Robei 3.1 Software User Guide

—Guosheng Wu

Robei is a cross platform chip design tool that aims to simplify design procedure, transparent intellectual properties and reduce complexity. It makes chip design like playing with boxes by breaking down hardware into three basic elements: module, port and wire. Through these elements, engineer can implement either top-down or bottom-up design. Standard Verilog code can be integrated with other EDA tools generated from design diagram. Robei also runs on embedded platforms, which makes it distinctive from other EDA design software.

I. Introduction

Robei aims to simplify user interface for FPGA design, transparent intellectual properties and reduce complexity. The modern user interface of Robei combines diagram based visual design for structure level and algorithm coding at function level. All these parts are designed to be as simple as possible for designers. By simply playing with Robei’s elements, FPGA designers can construct their project either by bottom-up or top-down mechanism.

The special features in Robei:

- Robei provides a visual design method.
- Robei is very user friendly and easy to learn.
- It provides code generation on structural level, which can reduce coding mistakes and increase productivity.
- It provides an integrated code editor for Verilog coding directly. This feature opens the door for complex design.
- Robei is also the first FPGA simulation tool that can work on embedded platforms.
- Robei is the world smallest EDA tool for FPGA design and simulation.
- Waveform viewer is small, efficient, modern and user friendly. For example, different colors are used to differentiate waves near to each other.
- Pin assignments constrain file support.

Copyright © Robei
Fig. 1-1 Robei visual user interface

Fig. 1-2 Robei diagram design example
II. User Interface

When you start the program, you can get to Robei user interface shows as Fig. 2-1.

![Robei User Interface](image)

Fig. 2-1 Robei main window

Like most of other software tools, it has menu and toolbar on top of the user interface. Icons on toolbar clearly indicate the meanings. On the left side, there is a "Toolbox" widget which contains all pre-designed models. Robei has some pre-installed models in "System" container. User defined models locate in "Current" container. The center is design workspace, which has a big rectangle diagram in current design with name "module1". Developer can view and modify each object's property through "Property" widget on the right side. Messages indicating result, error and warning are showing in the bottom "Output" widget.

2.1 Menus and Toolbar

Robei has both Menus and Toolbar for convenient operations. The most commonly used methods for file operation are placed in "File" menu. Copy, paste, cut and delete actions locate in "Edit "menu. Add module, port and connect ports with wire locate in "Tools" menu. Besides that, there are compile and run related buttons in "Build", and view related actions like zoom, code view and wave view in "View" menu. In case some widgets are closed by mistake, you can re-open it from "Window"
menu. In order to increase the design speed, Robei has some shortcuts on "Toolbar", so designer can easily refer to them by single click.

![Fig.2-2 Toolbar](image)

### 2.2 ToolBox

There are two containers initially installed in ToolBox, one is "System" model container, another is "Current" model container. "System" contains all the models created by Robei. The folder of it locates at: "C:\ProgramData\Robei". The "Current" container is used to hold models created by developer. "Current" container also has a folder, but folder location may variant as current project changes because folder locates at: "working directory". Besides that, you can create your container by right click and select "Add".

![Fig.2-3 Toolbox window (left) and add library dialog](image)

If models changed, you can reload all models in that container by selecting "Reload" on right click. You also can remove it by clicking on "Remove". Besides that, there are two other actions can help you to organize models, like "Sort"-"Ascending" and "Sort"-"Descending".

### 2.3 Property
The "Property" widget displays properties of selected item in "project". User can edit property values from it and the changes immediately shows on workspace. Some properties in model are not enabled to edit, and ports on model are protected in order to avoid miss editing.

![Property Window](image)

**Fig.2-4 Property window**

### 2.4 Output

The output widget is used for displaying messages generated from project like actions and error messages when running a simulation. So please check the message often for design mistakes.

![Output Widget](image)

**Fig. 2-5 output widget**

### 2.5 Workspace

Workspace is the graphical design area which enables developer to operate on current module by simply playing with ports, models, wires. Project has two different views, one is Graph view and another is Code view. Graph view displays diagram structure and code view is an editor for Verilog based algorithms.

Copyright © Robei
22 assign \( \{CO, S\} = A + B + CI; \)

Fig. 2-6 Graph View and Code View
III. Elements

Robei employs three minimum elements to represent Verilog components in hardware design: module, port and wire. In Verilog, circuits are represented by a set of "modules". A module may be only a gate, a flip-flop, a register, but also can be an ALU, a controller or a SOC system. We can consider a module as an abstract chip, which have different ports (pins) to communicate with other chips. A finished design module is considered as model, which locates in the Toolbox and can be reused by just clicking to add into current module. Port is the interface channel for each module and model. Wire used to connect the ports for signal communication.

3.1 Module

Module, the basic type of design procedure, can be considered as a black box. Inside this box, designer can place ports for outside connections, algorithm codes and models. Each module can have zero or more ports for communication with other modules or models. The code view tab allows engineers to add or modify algorithm code easily to realize certain behaviors.

Based on usage status, a module can have different types. Currently under developing one has the type of "module", but once is used in other modules, it automatic changes to "model" type as only certain properties can be modified in order to keep consistency with previous design. There are some another special types like "testbench" and "constrain". "Testbench" is the top level design, with stimulate code for simulation and "constrain" is the pin assignment and constrain description for FPGA chips.

(1) Module: the basic type of design procedure. It allows user to add and remove parts from it. Each module can have zero or more ports for communication with other modules or models.
(2) Model: pre-designed module, user can only modify color, name and parameters. Some properties are protected as the inside structures already designed.

(3) Testbench: test bench is very important for verifying design with stimulate. Testbench do not care about port's inout property.

(4) Constrain: constrain is used for pin assignment. Based on different FPGA vendor, pin assignment can generate different constrain file. For example, for Xilinx
FPGA, it can generate UCF file, but for Altera FPGA, it can generate QSF file. If one port on the design is a bus, you can specify each wire with numbers for connection.

![Fig. 3-4 constrain design](image1)

Based on the setting on FPGA vendor, Robei can generate different pin assignment files. In order to choose the right vendor, designer need to click on the menu: "Settings" -> "FPGA".

![Fig. 3-5 choose FPGA vendor](image2)

In "View" -> "CodeView", there is generated code base on constrain design.
3.2 Port

A port may correspond to a pin on a chip, an edge connector on a board, or any logical channel of communication with a block of hardware. The detail properties are listed in fig.3. The type of port varies a lot as Robei supports many types in Verilog, like reg, wire, supply, etc. There is "Datasize" option for port, which specifies the size of port if it is a bus. Some interesting features that worth to mention are port can only slide on edges of module, and when module changes, port sticking to edges all the time.

Remember, all ports on model cannot be modified except some color information, but you still can connect them to other ports and change the position by moving them along the edge of model.

3.3 Wire

Wire connects two ports and responses for signal transmission. Most of time, wire will inherent the color and datasize from the first connected ports. Based on
different datasize, wire has different thickness. Robei helps to check whether two connected ports have same datasize or not when compiling.

Module, port and wire are three basic elements in Robei. By breaking down structure level hardware design into three elements, Robei reduces FPGA entry level requirements.
VI. Examples

The fast way to learn and manage this software is playing with examples. Some examples with simulation present in this section to help designer quickly to get familiar with Robei.

4.1 Gate Logic

Consider two inputs "a" and "b" for AND logic, the truth table of output Y shows as follow. The algorithm is realized by operation: \( y = a \& b \).

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Fig. 4-1 AND gate and truth table

We can represent it very easy in Robei. Let’s try to design an AND gate on it. First open Robei, save the current new project with name "andgate" to a place, then add two ports as input, one port as output with the names: a, b, y. Make sure to change the "Inout" property for y as output. (By default, if you add port on the right side, it will automatic set inout property to output.)

Fig. 4-2 "andgate" model

The property for a,b and y shows as Fig. 4-3:

Copyright © Robei
Then click on the "Code" tab under project, insert code "assign y = a&b;".

Notice that the first line of code is not start from 1, which is because this is not complete Verilog code. If you want to see the full code, please go to menu: "View "->"CodeView".
Any errors will show up in output widget when you press "Run" button on the toolbar. In order to verify the functionality of this design, a testbench is necessary. Click on "New" button to create another project, "New Project Setting" dialog pops up, change "Project Type" to "testbench", and change input ports to 2, output ports to 1. This will automatic add 3 ports to the design.

Fig. 4-6 new project setting dialog

Then save it as "and_test" to the same directory with "andgate" model. If you forgot to set module type to testbench, you can changed it later in property editor, and remember to save it as testbench.
Note: if you don't save in the same directory, it cannot find "andgate" in Toolbox.

Remember to check the type of 3 ports, they should have the same property as in fig. 4-7.

<table>
<thead>
<tr>
<th>Name</th>
<th>Inout</th>
<th>DataType</th>
<th>Datasize</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>input</td>
<td>wire</td>
<td>1</td>
<td>first input</td>
</tr>
<tr>
<td>b</td>
<td>input</td>
<td>wire</td>
<td>1</td>
<td>second input</td>
</tr>
<tr>
<td>y</td>
<td>output</td>
<td>wire</td>
<td>1</td>
<td>output</td>
</tr>
</tbody>
</table>

Fig. 4-7 Property of ports on "and_test"

Find "andgate" model from "Current" Toolbox, click on it then click on "and_test" in workspace to add the model. Most of the properties in this model cannot be modified except the "Parameters", "Name" and "Color".

Fig. 4-8 Add andgate to "and_test"

Click on "Wire" in toolbar, then connect a to a, b to b and y to y.
Click "Code" tab and add code as shown as in fig.4-10.

```
initial begin
  // set initial value
  a = 1'b0;
  b = 1'b0;

  #5 // after 5 unit delay change a
  a = 1'b1;

  #5 // after 5 unit delay change b
  b = 1'b1;

  #5 // after 5 unit delay change a again
  a = 1'b0;

  #5 // change b again
  b = 1'b0;

  #5 // stop the simulation after 5 unit delay
  $finish;

  // remember to use $finish to stop simulation.
end
```

Fig. 4-10 "and_test" code view
Then click on "Run" to check errors. If everything works well, the compilation will finish in no time. If no errors show up, click on "Wave" button to see the waveform.

![Waveform](image)

**Fig. 4-11 Wave window**

Click on the right hand side signals to add on waveform viewer, then click on "zoom full" button to see the waveform. Move around with cursor to play with waveform and view the value changes. Compare the result with truth table to see whether this design can meet the requirement or not.

In gate level design, you can play with the algorithm to implement different gates. With the same testbench, the result behavior will be different. Here is a list of gates and operators in Verilog.

<table>
<thead>
<tr>
<th>Gate</th>
<th>Symbol</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
<td>&amp;</td>
<td>assign y=a&amp;b;</td>
</tr>
<tr>
<td>Or</td>
<td></td>
<td>assign y=a</td>
</tr>
<tr>
<td>Xor</td>
<td>^</td>
<td>assign y=a^b;</td>
</tr>
<tr>
<td>Not</td>
<td>~</td>
<td>assign y=!a;</td>
</tr>
</tbody>
</table>

### 4.2 Four bits counter

In this example, we want to build a four bits counter with Robei. The counter has reset, enable, clock and count ports. You can modify the color of each part, make this model pretty.
Similar to previous project, add four ports: clock, reset, enable and count to a new module, then change the "Inout" of count to "output", "Datatype" to "reg" and "Datasize" to "3:0".

After that, click on code tab, add the following code to code view.
Fig. 4-15 Code view of "counter" model

After that, save the current project as "counter.model", so the project name changed to "counter". Then you can check mistakes, generate code with "Run" button.

We can create a testbench to verify the counter. Create a new project and set "Module Type" to "testbench". Save it as "counter_test.test" in the same directory with "counter.model".

Fig. 4-16 new project for counter testbench

After that, change the ports name to clock, reset, enable and result correspondingly. Set "Data size" of result to "3:0". Modify "Datatype" of clock, reset and enable to "reg".

Copyright © Robei
Click on counter from ToolBox, add to "counter_test". Then connect all the ports in the same way as the following image. You can modify the color of wires, blocks, models and ports in order to distinguish them.

Click on "Code" view, add the following code:
Then check errors and try to compile it. After fixing all the errors, click on "Run" and "Wave" to see the simulation result. Here is the waveform window shown in fig. 4-20.

![Waveform](image)

Fig. 4-20 Waveform of "counter_test" testbench

Congratulation, you are done with this example. From now on, you can design your own models following similar procedures described in this example.
V. Register

You need a license to remove limitation in Robei. Click "Help" on the menu bar then select "Register". Use the Software ID, your name and email to purchase license by clicking on "Get License" button. Please note that you must provide the correct email address, otherwise, you may not be able to receive your license. Once got the license, fill into "License" area and click on Register.

![Register dialog](image1.png)

Fig. 5-1 Register dialog

Another dialog will show up, which indicates you have registered. Have fun!

![Registration complete](image2.png)

Fig. 5-2 Registration complete

If you are using organization group license, please change the "License Type" to "Organization", and use the company username and password provide by Robei to finish the registration. If you need to purchase group license, please contact us.
Fig. 5-3 Registration with group license
IV Conclusion

Robei aims to help developer visualizing their design and reduce mistakes from early stage. Thanks for choosing Robei, we'd like to hear any feed backs from you to improve the software. Please visit our website often for updates: http://robei.com and contact us by email: robeisale@gmail.com.