Data flow modeling

Prerequisites

Previous modules are required for familiarity with Verilog syntax. Basic knowledge of dataflow will also help.

Suggested Time

You should be able to learn the basics of data flow modeling in 4-5 hours.

Learning Objectives

After reading this module you will be able to

- Describe the continuous assignment \(\textit{assign}\) statements in verilog.
- Design practical digital circuits in Verilog using Dataflow model.
- Explain different types of delay for continuous assignment statements.

Introduction

Dataflow modeling is a higher level of abstraction compared to gate level modeling. To design a circuit in this abstraction level the designer should be aware of data flow of the design. The gate level modeling becomes very complex for a VLSI circuit, hence dataflow modeling became a very important way of implementing the design. In dataflow modeling most of the design is implemented using continuous assignments, which are used to drive a value onto a net. The continuous assignments are made using the keyword assign, which we will learn in the next section.
The assign statement

The assign statement is used to make continuous assignment in the dataflow modeling. The assign statement usage is given below:

Assign $out = in0 + in1$; // $in0 + in1$ is evaluated and then assigned to $out$

Please note that:

- The LHS of assign statement must always be a scalar or vector net or a concatenation, it cannot be a register.
- Continuous statements are always active statements, which mean that if any value on the RHS changes the value of LHS changes automatically.
- Registers or nets or function calls can come in the RHS of the assignment.
- The RHS expression is evaluated whenever one of its operands changes. Then the result is assigned to the LHS.
- Delays can be specified in assign statement.

Examples:

Assign $out [3:0] = in0 [3:0] \& in1 [3:0]$; // does bit wise multiplication of $in0$ and $in1$ and stores the results in $out$

Assign $\{o3, o2, o1, o0\} = in0 [3:0] \| \{in1 [2:0], in2\}$; // Use of concatenation.

Concatenations are used to combine data, the above example does bit wise or of $in0$ and $\{in1 [2:0], in2\}$ which essentially means that

$$o0 = in0 [0]*in2; o1 = in0 [1]*in1 [0];$$
$$o2 = in0 [2]*in1 [1]; o3 = in0 [3]*in1 [2];$$
**Implicit Continuous Assignment**

While declaring the net we can do continuous assignment using regular continuous assignment, example is given below

Wire out;
Assign out = in0 ^ in1;

Here we have declared the out wire and assigned the in0^in1 to it, we can achieve the same with

Wire out = in0 ^ in1;

Instead of declaring a net and then writing a continuous assignment on the net. Verilog provides a shortcut by which a continuous assignment can be placed on a net when it is declared. There can be only one implicit declaration assignment per net because a net is declared only once.

**Implicit Net Declaration**

In Verilog during implicit assignment, if LHS is declared than it will assign the RHS to the declared net but if the LHS is not defined it will automatically create a net for the signal name.

Wire in0, in1;
Assign out = in0 ^ in1;

In the above example out is undeclared, but Verilog makes an implicit net declaration for out.

Introduction and examples in data flow modeling: This ppt explains the continuous assignments in data-flow level. It also gives the example of 4X1 MUX and full adder to explain the data flow modeling. PPT link ==>
Delays

There are three types of delays associated with dataflow modeling. They are: Normal/regular assignment delay, implicit continuous assignment delay and net declaration delay.

Normal/regular assignment delay

This delay is applicable when the signal in LHS is already defined and this delay represents the delay in changing the value of the already declared net. Assign #10 out = in0 | in1;

If there is any change in the operands in the RHS, then RHS expression will be evaluated after 10 units of time and evaluated expression will be assigned to LHS. Let’s say that at time t, if there is change in one of the operands in the above example, then the expression is calculated at t+10 units of time. It means that if in0 or in1 changes value again before 10 time units than the values of in1 and in2 at the time of re-computation (t+10) are considered.

Implicit continuous assignment delay

An equivalent method is to use an implicit continuous assignment to specify both a delay and an assignment on the net.

Wire #10 out = in0 ^ in1;

Is same as

Wire out;

Assign #10 out = in0 ^ in1;
Net declaration delay
In this case the delay is associated with net instead of the assignment. If any changes applied to net is delayed according to this delay.

Wire #10 out;
Assign out = in;
Is same as
Wire out;
Assign #10 out = in;

To read more about delays in data flow please refers to Media: delay in data flow rev2.pptx.

Operators
Verilog provides many different operator types. Operators can be arithmetic, logical, relational, equality, bitwise, reduction, shift, concatenation, or conditional. Some of these operators are similar to the operators used in the C programming language. Each operator type is denoted by a symbol. Table 1 shows the complete listing of operator symbols classified by category. This list is incomplete and includes most basic operators. A more complete list of operators can be accessed online.

Verilog data flow level operators
<table>
<thead>
<tr>
<th>Operator Symbol</th>
<th>Operation Performed</th>
<th>Number of Operands</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Multiply</td>
<td>2</td>
</tr>
<tr>
<td>/</td>
<td>Divide</td>
<td>2</td>
</tr>
<tr>
<td>+</td>
<td>Addition</td>
<td>2</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>2</td>
</tr>
<tr>
<td>%</td>
<td>modulus</td>
<td>2</td>
</tr>
<tr>
<td>**</td>
<td>Power</td>
<td>2</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>2</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>2</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than equal to</td>
<td>2</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than equal to</td>
<td>2</td>
</tr>
<tr>
<td>==</td>
<td>equality</td>
<td>2</td>
</tr>
<tr>
<td>!=</td>
<td>inequality</td>
<td>2</td>
</tr>
<tr>
<td>~</td>
<td>bitwise negation</td>
<td>2</td>
</tr>
<tr>
<td>&amp;</td>
<td>bitwise and</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bitwise or</td>
</tr>
<tr>
<td>^</td>
<td>bitwise xor</td>
<td>2</td>
</tr>
<tr>
<td>^~ or ^~</td>
<td>bitwise xnor</td>
<td>2</td>
</tr>
<tr>
<td>{}</td>
<td>Concatenation</td>
<td>Any numbers</td>
</tr>
<tr>
<td>?:</td>
<td>Conditional</td>
<td>3</td>
</tr>
</tbody>
</table>

This ppt explains operators in data flow level abstraction level in Verilog. Operator uses, their syntax and some examples are given. The operators explained in this PPT are assign operator, Concatenation Operator, shift operator etc. PPT link ==> Media: Expressions, Operators & Operands - Part 1.ppt

This ppt is the continuation of the previous ppt on Expression, Operators and Operands Media: Expressions, Operators & Operands - Part 2.ppt