3.4. Использование редактора блок-диаграмм пакета Active-HDL

Среда Active-HDL предоставляет пользователям широкий набор средств эффективного управления процессами разработки и верификации цифровых электронных устройств. В частности, время, затрачиваемое на построение структурной модели, значительно сокращается при использовании входящего в состав пакета Active-HDL развитого редактора блок-диаграмм (Block diagram editor).

Панели инструментов редактора блок-диаграмм представлены на рис. 3.4. Как видно из рис. 3.4, интерфейс редактора является стандартным для Windows-приложений и легко понимается интуитивно.

Созданная в редакторе блок-диаграмма транслируется специальной программой в код на языке описания оборудования (VHDL или Verilog, в соответствии с установками пользователя). При помощи соответствующей кнопки можно просмотреть результирующий код, что особенно полезно для начинающих пользователей в качестве примера хорошего стиля в структурном моделировании на языке Verilog.

На рис. 3.5 показаны обозначения, принятые в блок-диаграммах среды Active-HDL.

Пример блок-диаграммы, реализующей сумматор для восьмиразрядных беззнаковых целых чисел в среде Active-HDL, показан на рис. 3.6.

Блок-диаграммы могут состоять из блоков, символов, связей специальных знаков (заземление, высокий уровень сигнала и т.д.), а также текстовых и графических комментариев. Кроме того, к блок-диаграммам можно добавлять фрагменты кода на языке Verilog, которые будут помещены в итоговую программу.

Следует обратить внимание, что приведенный пример содержит 7 включений устройства *Full_Adder*. Они имеют одинаковое имя, так как ссылаются на один и тот же модуль, написанный на языке Verilog, но в то же время отдельные включения различаются по уникальному идентификатору, расположенному над компонентом (например, *U3*, *U12*). Каждый блок (компонент) может быть реализован кодом на языках Verilog или VHDL, а также с помощью диаграмм конечных автоматов или блок-диаграмм. Все шины и цепи на схеме также могут иметь уни-

кальные идентификаторы.

Ниже приведен Verilog-код, полученный в результате автоматической трансляции блок-диаграммы, представленной на рис. 3.6.

```
//-------------------------------------------------
// Title : Adder_8bit
// Design : Verilog_book
// Author : serh_syd
// Company : NPK_SIGNAL
//
//-------------------------------------------------
// File       :
// c:\My_Designs\Verilog_book\compile\Adder_8bit.v
// Generated : Thu Jul 18 17:03:06 2002
// From :
// c:\My_Designs\Verilog_book\src\Adder_8bit.bde
// By : Bde2Verilog ver. 2.01
//
//-------------------------------------------------
// Description :
// Simple adder for 8-bit unsigned numbers
//
//-------------------------------------------------

// synopsys translate_off
`ifdef _VCP
`else
`define library(a,b)
`endif
// synopsys translate_on

// ---------- Design Unit Header ----------
`timescale 1ps / 1ps

module Adder_8bit (X1, X2, Carry, Sum) ;

// ------------ Port declarations ---------
input [7:0] X1,X2;
wire [7:0] X1,X2;
```

```
output Carry;
wire Carry;
output [7:0] Sum;
wire [7:0] Sum;

// ----------- Signal declarations --------
wire NET1450, NET1462, NET1474, NET1486;
wire NET1498, NET1510, NET1522, NET1534;
wire NET1538, NET1553, NET1561, NET1565;
wire NET1573, NET1581, NET1589, NET1597;
wire NET1605, NET1653, NET1679, NET1687;
wire NET1695, NET1703, NET1711, NET1745;
wire NET1753, NET1761, NET1769, NET1775;
wire NET1783, NET1791, NET1799;

// -------- Component instantiations -------
DeMux U1
(
   .X(X1),
   .Y0(NET1534),
   .Y1(NET1605),
   .Y2(NET1653),
   .Y3(NET1679),
   .Y4(NET1687),
   .Y5(NET1695),
   .Y6(NET1703),
   .Y7(NET1711)
);

Mux U11
(
   .X(Sum),
   .Y0(NET1745),
   .Y1(NET1753),
   .Y2(NET1761),
   .Y3(NET1769),
   .Y4(NET1799),
   .Y5(NET1791),
   .Y6(NET1783),
```

```
    .Y7(NET1775)
);

Full_Adder U12
(
    .Carry_In(NET1450),
    .Carry_Out(NET1462),
    .S(NET1753),
    .X1(NET1605),
    .X2(NET1553)
);
DeMux U2
(
    .X(X2),
    .Y0(NET1538),
    .Y1(NET1553),
    .Y2(NET1561),
    .Y3(NET1565),
    .Y4(NET1573),
    .Y5(NET1581),
    .Y6(NET1589),
    .Y7(NET1597)
);

Full_Adder U3
(
    .Carry_In(NET1462),
    .Carry_Out(NET1474),
    .S(NET1761),
    .X1(NET1653),
    .X2(NET1561)
);

Full_Adder U4
(
    .Carry_In(NET1474),
    .Carry_Out(NET1486),
    .S(NET1769),
    .X1(NET1679),
```

```
    .X2(NET1565)
);

Half_Adder U5
(
    .Carry(NET1450),
    .S(NET1745),
    .X1(NET1534),
    .X2(NET1538)
);

Full_Adder U6
(
    .Carry_In(NET1486),
    .Carry_Out(NET1498),
    .S(NET1799),
    .X1(NET1687),
    .X2(NET1573)
);

Full_Adder U7
(
    .Carry_In(NET1498),
    .Carry_Out(NET1510),
    .S(NET1791),
    .X1(NET1695),
    .X2(NET1581)
);

Full_Adder U8
(
    .Carry_In(NET1510),
    .Carry_Out(NET1522),
    .S(NET1783),
    .X1(NET1703),
    .X2(NET1589)
);

Full_Adder U9
```

```
(
   .Carry_In(NET1522),
   .Carry_Out(Carry),
   .S(NET1775),
   .X1(NET1711),
   .X2(NET1597)
);

endmodule
```