Appendix B : Genlib Format

This is a description of the **genlib** format which is used to specify library gates in SIS.

1 Combinational Gate Specification

A cell is specified in the following format:

```
GATE <cell-name> <cell-area> <cell-logic-function>
<pin-info>
    .
    .
<pin-info>
```

<*cell-name*> is the name of the cell in the cell library. The resulting net-list will be in terms of these names.

<*cell-area*> defines the relative area cost of the cell. It is a floating point number, and may be in any unit system convenient for the user.

<*cell-logic-function*> is an equation written in conventional algebraic notation using the operators '+' for OR, '*' or nothing (space) for AND, '!' or "' (post-fixed) for NOT, and parentheses for grouping. The names of the literals in the equation define the input pin names for the cell; the name on the left hand side of the equation defines the output of the cell. The equation terminates with a semicolon.

Only single-output cells may be specified. The '!' operator may only be used on the input literals, or on the final output; it is not allowed internal to an expression. (This constraint may disappear in the future).

Also, the actual factored form is significant when a logic function has multiple factored forms. In principle, all factored forms could be derived for a given logic function automatically; this is not yet implemented, so each must be specified separately. Note that factored forms which differ by a permutation of the input variables (or by De Morgan's law) are not considered unique.

Each <*pin-info*> has the format:

```
PIN <pin-name> <phase> <input-load> <max-load>
    <rise-block-delay> <rise-fanout-delay>
    <fall-block-delay> <fall-fanout-delay>
```

<*pin-name*> must be the name of a pin in the <*cell-logic-function*>, or it can be * to specify identical timing information for all pins.

<*phase*> is INV, NONINV, or UNKNOWN corresponding to whether the logic function is negative-unate, positive-unate, or binate in this input variable respectively. This is required for the separate rise-fall delay model. (In principle, this information is easily derived from the logic function; this field may disappear in the future).

<*input-load*> gives the input load of this pin. It is a floating point value, in arbitrary units convenient for the user.

<*max-load*> specifies a loading constraint for the cell. It is a floating point value specifying the maximum load allowed on the output.

<*rise-block-delay*> and <*rise-fanout-delay*> are the rise-time parameters for the timing model. They are floating point values, typically in the units nanoseconds, and nanoseconds/unit-load respectively.

<*fall-block-delay*> and <*fall-fanout-delay*> are the fall-time parameters for the timing model. They are floating point values, typically in the units nanoseconds, and nanoseconds/unit-load repsectively.

All of the delay information is specified on a pin-by-pin basis. The meaning is the delay information for the most critical pin is used to determine the delay for the gate.

2 Latch Specification

Latches are specified as follows:

```
LATCH <cell-name> <cell-area> <cell-logic-function>
<pin-info>
    .
    .
<pin-info>
<latch-spec>
[<clock-delay-info>]
[<constraint-info>]
```

<*cell-name*>, <*cell-area*>, <*cell-logic-function*> and <*pin-info*> are the same as in the combinational case.

<*latch-spec*> has the following format:

```
SEQ <latch-input> <latch-output> <latch-type>
```

<*latch-input*> must be the name of the output of the cell. Thus, <*latch-input*> is described as a function of <*latch-output*> in the <*cell-logic-function*>. A special name "ANY" can be used for <*latch-output*> to specify to the mapper that <*latch-output*> needs not be a signal name in the <*cell-logic-function*>. This is useful for describing D-type flip-flops and latches. <*latch-type*> can be ACTIVE_HIGH or ACTIVE_LOW for transparent latches; RISING_EDGE or FALLING_EDGE for edge-triggered flip-flops; or ASYNCH for asychronous latches.

<*clock-delay-info*> is necessary for synchronous latches and flip-flops. It gives the propagation delay from the clock pin to the output. Its format is as follows:

```
CONTROL <clock-pin-name> <input-load> <max-load>
    <rise-block-delay> <rise-fanout-delay>
    <fall-block-delay> <fall-fanout-delay>
```

<*constraint-info*> gives the setup and hold time constraints with respect to the clock signal. The format is:

```
CONSTRAINT <pin-name> <setup-time> <hold-time>
```

If not specified, the values of 0.0 are assumed.

3 Examples

Example 0 of latch specification:

```
LATCH "d-latch" 80 Q=D;
PIN D NONINV 1 999 1 .2 1 .2
SEQ Q ANY ACTIVE_HIGH
CONTROL CLK 1 999 1 .2 1 .2
CONSTRAINT D 0.2 0.2
```

Example 1 of <cell-logic-function>:

```
O = !(!I1 * I2);
```

This is legal and defines a NAND gate with 1 inverted input; it could also be specified as

```
O = I1 + !I2;
```

There is no advantage or disadvantage to either representation; only one need be given.

Example 2 of <cell-logic-function>:

```
O = !((I1 * I2) + !(I3 + I4));
```

is incorrectly specified. It must be re-written as either

```
O = !((I1 * I2) + (!I3 * !I4));
```

or

```
O = (!I1 + !I2)*(I3 + I4);
```

Again, there is no advantage or disadvantage to either representation as they differ only by De Morgan’s law. Only one need be given. Note that there are no other factored forms for this equation.

Example 3 of <cell-logic-function>:

```
O = a*b + a*c + b*c;
```

This is one specification of a majority gate. Another possible specification (which IS considered different) is:

```
O = a*(b + c) + b*c;
```

Any permutation of the input variables does not provide a new specification; hence, these are the only two useful representations for a majority gate logic function. Both should be provided to the technology mapping program, until further notice.