Library Harmonization for Timing

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>2003/11/17</td>
</tr>
<tr>
<td>0.1</td>
<td>2003/12/12</td>
</tr>
<tr>
<td>0.2</td>
<td>2004/01/19</td>
</tr>
<tr>
<td>0.3</td>
<td>2004/02/23</td>
</tr>
</tbody>
</table>

Template for liberty/ALF xref examples

/* liberty */  /* ALF */
Table of contents

1.0 Basic description of timing arcs.................................................................3
   1.1 Timing measurement overview.................................................................3
   1.2 Delay and slew.........................................................................................4
   1.3 Delay and slew with retain.................................................................6
   1.4 Setup and hold .......................................................................................8
   1.5 Recovery and removal .................................................................10
   1.6 Co-dependent setup and hold............................................................11
   1.7 Co-dependent recovery and removal ..............................................12
   1.8 Setup and hold with nochange ......................................................13
   1.9 Maximum skew constraint...............................................................15
   1.10 Minimum period and minimum pulsewidth constraints .............16
   1.11 Threshold definitions .......................................................................18
   1.12 Conditional timing arcs .................................................................20
   1.13 Timing arcs involving bus pins ......................................................22

2.0 Interoperability with SDF ......................................................................23
1.0 Basic description of timing arcs

1.1 Timing measurement overview

Timing arcs are defined not only by standalone statements but also by the context in which the statements appear. In both liberty and ALF, a timing arc is defined in the context of a CELL identified by a Cell Name. A declaration of each PIN involved in the timing arc is required, referred herein as the Pin Name and the Related Pin Name.

In liberty, the timing model is further defined inside the declaration of the Pin Name. The occurring edge combinations are defined by Timing Type and Timing Sense.

Some timing data in liberty appear as timing model, others appear as a timing attribute. A timing attribute supports only a scalar value, whereas a timing model supports a mathematical calculation model. A timing arc description in liberty is shown in Figure 1 on page 3.

FIGURE 1. Timing arc description in liberty

```vlib
/* liberty */
cell (CellName) {  
  pin(RelatedPinName) {  
    direction : RelatedPinDirection;  
  }  
  pin(PinName) {  
    direction : PinDirection;  
    timing() {  
      timing_type : TimingType;  
      timing_sense : TimingSense;  
      related_pin : "RelatedPinName";  
      /* lib_TimingModel */  
      ModelKeyword (CalculationType) { values ( /* lib_Data */ ); }  
    }  
  }  
/* lib_TimingAttribute */  
AttributeKeyword : AttributeValue;  
}
```

In ALF, pins and timing arcs are declared separately. A timing arc is established by the declaration of a VECTOR, separate from the declaration of each PIN involved in the timing arc. The edge combinations are defined by a Vector Expression. A timing arc description in ALF is shown on Figure 2 on page 4.
FIGURE 2. Timing arc description in ALF

```plaintext
/* ALF */
CELL CellName {
    PIN RelatedPinName {
        DIRECTION = RelatedPinDirection;
    }
    PIN PinName {
        DIRECTION = PinDirection;
    }
    VECTOR (VectorExpression) {
        /* ALF_TimingModel */
        // see Figure 4 on page 6 through Figure 21 on page 17
    }
}
```

The ALF description of a timing model and its mapping to a liberty construct depends on the nature of the timing measurement. The following table shows an overview of measurement and the pointer to the corresponding ALF description and the liberty to ALF mapping table.

<table>
<thead>
<tr>
<th>Measurement</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>delay, slew</td>
<td>see Section 1.2 on page 4</td>
</tr>
<tr>
<td>delay, retain, slew</td>
<td>see Section 1.3 on page 6</td>
</tr>
<tr>
<td>independent setup, hold</td>
<td>see Section 1.4 on page 8</td>
</tr>
<tr>
<td>independent recovery, removal</td>
<td>see Section 1.5 on page 10</td>
</tr>
<tr>
<td>co-dependent setup, hold</td>
<td>see Section 1.6 on page 11</td>
</tr>
<tr>
<td>co-dependent recovery, removal</td>
<td>see Section 1.7 on page 12</td>
</tr>
<tr>
<td>setup, hold with nochange constraint</td>
<td>see Section 1.8 on page 13</td>
</tr>
<tr>
<td>maximum skew constraint</td>
<td>see Section 1.9 on page 15</td>
</tr>
<tr>
<td>minimum period and minimum pulsewidth constraint</td>
<td>see Section 1.10 on page 16</td>
</tr>
</tbody>
</table>

### 1.2 Delay and slew
FIGURE 3. Delay and slew measurements

TABLE 2. Mapping of liberty and ALF constructs for delay and slew measurements

<table>
<thead>
<tr>
<th>liberty construct</th>
<th>ALF construct</th>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>combinational</td>
<td></td>
<td>positive_unate</td>
<td>cell_rise</td>
<td>DELAY</td>
<td>01 RPN -&gt; 01 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>rise_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>cell_fall</td>
<td>DELAY</td>
<td>10 RPN -&gt; 10 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>fall_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative_unate</td>
<td></td>
<td>cell_rise</td>
<td>DELAY</td>
<td>10 RPN -&gt; 01 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall</td>
<td>DELAY</td>
<td>01 RPN -&gt; 10 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>non_unate</td>
<td></td>
<td>cell_rise</td>
<td>DELAY</td>
<td>?? RPN -&gt; 01 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall</td>
<td>DELAY</td>
<td>?? RPN -&gt; 10 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>three_state_enable</td>
<td></td>
<td>positive_unate</td>
<td>cell_rise ?</td>
<td>DELAY</td>
<td>01 RPN -&gt; Z1 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_rise ?</td>
<td></td>
<td></td>
<td>01 RPN -&gt; Z0 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall ?</td>
<td></td>
<td></td>
<td>01 RPN -&gt; Z0 PN</td>
<td></td>
</tr>
<tr>
<td>three_state_disable</td>
<td></td>
<td>negative_unate</td>
<td>cell_rise ?</td>
<td>DELAY</td>
<td>10 RPN -&gt; Z1 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall ?</td>
<td></td>
<td></td>
<td>10 RPN -&gt; Z0 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall ?</td>
<td></td>
<td></td>
<td>10 RPN -&gt; Z0 PN</td>
<td></td>
</tr>
<tr>
<td>rising_edge</td>
<td></td>
<td>cell_rise</td>
<td>DELAY</td>
<td>01 RPN -&gt; 01 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall</td>
<td>DELAY</td>
<td>01 RPN -&gt; 10 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td>SLEWRATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TABLE 2. Mapping of liberty and ALF constructs for delay and slew measurements

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>falling_edge</td>
<td>?</td>
<td>cell_rise</td>
<td>DELAY</td>
<td>10 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
<tr>
<td></td>
<td>?</td>
<td>cell_fall</td>
<td>DELAY</td>
<td>10 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
<tr>
<td>preset</td>
<td>positive_unate</td>
<td>cell_rise</td>
<td>DELAY</td>
<td>01 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
<tr>
<td>negative_unate</td>
<td>cell_rise</td>
<td></td>
<td>DELAY</td>
<td>10 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
<tr>
<td>clear</td>
<td>positive_unate</td>
<td>cell_fall</td>
<td>DELAY</td>
<td>10 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
<tr>
<td></td>
<td>negative_unate</td>
<td>cell_fall</td>
<td>DELAY</td>
<td>01 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td></td>
<td>SLEWRATE</td>
</tr>
</tbody>
</table>

FIGURE 4. Description of delay and slew measurements in ALF

VECTOR (VectorExpression) {
    /* ALF_TimingModel */
    DELAY {
        FROM { PIN = RelatedPinName ; }
        TO { PIN = PinName ; }
        /* ALF_data */
    }
    SLEWRATE {
        PIN = PinName ;
        /* ALF_data */
    }
}

1.3 Delay and slew with retain
FIGURE 5. Retain, delay, and slew measurements

TABLE 3. Mapping of liberty and ALF constructs for retain, delay, and slew measurements

<table>
<thead>
<tr>
<th>Liberty Construct</th>
<th>ALF Construct</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timing Type</strong></td>
<td><strong>Timing Sense</strong></td>
</tr>
<tr>
<td><img src="image" alt="input signal" /></td>
<td><img src="image" alt="delay" /></td>
</tr>
<tr>
<td><img src="image" alt="slew" /></td>
<td><img src="image" alt="slew" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="retain" /></td>
<td><img src="image" alt="positive_unate" /></td>
<td>retaining_rise</td>
<td>RETAIN</td>
<td>01 RPN -&gt; 0* PN -&gt; *1 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>retain_rise_slew</td>
<td>SLEWRATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell Rise</td>
<td>DELAY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>rise_transition</td>
<td>SLEWRATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>retaining_fall</td>
<td>RETAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>retain_fall_slew</td>
<td>SLEWRATE</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>cell_fall</td>
<td>RETAIN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_transition</td>
<td>SLEWRATE</td>
<td></td>
</tr>
</tbody>
</table>

| ![negative_unate](image) | ![retain](image) | retaining_rise | RETAIN | 10 RPN -> 1* PN -> *0 PN |
|                         | ![positive_unate](image) | retain_rise_slew | SLEWRATE |        |
|                         | ![positive_unate](image) | cell_Rise | DELAY |        |
|                         | ![positive_unate](image) | rise_transition | SLEWRATE |        |
|                         | ![negative_unate](image) | retaining_fall | RETAIN |        |
|                         | ![negative_unate](image) | retain_fall_slew | SLEWRATE |        |
|                         | ![negative_unate](image) | cell_fall | RETAIN |        |
|                         | ![negative_unate](image) | fall_transition | SLEWRATE |        |

| ![non_unate](image) | ![retain](image) | retaining_rise | RETAIN | ?! RPN -> 0* PN -> *1 PN |
|                     | ![non_unate](image) | retain_rise_slew | SLEWRATE |        |
|                     | ![non_unate](image) | cell_Rise | DELAY |        |
|                     | ![non_unate](image) | rise_transition | SLEWRATE |        |
|                     | ![non_unate](image) | retaining_fall | RETAIN |        |
|                     | ![non_unate](image) | retain_fall_slew | SLEWRATE |        |
|                     | ![non_unate](image) | cell_fall | DELAY |        |
|                     | ![non_unate](image) | fall_transition | SLEWRATE |        |
FIGURE 6. Description of retain, delay, and slew measurements in ALF

VECTOR (VectorExpression) {
   /* ALF_TimingModel */
   RETAIN {
      FROM { PIN = RelatedPinName ; }
      TO { PIN = PinName ; EDGE_NUMBER = 0 ; }
      /* ALF_data */
   }
   SLEWRATE SlewForEdgeNumber0 {
      PIN = PinName ; EDGE_NUMBER = 0 ;
      /* ALF_data */
   }
   DELAY {
      FROM { PIN = RelatedPinName ; }
      TO { PIN = PinName ; EDGE_NUMBER = 1 ; }
      /* ALF_data */
   }
   SLEWRATE SlewForEdgeNumber1 {
      PIN = PinName ; EDGE_NUMBER = 1 ;
      /* ALF_data */
   }
}

1.4 Setup and hold

FIGURE 7. Setup and hold measurements

data signal

setup

hold

clock signal
TABLE 4. Mapping of liberty and ALF constructs for independent setup, hold

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup_rising</td>
<td>?</td>
<td>rise_constraint</td>
<td>SETUP</td>
<td>01 PN -&gt; 01 RPN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint</td>
<td></td>
<td>10 PN -&gt; 01 RPN</td>
</tr>
<tr>
<td>setup_falling</td>
<td>?</td>
<td>rise_constraint</td>
<td>01 PN -&gt; 10 RPN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint</td>
<td>10 PN -&gt; 10 RPN</td>
<td></td>
</tr>
<tr>
<td>hold_rising</td>
<td>?</td>
<td>rise_constraint</td>
<td>HOLD</td>
<td>01 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint</td>
<td></td>
<td>01 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td>hold_falling</td>
<td>?</td>
<td>rise_constraint</td>
<td>10 RPN -&gt; 01 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint</td>
<td>10 RPN -&gt; 10 PN</td>
<td></td>
</tr>
<tr>
<td>non_seq_setup_rising</td>
<td>intrinsic_rise ?</td>
<td>SETUP</td>
<td>01 PN -&gt; 01 RPN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td>10 PN -&gt; 01 RPN</td>
</tr>
<tr>
<td>non_seq_setup_falling</td>
<td>intrinsic_rise ?</td>
<td>01 PN -&gt; 10 RPN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td>10 PN -&gt; 10 RPN</td>
</tr>
<tr>
<td>non_seq_hold_rising</td>
<td>intrinsic_rise ?</td>
<td>HOLD</td>
<td>01 RPN -&gt; 01 PN</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td>01 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td>non_seq_hold_falling</td>
<td>intrinsic_rise ?</td>
<td>10 RPN -&gt; 01 PN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td>10 RPN -&gt; 10 PN</td>
</tr>
</tbody>
</table>

FIGURE 8. Description of independent setup and hold in ALF

VECTOR (VectorExpression) {
/* ALF_TimingModel */
SETUP {
   FROM { PIN = PinName ; }
   TO { PIN = RelatedPinName ; }
   /* ALF_data */
}
}
VECTOR (VectorExpression) {
/* ALF_TimingModel */
HOLD {
   FROM { PIN = RelatedPinName ; }
   TO { PIN = PinName ; }
   /* ALF_data */
}
}
1.5 Recovery and removal

**FIGURE 9. Recovery and removal measurements**

![Diagram](image)

**TABLE 5. Mapping of liberty and ALF constructs for independent recovery, removal**

<table>
<thead>
<tr>
<th>liberty construct</th>
<th>ALF construct</th>
<th>PN = Pin Name, RPN = Related Pin Name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timing Type</strong></td>
<td><strong>Timing Sense</strong></td>
<td><strong>Model Keyword</strong></td>
</tr>
<tr>
<td>recovery_rising</td>
<td>?</td>
<td>rise_constraint ?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint ?</td>
</tr>
<tr>
<td>recovery_falling</td>
<td>?</td>
<td>rise_constraint ?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint ?</td>
</tr>
<tr>
<td>removal_rising</td>
<td>?</td>
<td>rise_constraint ?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint ?</td>
</tr>
<tr>
<td>removal_falling</td>
<td>?</td>
<td>rise_constraint ?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>fall_constraint ?</td>
</tr>
</tbody>
</table>
FIGURE 10. Description of independent recovery and removal in ALF

VECTOR (VectorExpression) {
   /* ALF_TimingModel */
   RECOVERY {
      FROM { PIN = PinName ; }
      TO { PIN = RelatedPinName ; }
      /* ALF_data */
   }
   REMOVAL {
      FROM { PIN = RelatedPinName ; }
      TO { PIN = PinName ; }
      /* ALF_data */
   }
}

1.6 Co-dependent setup and hold

FIGURE 11. Co-dependent setup and hold measurements

TABLE 6. Mapping of liberty and ALF for co-dependent setup, hold

<table>
<thead>
<tr>
<th>liberty construct</th>
<th>ALF construct</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Timing Type</strong></td>
<td><strong>Timing Sense</strong></td>
</tr>
<tr>
<td>setup_rising</td>
<td>?</td>
</tr>
<tr>
<td>hold_rising</td>
<td>?</td>
</tr>
<tr>
<td>setup_rising</td>
<td>?</td>
</tr>
<tr>
<td>hold_rising</td>
<td>?</td>
</tr>
</tbody>
</table>
### TABLE 6. Mapping of liberty and ALF for co-dependent setup, hold

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>ALF construct</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup_falling</td>
<td>?</td>
<td>rise_constraint</td>
<td>SETUP</td>
<td>01 PN -&gt; 10 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td>hold_falling</td>
<td>?</td>
<td>fall_constraint</td>
<td>HOLD</td>
<td></td>
</tr>
<tr>
<td>setup_falling</td>
<td>?</td>
<td>fall_constraint</td>
<td>SETUP</td>
<td>10 PN -&gt; 10 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td>hold_falling</td>
<td>?</td>
<td>rise_constraint</td>
<td>HOLD</td>
<td></td>
</tr>
</tbody>
</table>

### FIGURE 12. Description of co-dependent setup and hold in ALF

```
VECTOR (VectorExpression) {  
/* ALF_TimingModel */
   SETUP {  
      FROM { PIN = PinName ; EDGE_NUMBER = 0 ; }
      TO { PIN = RelatedPinName ; }
      /* ALF_data */
   }  
   HOLD {  
      FROM { PIN = RelatedPinName ; }
      TO { PIN = PinName ; EDGE_NUMBER = 1 ; }
      /* ALF_data */
   }
}
```

### 1.7 Co-dependent recovery and removal

### FIGURE 13. Co-dependent recovery and removal measurements

- **asynchronous signal**
- **active**
- **inactive**
- **clock signal**
- **recovery**
- **removal**
TABLE 7. Mapping of liberty and ALF for co-dependent recovery, and removal

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>recovery_rising</td>
<td>?</td>
<td>rise_constraint ?</td>
<td>RECOVERY</td>
<td>01 PN &lt;&amp;&gt; 01 RPN</td>
</tr>
<tr>
<td>removal_rising</td>
<td>?</td>
<td>rise_constraint ?</td>
<td>REMOVAL</td>
<td></td>
</tr>
<tr>
<td>recovery_rising</td>
<td>?</td>
<td>fall_constraint ?</td>
<td>RECOVERY</td>
<td>10 PN &lt;&amp;&gt; 01 RPN</td>
</tr>
<tr>
<td>removal_rising</td>
<td>?</td>
<td>fall_constraint ?</td>
<td>REMOVAL</td>
<td></td>
</tr>
<tr>
<td>recovery_falling</td>
<td>?</td>
<td>rise_constraint ?</td>
<td>RECOVERY</td>
<td>01PN &lt;&amp;&gt; 10 RPN</td>
</tr>
<tr>
<td>removal_falling</td>
<td>?</td>
<td>rise_constraint ?</td>
<td>REMOVAL</td>
<td></td>
</tr>
<tr>
<td>recovery_falling</td>
<td>?</td>
<td>fall_constraint ?</td>
<td>RECOVERY</td>
<td>10 PN &lt;&amp;&gt; 10 RPN</td>
</tr>
<tr>
<td>removal_falling</td>
<td>?</td>
<td>fall_constraint ?</td>
<td>REMOVAL</td>
<td></td>
</tr>
</tbody>
</table>

VECTOR (VectorExpression) {
   /* ALF_TimingModel */
   RECOVERY {
      FROM { PIN = PinName ; }
      TO { PIN = RelatedPinName ; }
      /* ALF_data */
   }
   REMOVAL {
      FROM { PIN = RelatedPinName ; }
      TO { PIN = PinName ; }
      /* ALF_data */
   }
}

1.8 Setup and hold with nochange
FIGURE 15. Setup and hold measurements with nochange constraint

![Diagram showing setup and hold measurements with nochange constraint]

TABLE 8. Mapping of liberty and ALF for setup and hold with nochange constraint

<table>
<thead>
<tr>
<th>liberty construct</th>
<th>ALF construct</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>Timing Type</strong></td>
</tr>
<tr>
<td>nochange_high_high</td>
<td>SETUP</td>
</tr>
<tr>
<td></td>
<td>HOLD</td>
</tr>
<tr>
<td>nochange_high_low</td>
<td>SETUP</td>
</tr>
<tr>
<td></td>
<td>HOLD</td>
</tr>
<tr>
<td>nochange_low_high</td>
<td>SETUP</td>
</tr>
<tr>
<td></td>
<td>HOLD</td>
</tr>
<tr>
<td>nochange_low_low</td>
<td>SETUP</td>
</tr>
<tr>
<td></td>
<td>HOLD</td>
</tr>
</tbody>
</table>
1.9 Maximum skew constraint

**FIGURE 17. Maximum skew constraint**

```
VECTOR (VectorExpression) {
    /* ALF_TimingModel */
    SETUP {
        FROM { PIN = PinName ; EDGE_NUMBER = 0 ; }
        TO { PIN = RelatedPinName ; EDGE_NUMBER = 0 ; }
        /* ALF_data */
    }
    HOLD {
        FROM { PIN = RelatedPinName ; EDGE_NUMBER = 1 ; }
        TO { PIN = PinName ; EDGE_NUMBER = 1 ; }
        /* ALF_data */
    }
    NOCHANGE {
        FROM { PIN = RelatedPinName ; EDGE_NUMBER = 0 ; }
        TO { PIN = RelatedPinName ; EDGE_NUMBER = 1 ; }
    }
}
```
TABLE 9. Mapping of liberty and ALF constructs for maximum skew constraint

<table>
<thead>
<tr>
<th>Timing Type</th>
<th>Timing Sense</th>
<th>Model Keyword</th>
<th>Keyword</th>
<th>Vector Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>skew_rising</td>
<td>?</td>
<td>intrinsic_rise ?</td>
<td>SKEW</td>
<td>01 RPN -&gt; 01 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td></td>
</tr>
<tr>
<td>skew_falling</td>
<td>?</td>
<td>intrinsic_rise ?</td>
<td>10 RPN</td>
<td>01 RPN -&gt; 10 PN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>intrinsic_fall ?</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FIGURE 18. Description of maximum skew constraint in ALF

VECTOR (VectorExpression) {
/* ALF_TimingModel */
 LIMIT {
 SKEW {
 PIN { PinName RelatedPinName } // pin order is irrelevant here
 MAX {
 /* ALF_data */
 }
}
}
}

1.10 Minimum period and minimum pulsewidth constraints

FIGURE 19. Minimum period and minimum pulsewidth constraints
TABLE 10. Mapping of liberty and ALF constructs for minimum period and minimum pulsewidth constraints

<table>
<thead>
<tr>
<th>liberty construct</th>
<th>ALF construct</th>
<th>Vector Expression</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>min_period</td>
<td>PERIOD</td>
<td>01 PN</td>
<td>for positive edge triggered clock</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10 PN</td>
<td>for negative edge triggered clock</td>
</tr>
<tr>
<td>min_pulse_width_high</td>
<td>PULSEWIDTH</td>
<td>01 PN -&gt; 10 PN</td>
<td>01 PN -&gt; 10 PN</td>
</tr>
<tr>
<td>min_pulse_width_low</td>
<td></td>
<td>10 PN -&gt; 01 PN</td>
<td>10 PN -&gt; 01 PN</td>
</tr>
</tbody>
</table>

**min_period or minimum_period (inconsistency in liberty ref manual)?**

FIGURE 20. Description of minimum period constraint in ALF

```plaintext
VECTOR (VectorExpression) {
/* ALF_TimingModel */
LIMIT {
    PERIOD {
        MIN { /* ALF_data */ } 
    }
}
}
```

FIGURE 21. Description of minimum pulsewidth constraint in ALF

```plaintext
VECTOR (VectorExpression) {
/* ALF_TimingModel */
LIMIT {
    PULSEWIDTH {
        PIN = PinName ;
        MIN { /* ALF_data */ } 
    }
}
}
```
1.11 Threshold definitions

Do the input thresholds always apply for the related pin and the output thresholds for the parent pin or do the applicable thresholds depend on the direction of the pin (not clear in case of setup, hold, recovery, removal)?

The thresholds for delay and slew measurements in liberty are normalized values between 0 and 100, to be interpreted as percentage values. The corresponding thresholds in ALF are normalized values between 0 and 1. Therefore, the conversion involves either dividing liberty data by 100 or multiplying ALF data by 100.

The slew data in the library are understood to be the measured values according to the slew threshold definitions. However, the slew data might be represented in a normalized way, for example scaled from rail-to-rail. In order to allow for such a normalized representation, a scaling factor can be defined. The slew data multiplied with the scaling factor is then understood to be the measured values according to the slew threshold definitions. In liberty, the keyword slew_derate_from_library defines the scaling factor. The scaling factor multiplied with the base unit defines the absolute slew data. In ALF, the UNIT annotation defines the multiplier, i.e., the product of scaling factor and base unit.

To make the differences between liberty and ALF clearer, numerical values are shown in the following Figure 23 on page 19 and Figure 24 on page 19.
**FIGURE 23. Liberty description of library threshold definitions**

```plaintext
/* liberty */
library (LibraryName) {
    time_unit : "1ns" ;
    input_threshold_pct_rise : 45 ;
    input_threshold_pct_fall : 55 ;
    output_threshold_pct_rise : 35 ;
    output_threshold_pct_fall : 65 ;
    slew_lower_threshold_pct_rise : 30 ;
    slew_upper_threshold_pct_rise : 50 ;
    slew_upper_threshold_pct_fall : 70 ;
    slew_lower_threshold_pct_fall : 50 ;
    slew_derate_from_library : 0.2 ;
}
```

**FIGURE 24. ALF description of library threshold definitions**

```plaintext
/* ALF */
LIBRARY LibraryName {
    TIME { UNIT = 1e-9 ; }
    DELAY {
        FROM {
            THRESHOLD {
                RISE = 0.45 ;
                FALL = 0.55 ;
            };
        }
        TO {
            THRESHOLD {
                RISE = 0.35 ;
                FALL = 0.65 ;
            };
        }
    }
    SLEWRATE {
        UNIT = 0.2e-9 ;
        FROM {
            THRESHOLD {
                RISE = 0.3 ;
                FALL = 0.7 ;
            };
        }
        TO {
            THRESHOLD {
                RISE = 0.5 ;
                FALL = 0.5 ;
            };
        }
    }
}
```
According to this example, a numerical slew value of i1i really means 0.2ns, measured from 30% to 50% for rising transition and from 70% to 50% for falling transition, respectively.

1.12 Conditional timing arcs

The *existence condition* for a timing arc is the necessary and sufficient condition for a timing arc to be activated. A *value condition* is a sufficient condition.

Mathematically, the existence condition can be expressed as a boolean expression in a sum-of-product form.

For example, a timing arc from input A to output Y can be activated, if the existence condition \((E1|E2)\) is satisfied, where \(E1\) and \(E2\) are side inputs. The sum-of-product form of the existence condition reads as follows:

\[
E1 \mid E2 = E1 \& E2 \mid E1 \& !E2 \mid !E1 \& E2
\]

The delay from A to Y depends possibly on the state of \(E1\) and \(E2\). The value condition is a particular state for which a particular value applies. It can be either \((E1\&E2)\) or \((E1\&!E2)\) or \((!E1\&E2)\).

In liberty, the *value condition* is expressed in a *when* statement. In ALF, the value condition is expressed as a co-factor within the vector expression.

In liberty, the *existence condition* can not be described explicitly. However, the existence condition can be inferred either by evaluation of the *when* statement or by combining all the *when* statements of all timing groups with same pin, same related pin, same timing_type and same timing_sense. The same inference can be applied to ALF. However, ALF supports also an explicit statement for existence condition.
A `i when_start` and a `i when_end` statement in liberty means that the condition is checked at the time of the FromPin event and the ToPin event, respectively.

In ALF, these conditions are described as co-factors in the vector expression.

**FIGURE 25. Conditional timing and existence condition example in liberty and ALF**

```c
/* liberty */
pin(Y) {
  timing() {
    timing_type : combinational;
    timing_sense : positive_unate;
    related_pin : "A";
    when : "E1&E2";
    cell_rise ... 
    rise_transition ...
  }
  timing() {
    timing_type : combinational;
    timing_sense : positive_unate;
    related_pin : "A";
    when : "E1!E2";
    cell_rise ... 
    rise_transition ...
  }
  timing() {
    timing_type : combinational;
    timing_sense : positive_unate;
    related_pin : "A";
    when : "!E1&E2";
    cell_rise ... 
    rise_transition ...
  }
}
/* inferred existence condition:
   E1&E2 | E1!E2 | !E1&E2 */
```

```c
/* ALF */
VECTOR ((01 A -> 01 Y)&(E1&E2)) {
  EXISTENCE_CONDITION
  = E1&E2 | E1!E2 | !E1&E2 ;
  DELAY ...
  SLEWRATE ...
}
VECTOR ((01 A -> 01 Y)&(E1!E2)) {
  EXISTENCE_CONDITION
  = E1&E2 | E1!E2 | !E1&E2 ;
  DELAY ...
  SLEWRATE ...
}
VECTOR ((01 A -> 01 Y)&(!E1&E2)) {
  EXISTENCE_CONDITION
  = E1&E2 | E1!E2 | !E1&E2 ;
  DELAY ...
  SLEWRATE ...
}
```

**FIGURE 26. Timing with start and end condition in liberty and ALF**

```c
/* liberty */
pin(Y) {
  timing() {
    timing_type : combinational;
    timing_sense : positive_unate;
    related_pin : "A";
    when_start : "E1";
    when_end : "E2";
    cell_rise ...
    rise_transition ...
  }
}
```

```c
/* ALF */
VECTOR ((01 A -> 01 Y) & E1 ~> (01 Y) & E2) {
  DELAY ...
  SLEWRATE ...
}
```
1.13 Timing arcs involving bus pins

*Fill in liberty examples*

**FIGURE 27.** Timing arc on a bus with bit-to-bit extension in ALF

```liberty
CELL CellName {
    GROUP DataBit { 1 : 8 }
    PIN [1:8] DataBusIn { DIRECTION = input ; }
    PIN [1:8] DataBusOut { DIRECTION = output ; }
    VECTOR ( 01 DataBusIn[DataBit] -> 01 DataBusOut[DataBit] ) {
        DELAY = 1.0 {
            FROM { PIN = DataBusIn[DataBit] ; }
            TO { PIN = DataBusOut[DataBit] ; }
        }
    }
}
```

**FIGURE 28.** Timing arc on a bus with all-to-all extension in ALF

```liberty
CELL CellName {
    GROUP AddressBit { 0 : 3 }
    GROUP DataBit { 1 : 8 }
    PIN [3:0] AddressBus { DIRECTION = input ; }
    PIN [1:8] DataBusOut { DIRECTION = output ; }
    VECTOR ( 01 AddressBus[AddressBit] -> 01 DataBusOut[DataBit] ) {
        DELAY = 1.0 {
            FROM { PIN = AddressBus[AddressBit] ; }
            TO { PIN = DataBusOut[DataBit] ; }
        }
    }
}
```
2.0 Interoperability with SDF

TABLE 11.

<table>
<thead>
<tr>
<th>SDF construct</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATHPULSE</td>
<td></td>
</tr>
<tr>
<td>PATHPULSEPERCENT</td>
<td></td>
</tr>
<tr>
<td>ABSOLUTE</td>
<td></td>
</tr>
<tr>
<td>INCREMENT</td>
<td></td>
</tr>
<tr>
<td>IOPATH</td>
<td>delay measurement, see Table 2 on page 5, Figure 4 on page 6</td>
</tr>
<tr>
<td>RETAIN</td>
<td>see Table 3 on page 7, Figure 6 on page 8</td>
</tr>
<tr>
<td>COND</td>
<td></td>
</tr>
<tr>
<td>CONDELS</td>
<td></td>
</tr>
<tr>
<td>PORT</td>
<td></td>
</tr>
<tr>
<td>INTERCONNECT</td>
<td></td>
</tr>
<tr>
<td>NETDELAY</td>
<td></td>
</tr>
<tr>
<td>DEVICE</td>
<td></td>
</tr>
<tr>
<td>SETUP</td>
<td>see Table 4 on page 9, Figure 10 on page 11</td>
</tr>
<tr>
<td>HOLD</td>
<td>see Table 4 on page 9, Figure 10 on page 11</td>
</tr>
<tr>
<td>SETUPHOLD</td>
<td>see Table 6 on page 11, Figure 12 on page 12</td>
</tr>
<tr>
<td>RECOVERY</td>
<td>see Table 4 on page 9, Figure 10 on page 11</td>
</tr>
<tr>
<td>REMOVAL</td>
<td>see Table 4 on page 9, Figure 10 on page 11</td>
</tr>
<tr>
<td>RECREM</td>
<td>see Table 6 on page 11, Figure 12 on page 12</td>
</tr>
<tr>
<td>SKEW</td>
<td>see Table 9 on page 16, Figure 18 on page 16</td>
</tr>
<tr>
<td>BIDIRECTSKEW</td>
<td></td>
</tr>
<tr>
<td>WIDTH</td>
<td>see Table 10 on page 17, Figure 21 on page 17</td>
</tr>
<tr>
<td>PERIOD</td>
<td>see Table 10 on page 17, Figure 20 on page 17</td>
</tr>
<tr>
<td>NOCHANGE</td>
<td>see Table 8 on page 14, Figure 16 on page 15</td>
</tr>
<tr>
<td>SCOND</td>
<td></td>
</tr>
<tr>
<td>CCOND</td>
<td></td>
</tr>
<tr>
<td>LABEL</td>
<td></td>
</tr>
</tbody>
</table>

Conditions in SDF are expressed in Verilog syntax, which is different from Liberty syntax. Therefore, liberty provides \texttt{i SDF\_cond\_i}, \texttt{i SDF\_cond\_start\_i}, \texttt{i SDF\_cond\_end\_i} statements, which are basically \texttt{i when\_i}, \texttt{i when\_start\_i}, \texttt{i when\_end\_i} statements translated into Verilog syntax.
The ALF syntax for conditions closely matches the Verilog syntax. Therefore, `i SDF_cond`, `i SDF_cond_start`, `i SDF_cond_end` are not provided as standard annotations in ALF. However, if desired, they can be defined as library-specific annotations in the following way:

```alibaba
KEYWORD SDF_cond = single_value_annotation {
  VALUETYPE  = quoted_string;
  CONTEXT    = VECTOR;
}
KEYWORD SDF_cond_start = single_value_annotation {
  VALUETYPE  = quoted_string;
  CONTEXT    = VECTOR;
}
KEYWORD SDF_cond_end = single_value_annotation {
  VALUETYPE  = quoted_string;
  CONTEXT    = VECTOR;
}
```