Semantic Interpretations of Arithmetic Models in ALF for Use in OLA

Wolfgang Roethig

Oct. 21, 1998

The purpose of this document is to facilitate the translation and interpretation of ALF arithmetic models in OLA. In particular, the document should be useful to establish a mapping between ALF constructs and corresponding DCL constructs. The ALF constructs are 1.0, unless indicated indicated differently.

Table of contents

1.0 Summary of arithmetic models

The summary is given in the following form:

<model_keyword_in_uppercase> (value): explanation

1.1 Interpolatable Arithmetic Models

1.1.1 Timing measurements

DELAY (number): time between two threshold crossing within two consecutive events on two pins

SLEWRATE (non-negative number): time between two threshold crossings within one event on one pin

RETAIN[1] (number): time between two threshold crossing within two consecutive events on two pins, in conjuntion with DELAY measurement

1.1.2 Timing constraints

SETUP, HOLD, RECOVERY, REMOVAL (number): minimum time limit between two threshold crossings within two consecutive events on two pins

SKEW (number): maximum time limit between two threshold crossings within two consecutive events on two pins

PULSEWIDTH (non-negative number): minimum time limit between two threshold crossings within two consecutive and complementary events on one pin

PERIOD (non-negative number): minimum time limit between two identical events within a sequence of periodical events on one pin

NOCHANGE: (optional non-negative number): minimum limit between two threshold crossings within two arbitrary consecutive events on one pin

1.1.3 Analog measurements

CURRENT, VOLTAGE, POWER, ENERGY, TEMPERATURE, FREQUENCY, TIME (number)

1. proposed for ALF 1.1

1.1.4 Electrical components

RESISTANCE, CAPACITANCE (non-negative number)

INDUCTANCE[1] (non-negative number)

1.1.5 Layout measurements

LENGTH, WIDTH, HEIGHT, AREA (non-negative number)

DISTANCE (number)

1.1.6 Abstract cost function measurements

SIZE[2] (non-negative number)

1.1.7 normalized voltage measurements

THRESHOLD (0 < number < 1)

NOISE_MARGIN[3] (0 < number < 1)

1.1.8 discrete measurements

FANIN, FANOUT, CONNECTIONS, SWITCHING_BITS (unsigned, interpolatable to non-negative number)

1.2 Non-Interpolatable Arithmetic Models

1.2.1 Models indexed by identifiers which can be associated with measureable numbers

PROCESS (nom, snsp, snwp, wnsp, wnwp or custom identifier)

DERATE_CASE (nom, bccom, wccom, bcind, wcind, bcmil, wcmil or custom identifier)

any keyword for interpolatable arithmetic model (predefined identifier as for PROCESS, DERATE_CASE or custom identifier)

1.2.2 Models indexed by identifiers which have only symbolic meaning

DRIVER (custom identifier)

1. proposed for ALF 1.1
2. proposed for ALF 1.1
3. proposed for ALF 1.1

RECEIVER (custom identifier)

1.2.3 Models indexed by boolean literals

CONNECTIVITY (0, 1, ?)

2.0 Description of Arithmetic Models

2.1 Structure of Arithmetic Model

An arithmetic model may be a single number or a set of numbers within nested arithmetic models. Unless details are relevant for explanation, we use the following notation for a general arithmetic model:

```
<model_keyword> /*data*/
```

The outermost arithmetic model of a nested arithmetic model is henceforth called "arithmetic model root".

2.1.1 HEADER, TABLE, EQUATION

A HEADER contains arithmetic model(s) inside arithmetic model. The inside models are the arguments of the outside model. An arithmetic model containing a HEADER must contain data in a TABLE or an EQUATION. It may also contain both TABLE and EQUATION, in which case the EQUATION is only used outside the TABLE boundaries. Without EQUATION, the TABLE data is used for both interpolation and extrapolation.

```
<model_keyword> {
    HEADER { <model_keyword> /*data*/  .. <model_keyword> /*data*/  }
    TABLE { <numbers> }
}
```

or

```
<model_keyword> {
    HEADER { <model_keyword> /*data*/  .. <model_keyword> /*data*/  }
    EQUATION { <arithmetic_expression > }
}
```

or

```
<model_keyword> {
    HEADER { <model_keyword> /*data*/  .. <model_keyword> /*data*/  }
    TABLE { <numbers> }
    EQUATION { <arithmetic_expression > }
}
```

Nesting of arithmetic models with HEADER is allowed in the following way:

Interpolatable arithmetic model may contain TABLE or EQUATION. The HEADER inside interpolatable arithmetic models may contain interpolatable arithmetic models or models indexed by identifiers that can be associated with numbers.

Models indexed by identifiers and models indexed by boolean literals may contain TABLE. The HEADER of those models may contain identifiers or models indexed by identifiers.

Multiple nesting is allowed.

If the outer arithmetic model root contains TABLE, each model in the next inner HEADER must also contain TABLE. The number of indices in the outer TABLE must be the product of numbers of indices in the inner TABLEs.

2.2 Containers of Arithmetic Models

Containers of arithmetic models change or restrict the semantic interpretation of the arithmetic models. The modeling idea behind the concept of containers is, that the semantic meaning of the container is more global than the semantic meaning of the model iteself.

2.2.1 LIMIT

Container of arithmetic model root(s), each of which must contain a MIN or a MAX model.

Used for models of design or application limits.

LIMIT { <model_keyword> { MIN /*data*/ MAX /*data*/ }

Applicable to all interpolatable arithmetic models with the exception of timing constraints, since those are already defined as limits.

2.2.2 EARLY, LATE[1]

Container of arithmetic model root.

Defines the boundaries of timing measurements in one single analysis.

Only applicable to DELAY and SLEWRATE. Both of them must appear in both containers.

The quadrupel

EARLY { DELAY /*data*/ SLEWRATE /*data*/ }

LATE { DELAY /*data*/ SLEWRATE /*data*/ }

1. proposed for ALF 1.1

is used to calculate the envelope of the timing waveform at the TO point of a delay arc with respect to the timing waveform at the FROM point of a delay arc.

The EARLY DELAY is of course a smaller number (or a set of smaller numbers) than the LATE DELAY. However, the EARLY SLEWRATE is not necessarily smaller than the LATE SLEWRATE, since the SLEWRATE of the EARLY signal may be larger than the SLEWRATE of the LATE signal.

2.3 Qualifiers of Arithmetic Models[1]

The qualifiers *do not contain* the arithmetic model, they *become* the arithmetic model whereas the original keyword of the model eventually becomes a keyword for a model container. The modeling idea behind the concept of qualifiers is, that the semantic meaning of the model is more global than the semantic meaning of the qualifier.

2.3.1 MIN, TYP, MAX within the scope of arithmetic model root

They contain 3 distinct sets of data for MIN, TYP, MAX

```
<model_keyword> { MIN /*data*/ TYP /*data*/ MAX /*data*/ }
```

as opposed to a single set of data

```
<model_keyword> /*data*/
```

The arithmetic model root containing MIN, TYP, MAX must not contain HEADER or TABLE or EQUATION. Instead, the MIN, TYP, MAX models may contain HEADER or TABLE or EQUATION.

```
<model_keyword> {
    MIN {
        HEADER{ <model_keyword> /*data*/  .. <model_keyword> /*data*/ }
        TABLE /* or equation */ { <numbers> }
    }
    TYP {
        HEADER{ <model_keyword> /*data*/  .. <model_keyword> /*data*/ }
        TABLE /* or equation */ { <numbers> }
    }
    MAX {
        HEADER{ <model_keyword> /*data*/  .. <model_keyword> /*data*/ }
        TABLE /* or equation */ { <numbers> }
    }
}
```

1. proposed for ALF 1.1 as full arithmetic models, supported in ALF 1.0 as annotations only

The MIN, TYP, MAX represent a statistical distribution of data without specifying or implying a particular cause of the distribution. If process corners or derate cases are not modeled explicitly, MIN, TYP, MAX can be used for representing the distribution of data across processes or derate cases. If process corners or delay cases are modeled explicitly, MIN, TYP, MAX can be used for representing the distribution of data within each process corner or derate case.

2.3.2 MIN, MAX within the scope of arithmetic model root inside LIMIT

They contain the data for a lower or upper limit, respectively. There must be at least one limit, lower or upper, in each model, but not necessarily both, as shown in the example below.

```
LIMIT  {
    <model_keyword1> { MIN /*data*/  } // lower limit
    <model_keyword2> { MAX /*data*/  }// upper limit
    <model_keyword3> { MIN /*data*/ MAX /*data*/  }// lower and upper limit
}
```

The arithmetic model root inside LIMIT must not contain HEADER or TABLE or EQUATION. Instead, the MIN or MAX models may contain HEADER or TABLE or EQUATION.

```
LIMIT {
    <model_keyword> {
        MIN {
                HEADER{ <model_keyword> /*data*/  .. }
                TABLE /* or equation */ { <numbers> }
        }
        MAX {
                HEADER{ <model_keyword> /*data*/  .. }
                TABLE /* or equation */ { <numbers> }
        }
    }
}
```

2.3.3 MIN, MAX within the scope of arithmetic model inside HEADER

MIN, MAX inside a model inside a HEADER define the validity limits of the data. The model inside the HEADER may contain TABLE or EQUATION. It may also contain HEADER, which represents a nested arithmetic model.

```
<model_keyword> {
    HEADER {
        <model_keyword> {
```

```
            MIN /*data*/
            MAX /*data*/
            TABLE { <numbers> }
        }
    }
    TABLE { <numbers> }
}
```

The MIN, MAX data may be lumped numbers or they may be nested arithmetic models themselves. No dependency or correlation is implied or infered between the calculation of the MIN, MAX data and the calculation of the model data inside or outside the HEADER.

If MIN, MAX is not defined and the data is in a TABLE, the boundaries of the data in the TABLE shall be considered as validity limits.

2.3.4 RISE, FALL

Only inside an arithmetic model root. RISE, FALL contain data for transient measurements.

```
<model_keyword> { RISE /*data*/ FALL /*data*/ }
```

It is generally not required that both RISE and FALL appear in the arithmetic model root.

The arithmetic model root containing RISE, FALL must not contain MIN, TYP, MAX, HEADER, TABLE or EQUATION. Instead, the RISE, FALL models may either contain HEADER, TABLE, EQUATION or contain MIN, TYP, MAX which may contain HEADER, TABLE, EQUATION themselves.

```
<model_keyword> {
    RISE {
        HEADER{ <model_keyword> /*data*/  .. }
        TABLE /* or equation */ { <numbers> }
    }
    FALL {
        HEADER{ <model_keyword> /*data*/  .. }
        TABLE /* or equation */ { <numbers> }
    }
}
```

or

```
<model_keyword> {
    RISE {
        MIN /*data*/
        TYP /*data*/
```

```
        MAX /*data*/
    }
    FALL {
        MIN /*data*/
        TYP /*data*/
        MAX /*data*/
    }
}
```

Semantic meaning for RISE and FALL is provided for the following measurements:

- DELAY:

RISE, FALL is the switching direction on the PIN specified in the TO field.

If the TO field does not exist (a special case for port delay), RISE, FALL is the switching direction on the PIN specified in the FROM field.

- CAPACITANCE, RESISTANCE, INDUCTANCE, CURRENT, ENERGY, POWER, SLEWRATE, THRESHOLD:

RISE, FALL is the switching direction on the PIN. Either the PIN is specified as annotation inside the model, or the model is inside a PIN.

The arithmetic model root containing RISE, FALL may be inside a LIMIT container with the following rule:

- A model containing RISE or FALL must not contain MIN or MAX. Instead, the RISE or FALL model must contain MIN or MAX.

```
LIMIT {
    <model_keyword> {
        RISE { MIN /*data*/ MAX /*data*/ }
        FALL { MIN /*data*/ MAX /*data*/ }
    }
}
```

The arithmetic model root containing RISE, FALL may be inside EARLY, LATE containers with the following rules:

- If only RISE appears in one model, only RISE must appear in all models.
- If only FALL appears in one model, only FALL must appear in all models.
- If both RISE and FALL appear in one model, both RISE and FALL must appear in all models.

```
EARLY {
```

```
    DELAY { RISE /*data*/ FALL /*data*/ }
    SLEWRATE { RISE /*data*/ FALL /*data*/ }
    }
LATE {
    DELAY { RISE /*data*/ FALL /*data*/ }
    SLEWRATE { RISE /*data*/ FALL /*data*/ }
}
```

Semantic meaning for RISE and FALL is provided for the following LIMIT specifications, EARLY or LATE measurements:

- DELAY:

RISE, FALL is the switching direction on the PIN specified in the TO field.

Only if the TO field does not exist (a special case for port delay), RISE, FALL is the switching direction on the PIN specified in the FROM field (since the switching direction of the unspecified PIN in the TO field will be the same).

- SLEWRATE:

RISE, FALL is the switching direction on the PIN. Either the PIN is specified as annotation inside the model, or the model is inside a PIN.

2.3.5 HIGH, LOW[1]

Only inside an arithmetic model root. HIGH, LOW contain data for static measurements.

```
    <model_keyword> {HIGH /*data*/ LOW /*data*/ }
```

It is generally not required that both HIGH and LOW appear in the arithmetic model root.

The arithmetic model root containing HIGH, LOW must not contain MIN, TYP, MAX, HEADER, TABLE or EQUATION. Instead, the RISE, FALL models may either contain HEADER, TABLE, EQUATION or contain MIN, TYP, MAX which may contain HEADER, TABLE, EQUATION themselves.

```
<model_keyword> {
    HIGH {
        HEADER{ <model_keyword> /*data*/  .. }
        TABLE /* or equation */ { <numbers> }
    }
    LOW {
        HEADER{ <model_keyword> /*data*/  .. }
```

1. keywords proposed for ALF 1.1

```
        TABLE /* or equation */ { <numbers> }
    }
}
```

or

```
<model_keyword> {
    HIGH {
        MIN /*data*/
        TYP /*data*/
        MAX /*data*/
    }
    LOW {
        MIN /*data*/
        TYP /*data*/
        MAX /*data*/
    }
}
```

Semantic meaning for HIGH and LOW is provided for the following measurements:

- CAPACITANCE, RESISTANCE, INDUCTANCE, CURRENT, ENERGY, POWER, VOLTAGE, NOISE_MARGIN:

HIGH, LOW is the state on the PIN. Either the PIN is specified as annotation inside the model, or the model is inside a PIN.

The arithmetic model root containing HIGH, LOW may be inside a LIMIT container with the following rule:

- A model containing HIGH or LOW must not contain MIN or MAX. Instead, the HIGH or LOW model must contain MIN or MAX.

```
LIMIT {
    <model_keyword> {
        HIGH { MIN /*data*/ MAX /*data*/ }
        LOW { MIN /*data*/ MAX /*data*/ }
    }
}
```

Semantic meaning for HIGH and LOW is provided for the following LIMIT specifications:

- CURRENT, ENERGY, POWER, VOLTAGE

HIGH, LOW is the state on the PIN. Either the PIN is specified as annotation inside the model, or the model is inside a PIN.

2.4 Annotations and Annotation Containers for Arithmetic Models

Annotations and annotation containers described in this chapter are relevant for the semantic interpretation of the arithmetic models. Annotations and annotation containers not described in this chapters can be interpreted as simple properties or attributes of the arithmetic models.

2.4.1 FROM, TO annotation container

Annotation container used inside arithmetic models for timing measurement and timing constraints.

If said models apply for two pins, the FROM, TO containers shall each contain the PIN annotation. These annotations shall define the sense of measurement.

```
<model_keyword> {
    FROM { PIN = <pin_name> ; }
    TO { PIN = <pin_name> ; }
    /* data */
}
```

Otherwise, if said models apply only for one pin, the same PIN annotation may be repeated in both containers or the PIN annotation may be outside the FROM, TO container.

```
<model_keyword> {
    PIN = <pin_name> ;
    /* data */
}
```

If thresholds are needed for exact definition of the model data, the FROM, TO containers shall each contain the annotation or an arithmetic model[1] for THRESHOLD.

```
<model_keyword> {
    FROM { THRESHOLD /*data*/ }
    TO { THRESHOLD /*data*/ }
    /* data */
}
```

An annotation or arithmetic model for THRESHOLD outside a FROM or TO container shall only have a semantic meaning, if said annotation or arithmetic model contains a PIN annotation itself and this PIN annotation matches a PIN annotation in a FROM or TO container.

1. arithmetic model inside FROM, TO proposed for ALF 1.1, annotation supported in ALF 1.0

Example:

```
DELAY {
    FROM {
        PIN = pin1;
        THRESHOLD /*data*/
    }
    TO {
        PIN = pin2;
    }
    HEADER {
        THRESHOLD {
                PIN = pin2;
                TABLE { <numbers> }
        }
    TABLE { <numbers> }
}
```

Note: The data of the THRESHOLD at pin1 is calculated independently of DELAY, whereas DELAY is calucated as a function of THRESHOLD at pin2.

2.4.2 PIN annotation

The use of PIN annotation in arithmetic models other than timing measurements and timing constraints is defined here.[1]

If the PIN annotation appears inside an arithmetic model within the scope of a HEADER or a LIMIT, the physcical quantity identified by the model keyword is *applied* to the PIN. Otherwise, if the PIN annotation appears inside an arithmetic model root which is not within the scope of a LIMIT, the physical quantity identified by the model keyword is *measured* at the PIN.

Example:

```
// intrinsic capacitance of pin1
CAPACITANCE {
    PIN = pin1;
    /*data*/
}
// maximum allowed capacitance on a net connected to pin2
LIMIT {
    CAPACITANCE {
```

1. amendment of ALF 1.0 semantics

```
            PIN = pin2;
            MAX /*data*/
        }
}
// delay measured as function of capacitance on a net connected to pin3
DELAY {
    HEADER {
        CAPACITANCE {
                PIN = pin3;
        }
    }
    /*data*/
}
```

If the arithmetic model is within the scope of a PIN object, a PIN annotation is illegal according to the visibility rules of ALF, since a PIN cannot be visible inside another PIN, with the following exception: The PIN outside the arithmetic model is a bus, and the PIN annotation inside the arithmetic model refers to a bit of the bus.

Example:

```
PIN [1:2] bus_pin {
// intrinsic capacitance of bus_pin[1]
    CAPACITANCE {
        PIN = bus_pin[1];
        /*data*/
    }
// maximum allowed capacitance on a net connected to bus_pin[2]
    LIMIT {
        CAPACITANCE {
                PIN = bus_pin[2];
                /*data*/
        }
    }
}
```

If an arithmetic model root appears within the scope of a LIMIT inside a PIN, the physcical quantity identified by the model keyword is *applied* to the PIN. Otherwise, if an arithmetic model root appears directly inside a PIN, the physical quantity identified by the model keyword is *measured* at the PIN.

Example:

```
PIN scalar_pin {
// intrinsic capacitance of scalar_pin
    CAPACITANCE {
        /*data*/
    }
// maximum allowed capacitance on a net connected to scalar_pin
    LIMIT {
        CAPACITANCE {
            /*data*/
        }
    }
}
```