in the laboratory teaching: 1) the types of devices do not kept the latest, and the scale of device library is not large enough, 2) the device consumption is high, and 3) students cannot observe and understand the working process of chips and circuits visually. Fortunately, with the enormous advancements in the fields of modern communication technology, network and virtual reality, various kinds of application software are used to support the teaching processes [1].

Being flexible and helpful in practical, the virtual platform for digital circuit experiments is coming along to solve the above mentioned problems of the laboratory and has been widely applied in the engineering education. More and more universities have developed virtual platforms, such as University of Hannover [2]. In experimental teaching, the simulation software can help students understanding the engineering concepts in a better way [3]. By using virtual platforms, students will achieve more improvement and effective deep-learning experience [4], especially in learning the computer-related courses, such as Digital Logic and Computer Interface.

The virtual component library of digital circuits (VCLoDC) is the core of the virtual platform and affects directly the quality of the virtual platform in expansibility and secondary development [5].

Related work

Modeling makes the developers understand the design intent and improve the development speed. The model should be well structured and organized for its reusability [3]. In the early research work, two modeling methods were proposed for electronic circuit designing: Input/Output Buffer Information Specification (IBIS) and Simulation Program with Integrated Circuit Emphasis (SPICE) [6]. It is generally used to help students understand the functions of integrated circuits.

Contribution

A good model should be formalized and described more in detail for good implementation. This
paper mainly describes how to modeling on the logic function instead of the physical characteristics to show the principle of computing operation and its process.

The paper is organized as follows. Methodology explains the modeling and formal description of the combinational logic component and controlled combinational logic component. In model verification, we evaluate our work with a comparison to the analyze case. A conclusion is drawn at last.

Methodology

Preliminaries

In this section, two classical chips: 74LS138 is used to explain the features of two types of digital components, i.e., combinational logic component model and combinational logic component model with control inputs.

Modeling of Virtual Component

Based on the similarities and differences between CLC and CLCCI, the modeling of virtual component needs to be discussed from two aspects: modeling and verification. Before giving the model formal description, each attribute and its definition should be defined.

Combinational Logic Component (CLC)

There are some notations and symbols used for model formal description. P notes the set of pins. A pin is defined by a 4-tuple \(p(i, n, t, x)\), where \(i\) is the pin index, \(n\) is the pin name, \(t\) is the type of data transmission, and \(x\) is the pin value. If a pin is bidirectional, the type of its data transmission will be input (I), output (O), or input-and-output (I/O). \(x\) might be high-level (=1) or low-level (=0).

\(G\) is a set of pin groups, each of which is defined by a 2-tuple \(g(n, P^*)\), where \(n\) is the group name, which might be Data, Control, Power, or Address, and it is customizable. \(P^*\) is a proper subset of \(P\).

\(\forall g_i, g_j \in G \ (i \neq j), g_i \cap g_j = \emptyset\).

Virtual digital component (VDC) has some static attributes. A denotes the set of attributes of a VDC. The 2-tuple \(A(n, t)\) describes two essential attributes of a VDC, where \(n\) is the VDC name, and \(t\) is the package type of VDC, e.g., DIP, CP, PGA, BGA, CSP, MCM, etc. Note that the attributes can be extended by customers.

The function \(\Omega: P \rightarrow G\) denotes that \(P\) can be mapped to the \(G\). A pin \(p\) in \(P\) only can belong to one group \(g \in G\), and \(g = \Omega(p)\).

All of the possible input value and output value are represented by \(iv\) and \(ov\) respectively in the truth-table. IV and OV is the set of \(iv\) and \(ov\). And IV should be a min-term. A row of the truth-table is represented by \(v\), so the whole truth-table is represented by \(V\), which describes the mapping relation between IV and OV in a component. \(<iv, ov> \in V<IV, OV>\).

\(iv\) and \(ov\) should only be TRUE or FALSE. And, \(<IV, OV>\) can appear only once per truth-table.

\(F=IP[V]OP\) is a logic function, where IP and OP are the set of input pins and output pins, respectively. When IP has accepted a value, OP will get a value via a query in \(V\).

CLC with control inputs (CLCCI)

Besides the basic combination logic components, there is another type of components needing some control pins to make the output available. In order to guarantee reusability and extendibility, we introduce inheritance, a relationship between CLC and the CLCCI, to the virtual component model.

Symbol ‘\(\uparrow\)’ represents the inheritance relationship. \(A\uparrow B\) means A inherits from B.

\(E\) denotes the control expression with following constraints. It returns true or false.

• \(e\) and \(\overline{e}\) are both basic expressions. The value of \(e\) equals to the pin's value, and that of \(\overline{e}\) is contrary to the pin's value.

• If \(e_1\) and \(e_2\) are control expressions, \(e_1+e_2\), \(e_1\cdot e_2\), \(e_1\oplus e_2\), \(e_1\odot e_2\) and \((e_i)\) are control expressions too.
A control expression can be achieved if and only if above two constraints are finitely applied. Assume that \( e_1, e_2 \) and \( e_3 \) are known basic expressions, and a string of symbols \( "e_1 \cdot (e_2 + e_3)" \) is a control expression constructed by above constraint rules. The priority of an operator is the same as that of digital logic theory. The entire expression is delimited by the brackets.

**Definition 1.** CLCCI, which is denoted by a 7-tuple \( C_{\text{ctrl}} = (A, P, G, F, V, E) \). \( C_{\text{ctrl}} \) inherits from \( C \) and acquires the properties of \( C \). \( E \) is a control expression, and \( F \) is redefined as \( F = \text{IP}\{\text{V|E}\} \text{OP} \). \( \text{OP} \) will get a value from \( V \) when \( E \) is TRUE and \( \text{IP} \) accepts a value before executing the query in \( V \).

**Example 1.** Take 74LS138:

\[
C_{\text{ctrl}}^{74\text{LS}138} = (A, P, G, F, V, E) \tag{1}
\]

(a) \( A(n=74\text{LS}138, t=D\text{IP}) \)
(b) \( P = \{p(1, A, I, f), p(2, B, I, f), p(3, C, I, f), \ldots, p(14, Y4, O, f), p(16, Y6, O, f)\} \)
(c) \( G = \{g(\text{DataIn}, \{p_{1-6}\}), g(\text{DataOut}, \{p_{7,10-16}\}), g(\text{Power}, \{p_8, p_9\})\} \).
(d) \( E = (e_1) \cdot (e_2 + e_3) \), the values of \( e_1, e_2 \) and \( e_3 \) equal to \( p_6, p_4 \) and \( p_5 \) respectively.
(e) \( F = \text{IP}\{\text{V|E}\} \text{OP} \)

**Model Verification**

Uppaal is a model-checking tool that allows the timed automata to be built and verified [5]. An automaton template instantiated with parameters is a class of automata.

**Properties Analysis.** Three properties are specified with respect to the Uppaal system of components.
(a) Reachability, Uppaal system queries whether a given state formula can be satisfied by any reachable state.
(b) Safety, no state transition error will happen.
(c) Liveness, the state transition, which is expressed with the path formula \( A<>\phi \) in which \( \phi \) is satisfied, will eventually happen.

![Figure 1 74LS138 Model](image)

With the verifier via Uppaal, the validation analysis of 74LS138 model is given. According to the 74LS138 model shown in Figure 1, the function has been changed by the model that always detects the global variable \( \text{enable} \). When \( \text{enable} \) is TRUE the state transition is to be the Idle state, otherwise, the transition is to be the OFF state. The new control part is a self-loop state diagram,
and its edge needed a synchronizing signal to activate the function, in order to check the validity of the model. If the model is working, we set enable to TRUE, otherwise, enable is FALSE.

Table 1. 74LS138 Query.

<table>
<thead>
<tr>
<th>Property</th>
<th>Query statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reachability</td>
<td>$E&lt;&gt;Funtion.Func$</td>
<td>Function got to the Func state to calculate the result.</td>
</tr>
<tr>
<td>Safety</td>
<td>$A[]$ not deadlock</td>
<td>The system was deadlock free.</td>
</tr>
<tr>
<td>Liveness</td>
<td>$ControlProcess.Idle--&gt;$</td>
<td>The ControlProcess activated the ControlCheck function absolutely.</td>
</tr>
<tr>
<td></td>
<td>$ControlProcess.Idle$</td>
<td></td>
</tr>
</tbody>
</table>

Now, we have completed the translation of the component model to the automata in Uppaal. The query statements are to check whether the models are reachable, live and safe. When the properties are satisfied, the green light is turn on. Otherwise, the red will light to alert you. The verification results are shown in Figure 2.

![Figure 2. Verification of 74LS138.](image)

Comparison of Modeling and Virtual Platform

There are several ways to design and simulate the digital component model, LabView, Matlab/Simulink etc. Each of them has their own advantages and disadvantages. In the paper, the VDC model has been described by formal language and its properties have been analyzed and verified. As shown in the Table 1 and Table 2, it is worthwhile to undertake a comparison based on the VDC model and the other existing research.

Table 2. Comparative Analysis of Modeling.

<table>
<thead>
<tr>
<th></th>
<th>Pins Grouping</th>
<th>Component Connection</th>
<th>Inheritance</th>
<th>Complexity</th>
<th>General Utility</th>
<th>Learning Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modelica-based</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>Component-based</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
<tr>
<td>IBIS</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>H</td>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>VDC</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>L</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

NOTE: Y-yes; N-no; H-high; L-low

First, there is no pin group in the Modelica-based and Component-based modeling. Otherwise, a model, which can group the pins, is easily to assign the value away to the virtual component. Secondly, the connection between different components is not considered in IBIS and VDC. In this
paper, we only consider the simulation. Thirdly, VDC has a good inheritance that other models do not have. After designing a base model, the new model inherits from the base model if you intend to add a new component type.

Above all, VDC model has some advantages: 1) the grouping of the pins helps developers to understand the composition of pins and the structural characteristics of components clearly. 2) VDC model has hierarchical characteristics, reusability and extensibility. Implementation inheritance is the mechanism whereby a sub-model reuses the code in a base model. 3) VDC modeling method is simpler than the other methods with lower learning cost.

Conclusion

In this paper, we present a modeling method for designing digital logic components, including giving the formal description of the VDC. The model has no connectors between different components in the VDC. There are two kinds of the VDC model: CLC and CLCCI. The former makes the foundation for the latter, the latter is the inheritance to the former. And the simulation result verified by the Uppaal shows that, the VDC model has the properties of model: Safety, Reachable and Liveness. The VDC model has the advantages not only in the promotion of reuse but also the convenience for the development of the virtual component platform.

Reference


