

Library Harmonization for Power

	Version	Date
	0.0	2004/03/22
I	0.1	2004/05/01

Table of contents

1.0	Electrical Power	3
1.1	Leakage power	3
1.2	Switching power	4
1.3	Internal power	6
1.4	Power with state-dependency	8
1.5	Power involving a bus or a bundle of pins	9
1.6	Power involving a switching interval	14
1.7	Power involving functionally equivalent output pins	14
2.0	Multiple power supplies	15

1.0 Electrical Power

For CMOS circuits, power can be divided into subcategories: leakage, switching, internal.

- Switching power is associated with charge/discharge of external load capacitance. As it is a transient phenomenon, it is measured in units of energy rather than in units of power.
- Internal power is associated with short-circuit current and charge/discharge of internal capacitances. As it is a transient phenomenon, it is measured in units of energy rather than in units of power.
- Leakage power is dependent on transistor threshold voltages. As it is a static phenomenon, it is measured in units of power.

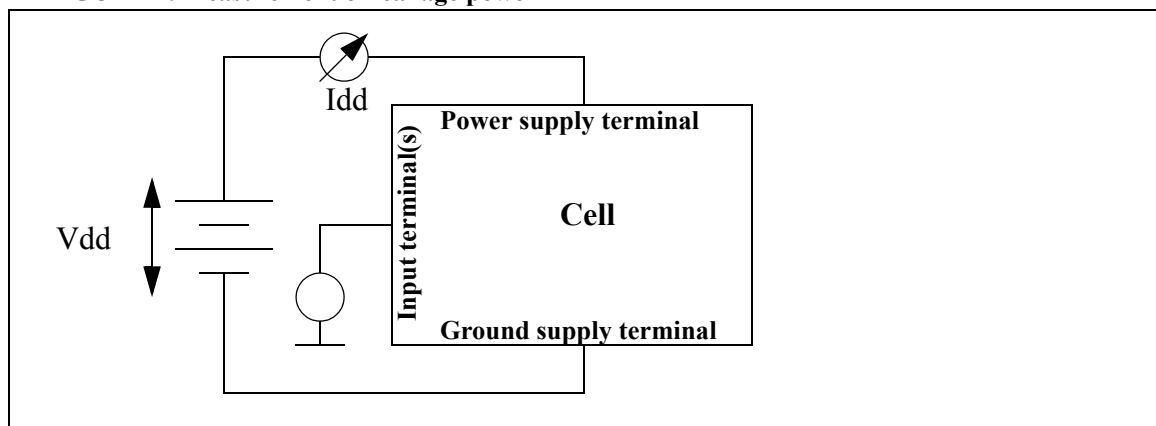
Switching and internal power, and to a lesser degree leakage power, are state-dependent.

In remainder of this section, a single constant power supply is assumed. Multiple power supplies are addressed in the next section. The voltage levels of signals in CMOS circuits are given by the electrical potentials of the power and ground supply.

1.1 Leakage power

Leakage power is measured by applying constant input voltages and a static current meter in series to the supply voltage to a cell, as illustrated in Figure 1 on page 3. The output pins are left unconnected.

FIGURE 1. Measurement of leakage power



The leakage power is given by the product $I_{dd} \cdot V_{dd}$.

Leakage power can be state-dependent or state-independent. The following Figure 2 on page 4 shows the description of both state-independent and state-dependent leakage power in Liberty.

FIGURE 2. Leakage power description in Liberty

```

/* Liberty */
cell (CellName) {
  cell_leakage_power : CellLeakagePowerValue ;
  leakage_power () {
    when : "BooleanExpression"
    value : LeakagePowerValue ;
  }
}

```

The following Figure 3 on page 4 shows the description of leakage power in ALF.

FIGURE 3. Leakage power description in ALF

```

/* ALF */
CELL CellName {
  POWER = CellLeakagePowerValue {
    MEASUREMENT = static ;
    CALCULATION = ABSOLUTE | INCREMENTAL ;
  }
  VECTOR (BooleanExpression) {
    POWER = LeakagePowerValue { MEASUREMENT = static ; }
  }
}

```

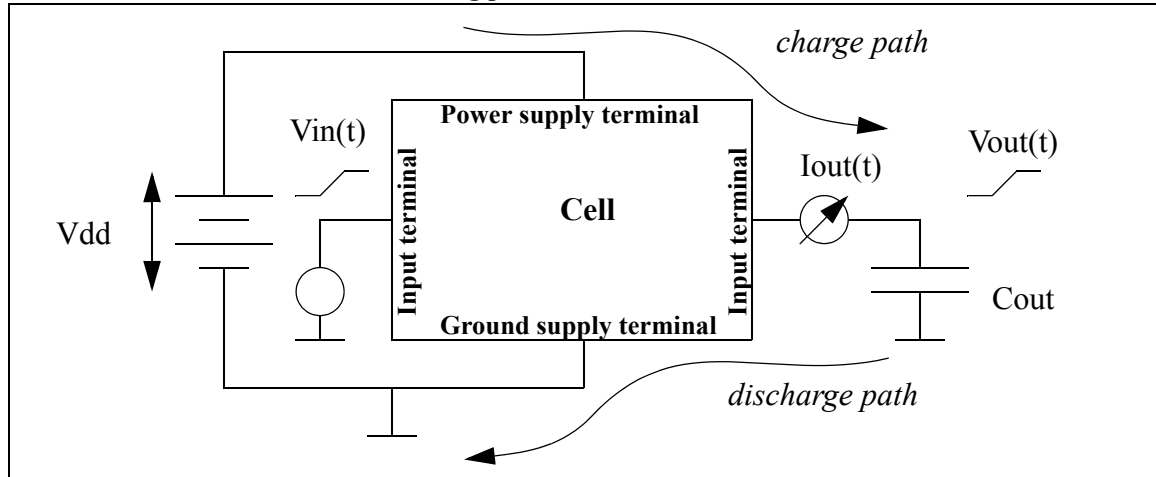
The **CALCULATION** annotation specifies whether or not the state-independent *CellLeakagePowerValue* includes the state-dependent *LeakagePowerValue*. An **INCREMENTAL** value shall be added to a state-dependent value. An **ABSOLUTE** value shall be used instead of a state-dependent value.

Question: How does Liberty handle concurrent state-dependent and state-independent values? Are they added up or used in a mutually exclusive way? Is the state-independent value the default, if no state applies?

Question: Does Liberty support tables or polynomial expressions for leakage power?

1.2 Switching power

Transient power is dissipated when a transient voltage $V_{in}(t)$ is applied to an input terminal that causes an output terminal to switch while the output terminal is connected to a capacitor C_{out} , as illustrated in Figure 4 on page 5. The energy transferred between the cell's power supply and the capacitor is called switching energy.

FIGURE 4. Measurement of switching power

The energy stored in the capacitor is given by the integral over $I_{out}(t) \cdot V_{out}(t)$, where $I_{out}(t) = C_{out} \cdot dV_{out}(t)/dt$.

This integral evaluates to $C_{out} \cdot 1/2 \cdot (V_{out}^2(t=T) - V_{out}^2(t=0))$, where T is the time it takes to completely charge or discharge C_{out} .

When the output signal rises, i.e., $V_{out}(t)$ changes from 0 to V_{dd} , $I_{out}(t)$ flows from the power supply terminal to C_{out} (see *charge path*). Therefore the supplied energy is given by the integral over $I_{out}(t) \cdot V_{dd}$.

When the output signal falls, i.e., $V_{out}(t)$ changes from V_{dd} to 0, $I_{out}(t)$ flows from C_{out} to the ground supply terminal (see *discharge path*). Therefore the supplied energy is zero.

Given this information, the energy components can be mathematically evaluated, as shown in the following table.

TABLE 1. Evaluation of switching energy

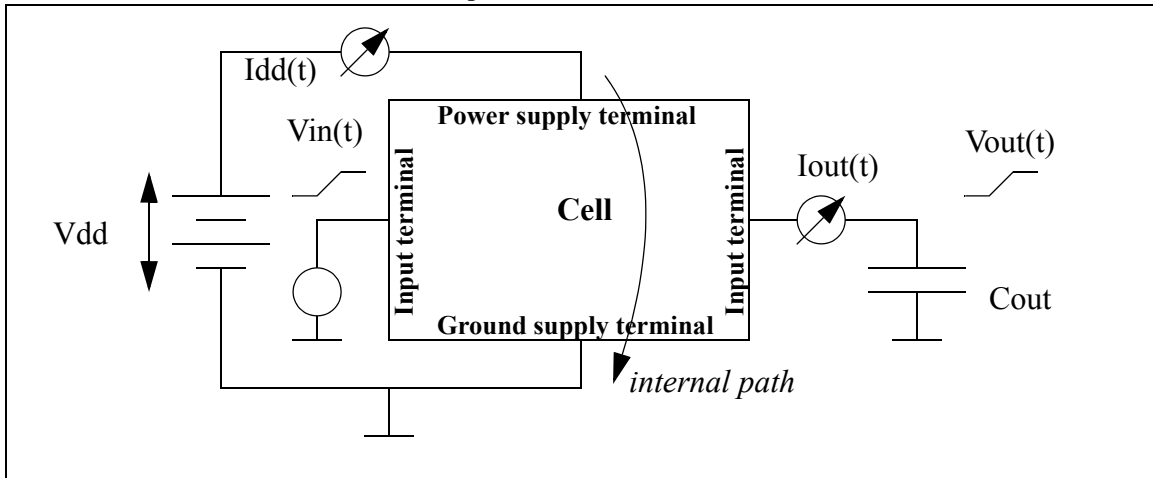
quantity	calculation	output rising	output falling
initial energy stored in C_{out}	$E_{initial} = 1/2 \cdot C_{out} \cdot V_{out}^2(t=0)$	0	$1/2 \cdot C_{out} \cdot V_{dd}^2$
final energy stored in C_{out}	$E_{final} = 1/2 \cdot C_{out} \cdot V_{out}^2(t=T)$	$1/2 \cdot C_{out} \cdot V_{dd}^2$	0
energy supplied through cell	E_{supply}	$C_{out} \cdot V_{dd}^2$	0
energy dissipated in the cell	$E_{supply} - (E_{final} - E_{initial})$	$1/2 \cdot C_{out} \cdot V_{dd}^2$	$1/2 \cdot C_{out} \cdot V_{dd}^2$

In a supply-oriented accounting method, the switching energy is simply given by the supplied energy. In a dissipation-oriented accounting method, the switching energy is given by the difference between the supplied energy and the stored energy in the capacitor. Regardless of which accounting method is used, the switching energy for output rising and output falling adds always up to $C_{out} \cdot V_{dd}^2$. Therefore no library characterization is needed for mathematical evaluation of the switching energy per se.

1.3 Internal power

Transient energy supplied and eventually dissipated in a cell is measured by applying a transient current meter in series to the power supply terminal of the cell, as illustrated in Figure 5 on page 6. The total transient energy is given by the integral over $V_{dd} \cdot I_{dd}(t)$. The internal energy is the difference between the total transient energy and the switching energy (see Section 1.2 on page 4).

FIGURE 5. Measurement of internal power



The following Figure 6 on page 6 shows the description of transient energy in Liberty.

FIGURE 6. Transient energy description in Liberty

```
/* Liberty */
cell (CellName) {
  pin(RelatedPinName) {
    direction : RelatedPinDirection;
  }
  pin(PinName) {
    direction : PinDirection;
    internal_power() {
      related_pin : "RelatedPinName"; /* this is optional */
      /* lib_PowerModel */
      ModelKeyword (CalculationType) { values ( /* lib_Data */ ); }
    }
  }
}
```

The following Figure 7 on page 7 shows the description of transient energy in ALF.

FIGURE 7. Transient energy description in ALF

```

/* ALF */
CELL CellName {
  PIN RelatedPinName {
    DIRECTION = RelatedPinDirection;
  }
  PIN PinName {
    DIRECTION = PinDirection;
  }
  VECTOR (VectorExpression) {
    ENERGY {
      /* ALF_data */
    }
  }
}

```

The ModelKeyword in Liberty specifies whether the transition observed at the pin associated with the transient energy is rising or falling. In ALF, the transition is specified within the vector expression.

The mapping between Liberty and ALF is shown in the following Table 2 on page 7.

TABLE 2. Mapping of Liberty and ALF constructs for transient energy measurements

Liberty construct		ALF construct	Comment
yes = with related pin no = without related pin		PN = Pin Name RPN = Related Pin Name	
Model Keyword		Vector Expression	see note ^a
power	no	?! PN	model applies for any transition
rise_power	no	01 PN	model applies for rise transition
fall_power	no	10 PN	model applies for fall transition
power	yes	?! RPN -> ?! PN	model applies for any transition
rise_power	yes	?! RPN -> 01 PN	model applies for rise transition
fall_power	yes	?! RPN -> 10 PN	model applies for fall transition

a. Note: The transition of the related pin (rise or fall) can not be expressed explicitly in Liberty. It has to be derived from the logical function and/or the unateness of a corresponding timing arc.

In Liberty, the data for transient energy measurements represents only the internal energy. In ALF, the data represents the total transient energy. To convert from Liberty to ALF data, one of the following methods can be used:

1. Add the switching energy data $\frac{1}{2} \cdot C_{out} \cdot V_{dd}^2$ to every “power” model. Add either $C_{out} \cdot V_{dd}^2$ to “rise_power” or $\frac{1}{2} \cdot C_{out} \cdot V_{dd}^2$ to both “rise_power” and “fall_power”.

2. Introduce one extra power vector (?! PN) or two extra power vectors

(01 PN) and (10 PN) with energy data $1/2 \cdot C_{out} \cdot V_{dd}^2$ for each.

Alternatively, add just one extra power vector (01 PN) with energy data $C_{out} \cdot V_{dd}^2$.

To convert from ALF to Liberty, subtract $1/2 \cdot C_{out} \cdot V_{dd}^2$ from every power vector featuring the sub-expression (... ?! PN). Subtract $1/2 \cdot C_{out} \cdot V_{dd}^2$ also from every power vector featuring one of the sub-expressions (... 01 PN) or (10 PN). Alternatively, subtract $C_{out} \cdot V_{dd}^2$ from (... 01 PN) only, subtract nothing from (10 PN). If the resulting energy data evaluates to zero, then the power vector can be eliminated altogether.

1.4 Power with state-dependency

State-dependent energy is expressed using the “when” statement in Liberty and a boolean co-factor in the vector expression in ALF. An example is shown below.

FIGURE 8. State-dependent energy example in Liberty and ALF

<pre> /* Liberty */ pin(Y) { internal_power() { related_pin : "A"; when : "E1&E2"; rise_power ... fall_power ... } internal_power() { related_pin : "A"; when : "E1&!E2"; rise_power ... fall_power ... } internal_power() { related_pin : "A"; when : "!E1&E2"; rise_power ... fall_power ... } } </pre>	<pre> /* ALF */ VECTOR ((?! A -> 01 Y)&(E1&E2)) { ENERGY ... } VECTOR ((?! A -> 10 Y)&(E1&E2)) { ENERGY ... } VECTOR ((?! A -> 01 Y)&(E1&!E2)) { ENERGY ... } VECTOR ((?! A -> 10 Y)&(E1&!E2)) { ENERGY ... } VECTOR ((?! A -> 01 Y)&(!E1&E2)) { ENERGY ... } VECTOR ((?! A -> 10 Y)&(!E1&E2)) { ENERGY ... } </pre>
---	--

Could not find the Liberty documentation about default condition. Is it just a power-statement without when, or is there a specific default keyword?

A default value condition in ALF is specified by a timing model with **CALCULATION** annotation value **absolute** in the context of an ALF vector without boolean co-factor¹.

The default condition shall apply in the following situations:

- None of the specified value conditions evaluates true
- The application context (e.g. a record of simulation events) does not provide enough information to decide whether one, and only one, specified value condition evaluates true
- The application tool does not support evaluation of a value condition

1.5 Power involving a bus or a bundle of pins

The following power relationships involving multiple pins can be defined in Liberty:

TABLE 3. power relationships involving multiple pins

object	related object	comment
scalar pin	scalar pin	described in Section 1.3 on page 6
scalar pin	multiple scalar pins	see Figure 9 on page 10
scalar pin	bus	not in Liberty doc
bundle	scalar pin	see Figure 10 on page 10
bundle	bitwidth-compatible bundle	see Figure 11 on page 10
bundle	multiple scalar pins and/or bitwidth-compatible bundle	see Figure 12 on page 11
bundle	bus	not in Liberty doc
bus	scalar pin	see Figure 13 on page 12
bus	bitwidth-compatible bus	see
bus	multiple scalar pins and/or bitwidth-compatible bus	see Figure 15 on page 13
bus	bus	see Figure 16 on page 14

1. If the value of the **CALCULATION** annotation is **absolute**, then the power data shall not be combined with any other power data. If the value of the **CALCULATION** annotation is **incremental**, then the power data can be combined with other power data.

FIGURE 9. Power relationship between scalar pin and multiple scalar pins

<pre> /* Liberty */ pin(A) { direction : input ; } pin(B) { direction : input ; } pin(C) { direction : input ; } pin(Y) { direction : output ; internal_power() { related_pin : "A B C"; power ... } } </pre>	<pre> /* ALF */ PIN A { DIRECTION = input; } PIN B { DIRECTION = input; } PIN C { DIRECTION = input; } PIN Y { DIRECTION = output; } GROUP A_B_C { A B C } VECTOR (?! A_B_C -> ?! Y) { ENERGY ... } </pre>
---	--

FIGURE 10. Power relationship between bundle and single scalar pin

<pre> /* Liberty */ pin(A) { direction : input ; } pin(X) { direction : output ; } pin(Y) { direction : output ; } pin(Z) { direction : output ; } bundle(X_Y_Z) { members (X,Y,Z); internal_power() { related_pin : "A"; power ... } } </pre>	<pre> /* ALF */ PIN A { DIRECTION = input; } PIN X { DIRECTION = output; } PIN Y { DIRECTION = output; } PIN Z { DIRECTION = output; } GROUP X_Y_Z { X Y Z } VECTOR (?! A -> ?! X_Y_Z) { ENERGY ... } </pre>
--	--

FIGURE 11. Power relationship between bundle and bitwidth-compatible bundle

<pre> /* Liberty */ pin(A) { direction : input ; } pin(B) { direction : input ; } pin(C) { direction : input ; } pin(X) { direction : output ; } pin(Y) { direction : output ; } pin(Z) { direction : output ; } bundle(A_B_C) { members (A,B,C); } bundle(X_Y_Z) { members (X,Y,Z); internal_power() { related_pin : "A_B_C"; power ... } } </pre>	<pre> /* ALF */ PIN A { DIRECTION = input; } PIN B { DIRECTION = input; } PIN C { DIRECTION = input; } PIN X { DIRECTION = output; } PIN Y { DIRECTION = output; } PIN Z { DIRECTION = output; } PINGROUP[1:3] A_B_C { MEMBERS { A B C } } PINGROUP[1:3] X_Y_Z { MEMBERS { X Y Z } } GROUP BundleBit { 1 : 3 } VECTOR (?! A_B_C[BundleBit] -> ?! X_Y_Z[BundleBit]) { ENERGY ... } </pre>
---	--

FIGURE 12. Power relationship between bundle and multiple scalar pins and/or bitwidth-compatible bundles

```

/* Liberty */
pin(A) { direction : input ; }
pin(B) { direction : input ; }
pin(C) { direction : input ; }
pin(D) { direction : input ; }
pin(E) { direction : input ; }
pin(X) { direction : output ; }
pin(Y) { direction : output ; }
pin(Z) { direction : output ; }
bundle(A_B_C) {
  members (A,B,C);
}
bundle(X_Y_Z) {
  members (X,Y,Z);
  internal_power() {
    related_pin : "A_B_C D E";
    power ...
  } /* equivalent to following:
  internal_power() {
    related_pin : "A_B_C";
    power ...
  }
  internal_power() {
    related_pin : "D E";
    power ...
  } */
}

/* ALF */
PIN A { DIRECTION = input; }
PIN B { DIRECTION = input; }
PIN C { DIRECTION = input; }
PIN D { DIRECTION = input; }
PIN E { DIRECTION = input; }
PIN X { DIRECTION = output; }
PIN Y { DIRECTION = output; }
PIN Z { DIRECTION = output; }
PINGROUP[1:3] A_B_C {
  MEMBERS { A B C }
}
PINGROUP[1:3] X_Y_Z {
  MEMBERS { X Y Z }
}
GROUP BundleBit { 1 : 3 }
GROUP D_E { D E }
VECTOR (
  ?! A_B_C[BundleBit] ->
  ?! X_Y_Z[BundleBit]) {
  ENERGY ...
}
VECTOR (
  ?! D_E ->
  ?! X_Y_Z[BundleBit]) {
  ENERGY ...
}

```

FIGURE 13. Power relationship between bus and single scalar pin

<pre> /* Liberty */ type(Bit) { base_type : array ; data_type : bit ; bit_width : 3 ; bit_from : 1 ; bit_to : 3 ; downto : false ; } pin(A) { direction : input ; } bus(Y) { direction : output ; bus_type : Bit ; internal_power() { related_pin : "A"; power ... } } </pre>	<pre> /* ALF */ GROUP Bit { 1 : 3 } PIN A { DIRECTION = input; } PIN [1:3] Y { DIRECTION=output; } VECTOR (?! A -> ?! Y[Bit]) { ENERGY ... } </pre>
---	--

FIGURE 14. Power relationship between bus and bitwidth-compatible bus

<pre> /* Liberty */ type(Bit) { base_type : array ; data_type : bit ; bit_width : 3 ; bit_from : 1 ; bit_to : 3 ; downto : false ; } bus(A) { direction : input ; bus_type : Bit ; } bus(Y) { direction : output ; bus_type : Bit ; internal_power() { related_pin : "A"; power ... } } </pre>	<pre> /* ALF */ GROUP Bit { 1 : 3 } PIN [1:3] A { DIRECTION = input; } PIN [1:3] Y { DIRECTION=output; } VECTOR (?! A[Bit] -> ?! Y[Bit]) { ENERGY ... } </pre>
--	--

FIGURE 15. Power relationship between bus and multiple scalar pins and/or bitwidth-compatible bus

<pre> /* Liberty */ type(Bit) { base_type : array ; data_type : bit ; bit_width : 3 ; bit_from : 1 ; bit_to : 3 ; downto : false ; } bus(A) { direction : input ; bus_type : Bit ; } pin(B) { direction : input ; } pin(C) { direction : input ; } bus(Y) { direction : output ; bus_type : Bit ; internal_power() { related_pin : "A B C"; power ... } /* equivalent to following: internal_power() { related_pin : "A"; power ... } internal_power() { related_pin : "B_C"; power ... } */ } </pre>	<pre> /* ALF */ GROUP Bit { 1 : 3 } PIN [1:3] A { DIRECTION = input; } PIN B { DIRECTION = input; } PIN C { DIRECTION = input; } PIN [1:3] Y { DIRECTION=output; } VECTOR (?! A[Bit] -> ?! Y[Bit]) { ENERGY ... } GROUP B_C { B C } VECTOR (?! B_C -> ?! Y[Bit]) { ENERGY ... } </pre>
---	--

FIGURE 16. Power relationship between bus and bus with arbitrary bitwidth

<pre> /* Liberty */ type(Bit) { base_type : array ; data_type : bit ; bit_width : 3 ; bit_from : 1 ; bit_to : 3 ; downto : false ; } bus(A) { direction : input ; bus_type : Bit ; } bus(Y) { direction : output ; bus_type : Bit ; internal_power() { related_bus_pins : "A"; power ... } } </pre>	<pre> /* ALF */ GROUP Bit { 1 : 3 } GROUP OtherBit { 1 : 3 } PIN [1:3] A { DIRECTION = input; } PIN [1:3] Y { DIRECTION=output; } VECTOR (?! A[Bit] -> ?! Y[OtherBit]) { ENERGY ... } </pre>
---	---

1.6 Power involving a switching interval

To do: Power involving a switching interval, rising|falling|switching_together

1.7 Power involving functionally equivalent output pins

To do: Power with equal_or_opposite output

2.0 Multiple power supplies

Multiple power supplies for a cell (e.g. level shifter, back-biasing) require each power supply voltage to become a characterization variable.

Association of rails with pins and with modes of power consumption is required in the library.

Liberty supports two variables for dual-supply voltage, `voltage` and `voltage1`. Also it supports a mapping construct between the variable and a power rail name.

```
variables(temperature, ..., voltage, voltage1) ;
mapping(voltage, VDD) ;
mapping(voltage1, VDD2) ;
```

The `power_level` attribute associates `internal_power` with a power rail.

FIGURE 17. Liberty description of association between energy and a power rail

```
/* Liberty */
library (LibraryName) {
  power_supply() {
    power_rail(VDD , 1.5) ;
    power_rail(VDD2 , 1.0) ;
  }
  cell (CellName) {
    pin(PinName) {
      direction : PinDirection;
      internal_power(E1) {
        power_level : VDD ;
        /* put lib_PowerModel here */
      }
      internal_power(E2) {
        power_level : VDD2 ;
        /* put lib_PowerModel here */
      }
    }
  }
}
```

FIGURE 18. ALF description of association between energy and a power rail

```

/* ALF */
LIBRARY LibraryName {
  CLASS VDD { USAGE = SUPPLY_CLASS ; VOLTAGE = 1.5 ; }
  CLASS VDD2 { USAGE = SUPPLY_CLASS ; VOLTAGE = 1.0 ; }
  CELL CellName {
    PIN PinName {
      DIRECTION = PinDirection ;
    }
    VECTOR (VectorExpression) {
      ENERGY E1 {
        SUPPLY_CLASS = VDD ;
        /* ALF_data */
      }
      ENERGY E2 {
        SUPPLY_CLASS = VDD2 ;
        /* ALF_data */
      }
    }
  }
}

```