2.7. Системные функции

Для эффективной работы разработчиков при отладке и тестировании созданных проектов зачастую необходимо включать в программный код операторы, обеспечивающие обмен информацией с пользователем, управление процессом моделирования (например, его остановкой) и т.п. Средства, позволяющие решать указанные задачи, называются в языке Verilog системными функциями (system tasks). Идентификаторы всех системных функций начинаются со знака «**$**», например:

```
$display
$write
$monitoron
```

В данной главе ограничимся рассмотрением только нескольких основных системных функций, наиболее часто встречающихся в Verilog-проектах.

Функция **$display** позволяет выводить в процессе моделирования информацию в стандартное устройство вывода. В среде Active-HDL вывод осуществляется в окно консоли.

Синтаксис функции **$display** приведен ниже:

```
$display(Список_Вывода);
```

Список_Вывода может включать строки, сигналы или выражения. Использование функции **$display** очень похоже на использование оператора printf в языке Си. Строки, стоящие в списке вывода, могут содержать управляющие символы, начинающиеся со знака процента «%». Управляющие символы непосредственно на экран не выводятся, хотя при этом изменяют определенным образом формат выводимой строки. Полный список управляющих символов системной функции **$display** и их функциональное назначение приведены в табл. 2.1.

Таблица 2.1

Управляющие символы системной функции **$display**

№	Управляющий символ	Функциональное описание
1	%d или %D	Отобразить значение в десятичной форме
2	%b или %B	Отобразить значение в двоичной форме
3	%s или %S	Отобразить строку
4	%h или %H	Отобразить значение в шестнадцатеричной форме
5	%c или %C	Отобразить ASCII-символ
6	%m или %M	Отобразить иерархическое имя (не требует аргументов)
7	%v или %V	Отобразить уровень приоритета сигнала
8	%o или %O	Отобразить значение в восьмеричном формате
9	%t или %T	Отобразить значение во временном формате
10	%e или %E	Отобразить действительное число в формате с плавающей запятой (например, 1.063e-5)
11	%f или %F	Отобразить действительное число в формате с фиксированной запятой (например, 103.452)
12	%g или %G	Отобразить действительное число в том из форматов, который даст более короткую запись

Аргументы для управляющих символов берутся в списке вывода, следующем за строкой по порядку появления. Например, чтобы вывес-

ти значение сигнала *A* в двоичной форме в сочетании с поясняющим текстом «*A* = », следует записать функцию **$display** таким образом:

```
$display ("A = %b", A);//Вывод на печать значения сигнала A
                       //в двоичном формате
```

В рассматриваемом примере в первой строке *Списка_Вывода* при помощи знака %b задана позиция для вывода значения сигнала *A* и его формат (двоичный), а идентификатор сигнала *A* помещен в список вывода следующим элементом. В этом случае следует говорить, что в строке присутствует один управляющий знак %b, а сигнал *A* является его параметром (аргументом).

Количество управляющих знаков в строке может быть произвольным. При этом параметры в списке вывода должны следовать в том же порядке, что и управляющие сигналы в строке. Например, если в текущий момент времени сигнал *A* = 8'hF0 и сигнал *B* = 8'h7A, то приведенная ниже системная функция

```
$display ("Сигнал A = %h; Сигнал B = %h.", A, B);
```

выведет на экран следующую строку:

```
Сигнал A = F0;  Сигнал B = 7A.
```

Если возникает необходимость в выводе на экран знака «%», то необходимо задавать его в строках парой символов « %% ». В противном случае знак процента и следующий за ним символ будут восприняты компилятором как управляющий символ, что, скорее всего, приведет к ошибке компиляции. Например, команда

```
$display("Выполнено 50%."); //Неправильно
```

вызовет ошибку, так как компилятор воспримет пару знаков «%.» как недопустимый управляющий символ. К необходимому результату «Выполнено 50%.» приведет команда

```
$display ("Выполнено 50%%."); //Правильно
```

Все допустимые в строках специальные символы ASCII-кода (такие как /n – перенос строки, /t – табуляция и т.п.) могут также использоваться в функциях **$display** с корректным выводом на экран.

Для определения и вывода на экран текущего значения модельного времени удобно применять системную функцию **$time**. Пример ее использования приведен ниже:

```
$display ("Время %d, A = %b; B = %b.", $time, A, B);
```

В результате выполнения данной команды может появиться, например, такая строка:

```
Время 2152, A = 11001001; B = 00xx1001.
```

Кроме функции **$display,** зачастую используются также специальные функции для отслеживания состояния сигнала во время моделирования – **$monitor**, **$monitoron** и **$monitoroff**.

Команда

```
$monitor (Список_Вывода);
```

включает режим отслеживания сигналов. При этом *Список_Вывода* формируется так же, как и для функции **$display**. Отличие функции **$monitor** состоит в том, что ее *Список_Вывода* выводится не один раз при выполнении команды, а всякий раз при изменении одного из сигналов, входящих в список.

Команды **$monitoroff** и **$monitoron** предназначены для временного приостановления и возобновления процесса отслеживания сигналов соответственно.

Рекомендуется установка в список вывода функции **$time** для повышения читаемости полученных таблиц значения сигналов.

Для управления имитационным моделированием используются системные функции **$stop** и **$finish**, первая из которых временно приостанавливает процесс моделирования, а вторая – прекращает моделирование полностью.

Следует отметить, что такие развитые приложения для разработки Verilog-проектов, как пакет Active-HDL, обладают гораздо более удобными средствами отладки по сравнению с теми, которые определены в стандарте языка Verilog. К их числу относятся, например, представляемые в графической и табличной форме временные диаграммы, стимуляторы, средства управления процессами моделирования и т.п. (полное описание перечисленных средств отладки и моделирования приведено в Приложении А).

В связи с этим в среде Active-HDL такие системные функции, как **$monitor**, **$monitoron**, **$monitoroff**, **$stop** и **$finish,** используются намного реже, нежели в случае применения более простых (свободно распространяемых) компиляторов языка Verilog.