A.5 Chip Body (Implementation)

A.5.1 Parts

A typical chip consists of several lower-level chips, connected to each other and to the chip input/output pins in a certain "logic" (connectivity pattern) designed to deliver the chip functionality. This logic, written by the HDL programmer, is described in the chip body using the format:

PARTS:

internal chip part;
internal chip part;
...
internal chip part;

Where each internal chip part statement describes one internal chip with all its connections, using the syntax:

chip name (*connection*, ..., *connection*);

Where each connection is described using the syntax:

part's pin names = *chip's pin name*

(Throughout this appendix, the presently defined chip is called chip, and the lower-level chips listed in the PARTS section are called *parts*).

A.5.2 Pins and Connections

Each *connection* describes how one pin of a part is connected to another pin in the chip definition. In the simplest case, the programmer connects a part's pin to an input or output pin of the chip. In other cases, a part's pin is connected to another pin of another part. This internal connection requires the introduction of an *internal pin*, as follows:

Internal Pins In order to connect an output pin of one part to the input pins of other parts, the HDL programmer can create and use an internal pin, say v, as follows:

```
Part1 (..., out=v);       // out of Part1 is piped into v
Part2 (in=v, ...);        // v is piped into in of Part2
Part3 (a=v, b=v, ...);    // v is piped into both a and b of Part3
```

Internal pins (like v) are created as needed when they are specified the first time in the HDL program, and require no special declaration. Each internal pin has fan-in 1 and unlimited fan-out, meaning that it can be fed from a single source only, yet it can feed (through multiple connections) many other parts. In the preceding example, the internal pin v simultaneously feeds both Part2 (through in) and Part3 (though a and b).

Input Pins Each input pin of a part may be fed by one of the following sources:

- an input pin of the chip
- an internal pin
- one of the constants true and false, representing 1 and 0, respectively

Each input pin has fan-in 1, meaning that it can be fed by one source only. Thus Part (in1=v, in2=v, ...) is a valid statement, whereas Part (in1=v, in1=u, ...) is not.

Output Pins Each output pin of a part may feed one of the following destinations:

- an output pin of the chip
- an internal pin

A.5.3 Buses

Each pin used in a connection—whether input, output, or internal—may be a multi-bit bus. The widths (number of bits) of input and output pins are defined in the chip header. The widths of internal pins are deduced implicitly, from their connections.

In order to connect individual elements of a multi-bit bus input or output pin, the pin name (say x) may be subscripted using the syntax x[i] or x[i...j]=v, where v is an internal pin. This means that only the bits indexed *i* to *j* (inclusive) of pin *x* are connected to the specified internal pin. An internal pin (like v above) may not be subscripted, and its width is deduced implicitly from the width of the bus pin to which it is connected the first time it is mentioned in the HDL program.

The constants true and false may also be used as buses, in which case the required width is deduced implicitly from the context of the connection.

Example

```
CHIP Foo {
   IN in[8]    // 8-bit input
   OUT out[8]  // 8-bit output
   // Foo's body (irrelevant to the example)
}
```

Suppose now that Foo is invoked by another chip using the part statement:

```
Foo(in[2..4]=v, in[6..7]=true, out[0..3]=x, out[2..6]=y)
```

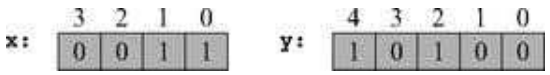
where v is a previously declared 3-bit internal pin, bound to some value. In that case, the connections in[2..4]=v and in [6..7]=true will bind the in bus of the Foo chip to the following values:

	7	6	5	4	3	2	1	0	(Bit)
in:	1	1	?	v[2]	v[1]	v[0]	?	?	(Contents)

Now, let us assume that the logic of the Foo chip returns the following output:

	7	6	5	4	3	2	1	0
out:	1	1	0	1	0	0	1	1

In that case, the connections out[0..3]=x and out[2..6]=y will yield:


x:
3 2 1 0
0 0 1 1
y:
4 3 2 1 0
1 0 1 0 0