第7章　数据流模型化

本章讲述Verilog HDL语言中连续赋值的特征。连续赋值用于数据流行为建模；相反，过程赋值用于(下章的主题)顺序行为建模。组合逻辑电路的行为最好使用连续赋值语句建模。

7.1　连续赋值语句

连续赋值语句将值赋给线网(连续赋值不能为寄存器赋值)，它的格式如下(简单形式)：

```
assign LHS_target = RHS_expression;
```

例如，

```
wire [3:0] Z, Preset, Clear;    /线网说明
assign Z = Preset & Clear;      /连续赋值语句
```

连续赋值的目标为Z，表达式右端为“Preset & Clear”。注意连续赋值语句中的关键词assign。

连续赋值语句在什么时候执行呢？只要在右端表达式的操作数上有事件(事件为值的变化)发生时，表达式即被计算；如果结果值有变化，新结果就赋给左边的线网。

在上面的例子中，如果*Preset*或*Clear*变化，就计算右边的整个表达式。如果结果变化，那么结果即赋值到线网Z。

连续赋值的目标类型如下：

1) 标量线网

2). 向量线网

3) 向量的常数型位选择

4) 向量的常数型部分选择

5) 上述类型的任意的拼接运算结果

下面是连续赋值语句的另一些例子：

```
assign BusErr = Parity| (One & OP);
assign Z = ~ (A | B) & (C | D) & (E | F);
```

只要*A*、*B*、*C*、*D*、*E*或*F*的值变化，最后一个连续赋值语句就执行。在这种情况下，计算右边整个表达式，并将结果赋给目标Z。

在下一个例子中，目标是一个向量线网和一个标量线网的拼接结果。

```
wire Cout, Cin;
wire [3:0] Sum, A, B;
. . .
assign {Cout, Sum} = A + B + Cin;
```

因为*A*和*B*是4位宽，加操作的结果最大能够产生5位结果。左端表达式的长度指定为5位(*Cout* 1位，*Sum* 4位)。赋值语句因此促使右端表达式最右边的4位的结果赋给*Sum*，第5位(进位位)赋给*Cout*。

下例说明如何在一个连续赋值语句中编写多个赋值方式。

```
assign Mux = (S = = 0)? A : 'bz,
       Mux = (S = = 1)? B : 'bz,
       Mux = (S = = 2)? C : 'bz,
       Mux = (S = = 3)? D : 'bz;
```

这是下述4个独立的连续赋值语句的简化书写形式。

```
assign Mux = (S = = 0)? A : 'bz;
assign Mux = (S = = 1)? B : 'bz;
assign Mux = (S = = 2)? C : 'bz;
assign Mux = (S = = 3)? D : 'bz;
```

7.2 举例

下例采用数据流方式描述1位全加器。

```
module FA_Df (A, B, Cin, Sum, Cout);
  input A, B, Cin;
  output Sum, Cout;

  assign Sum = A ^B ^Cin;
  assign Cout = (A & Cin) | (B & Cin) | (A & B);
endmodule
```

在本例中，有两个连续赋值语句。这些赋值语句是并发的，与其书写的顺序无关。只要连续赋值语句右端表达式中操作数的值变化（即有事件发生），连续赋值语句即被执行。如果 A 变化，则两个连续赋值都被计算，即同时对右端表达式求值，并将结果赋给左端目标。

7.3 线网说明赋值

连续赋值可作为线网说明本身的一部分。这样的赋值被称为线网说明赋值。例如：

```
wire [3:0] Sum = 4'b0;
wire Clear = 'b1;
wire A_GT_B = A > B, B_GT_A= B > A;
```

线网说明赋值说明线网与连续赋值。说明线网然后编写连续赋值语句是一种方便的形式。参见下例。

```
wire Clear;
assign Clear = 'b1;
```

等价于线网声明赋值：

```
wire Clear = 'b1;
```

不允许在同一个线网上出现多个线网说明赋值。如果多个赋值是必需的，则必须使用连续赋值语句。

7.4 时延

如果在连续赋值语句中没有定义时延，如前面的例子，则右端表达式的值立即赋给左端表达式，时延为0。如下例所示显式定义连续赋值的时延。

```
assign #6 Ask = Quiet | | Late;
```

规定右边表达式结果的计算到其赋给左边目标需经过6个时间单位时延。例如，如果在时刻5，*Late*值发生变化，则赋值的右端表达式被计算，并且 *Ask*在时刻11(= 5 +6)被赋于新值。

图7-1举例说明了时延概念。

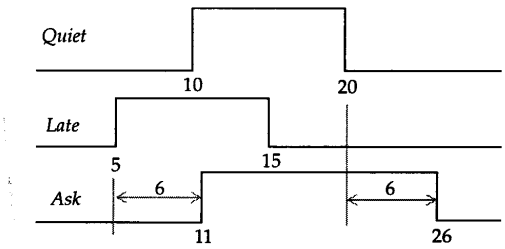

图7-1 连续赋值语句中的时延

如果右端在传输给左端之前变化，会发生什么呢？在这种情况下，应用最新的变化值。下例显示了这种行为：

```
assign #4 Cab = Drm;
```

图7-2显示了这种变化的效果。右端发生在时延间隔内的变化被滤掉。例如，在时刻 5，*Drm*的上升边沿预定在时刻9显示在*Cab*上，但是因为*Drm*在时刻8下降为0，预定在*Cab*上的值被删除。同样，*Drm*在时刻18和20之间的脉冲被滤掉。这也同样适用于惯性时延行为：即右端值变化在能够传播到左端前必须至少保持时延间隔；如果在时延间隔内右端值变化，则前面的值不能传输到输出。

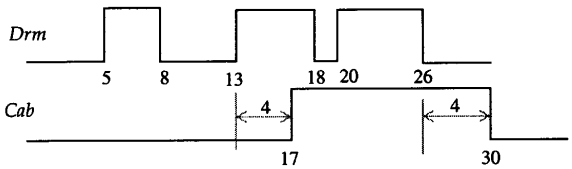

图7-2 值变化快于时延间隔

对于每个时延定义，总共能够指定三类时延值:

1) 上升时延

2) 下降时延

3) 关闭时延

这三类时延的语法如下:

```
assign # (rise, fall, turn-off) LHS_target = RHS_expression;
```

下面是当三类时延值定义为0时，如何解释时延的实例：

```
assign #4 Ask = Quiet || Late;      // One delay value.
assign # (4,8) Ask = Quick;         // Two delay values.
assign # (4,8,6) Arb = & DataBus;  // Three delay values.
assign Bus = MemAddr [7:4];         // No delay value.
```

在第一个赋值语句中，上升时延、下降时延、截止时延和传递到 x的时延相同，都为 4。在第二个语句中，上升时延为 4，下降时延为8，传递到x和z的时延相同，是4和8中的最小值，即4。在第3个赋值中，上升时延为 4，下降时延为8，截止时延为 6，传递到x的时延为4(4、8和6中的最小值)。在最后的语句中，所有的时延都为 0。

上升时延对于向量线网目标意味着什么呢？如果右端从非0向量变化到0向量，那么就使用

下降时延。如果右端值到达z，那么使用下降时延；否则使用上升时延。

7.5 线网时延

时延也可以在线网说明中定义，如下面的说明。

```
wire #5 Arb;
```

这个时延表明*Arb*驱动源值改变与线网*Arb*本身间的时延。考虑下面对线网*Arb*的连续赋值语句：

```
assign #2 Arb = Bod & Cap;
```

假定在时刻10，*Bod*上的事件促使右端表达式计算。如果结果不同，则在2个时间单位后赋值给*Arb*，即时刻12。但是因为定义了线网时延，实际对*Arb*的赋值发生在时刻17(= 10 + 2 + 5)。图7-3的波形举例说明了不同的时延。

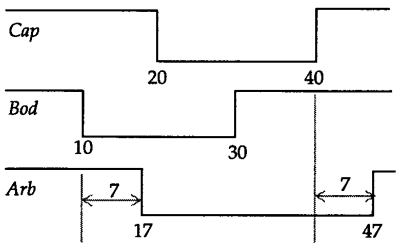

图7-3 带有赋值时延的线网时延

图7-4很好地描述了线网时延的效果。首先使用赋值时延，然后增加任意线网时延。

如果时延在线网说明赋值中出现，那么时延不是线网时延，而是赋值时延。下面是A的线网说明赋值，2个时间单位是赋值时延，而不是线网时延。

```
wire #2 A = B - C;      //赋值时延
```

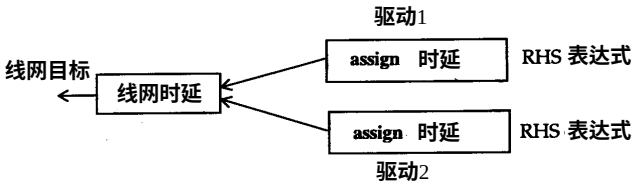

图7-4 线网时延的效果

7.6 举例

7.6.1 主从触发器

下面是图5-9所示的主从触发器的Verilog HDL模型。

```
module MSDFF_DF (D, C, Q, Qbar);
  input D, C;
  output Q, Qbar;
  wire NotC, NotD, NotY, Y, D1, D2, Ybar, Y1,;Y2

  assign NotD = ~ D;
  assign NotC = ~ C;
  assign NotY = ~ Y;

  assign D1 = ~ (D & C);
  assign D2 = ~ (C & NotD);
```

```
  assign Y = ~ (D1 & Ybar);
  assign Ybar = ~ (Y & D2);
  assign Y1 = ~ (Y & NotC);
  assign Y2 = ~ (NotY & NotC);
  assign Q = ~ (Qbar & Y1);
  assign Qbar = ~ (Y2 & Q);
endmodule
```

7.6.2 数值比较器

下面是8位(参数定义的)数值比较器数据流模型。

```
module MagnitudeComparator(A, B, AgtB, AeqB, AltB);
parameter BUS = 8;
parameter EQ_DELAY = 5, LT_DELAY = 8, GT_DELAY = 8;
input [1 : BUS]A, B;
output AgtB, AeqB, AltB;

assign #EQ_DELAY AeqB = A = = B;
assign #GT_DELAY AgtB = A > B;
assign #LT_DELAY AltB = A < B;
endmodule
```

习题

1. 举例说明截止时延在连续赋值语句中如何使用？
2. 当对同一目标有2个或多个赋值形式时，如何决定目标有效值？
3. 写出图5-10所示的奇偶产生电路的数据流模型描述形式。只允许使用 2个赋值语句，并规定上升和下降时延。
4. 使用连续赋值语句，描述图5-12所示的优先编码器电路的行为。
5. 假定：

```
tri0 [4:0] Qbus;
assign Qbus = Sbus;
assign Qbus = Pbus;
```

如果*Pbus*和*Sbus*均为高阻态z，*Qbus*上的值是什么？