Library Harmonization for Power

Version	Date
0.0	2004/03/22

Table of contents

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.
- Internal power is associated with short-circuit current and charge/discharge of internal capacitances.
- Leakage power is dependent on transistor threshold voltages.

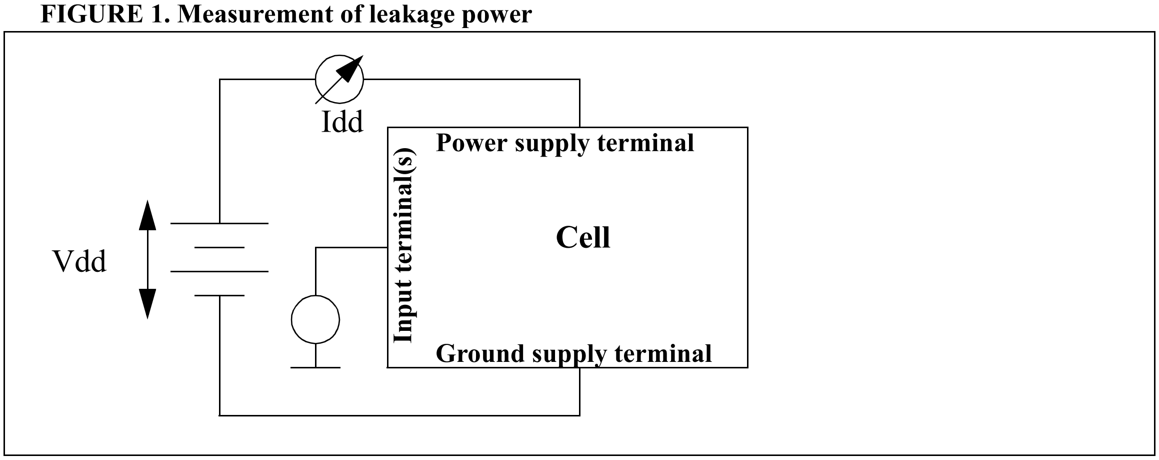
Switching and internal power, and to a lesser degree leakage power, are state-dependent.

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 Idd*Vdd.

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.

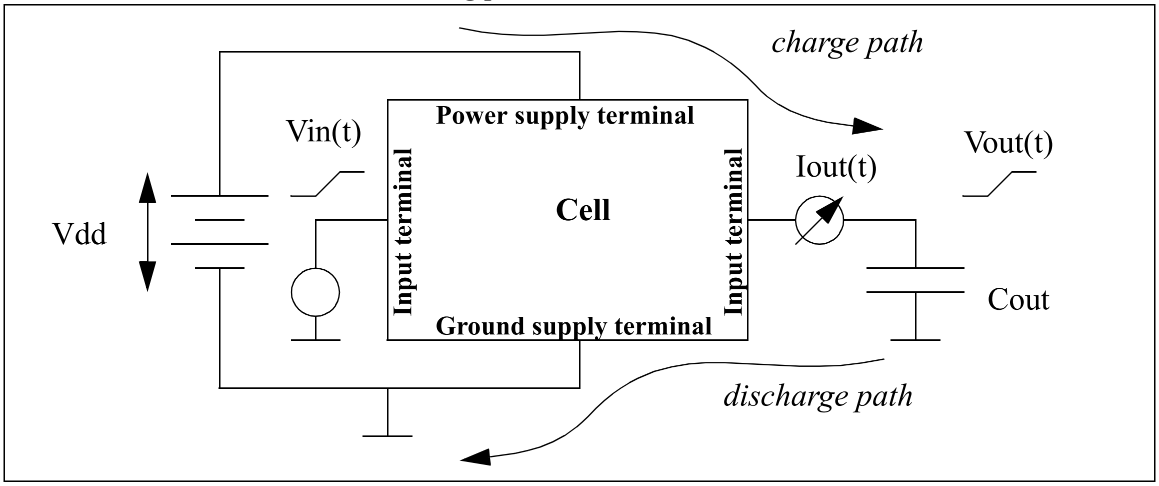
Question: How does liberty handle concurrent state-dependent and state-independent values? Are they added up or used in a mutually exclusive way? Is the state-independent value the default, if no state applies?

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

1.2 Switching power

Transient power is dissipated when a transient voltage Vin(t) is applied to an input terminal that causes an output terminal to switch while the output terminal is connected to a capacitor Cout, as illustrated in Figure 4 on page 5. The energy transfered between the cellís 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 Iout(t)*Vout(t), where Iout(t) = Cout * dVout(t)/dt.

This integral evaluates to Cout * 1/2 * ($Vout^2$(t=T) - $Vout^2$(t=0)), where T is the time it takes to completely charge or discharge Cout .

When the output signal rises, i.e., Vout(t) changes from 0 to Vdd, Iout(t) flows from the power supply terminal to Cout (see *charge path*). Therefore the supplied energy is given by the intergral over Iout(t)*Vdd.

When the output signal falls, i.e., Vout(t) changes from Vdd to 0, Iout(t) flows from Cout 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

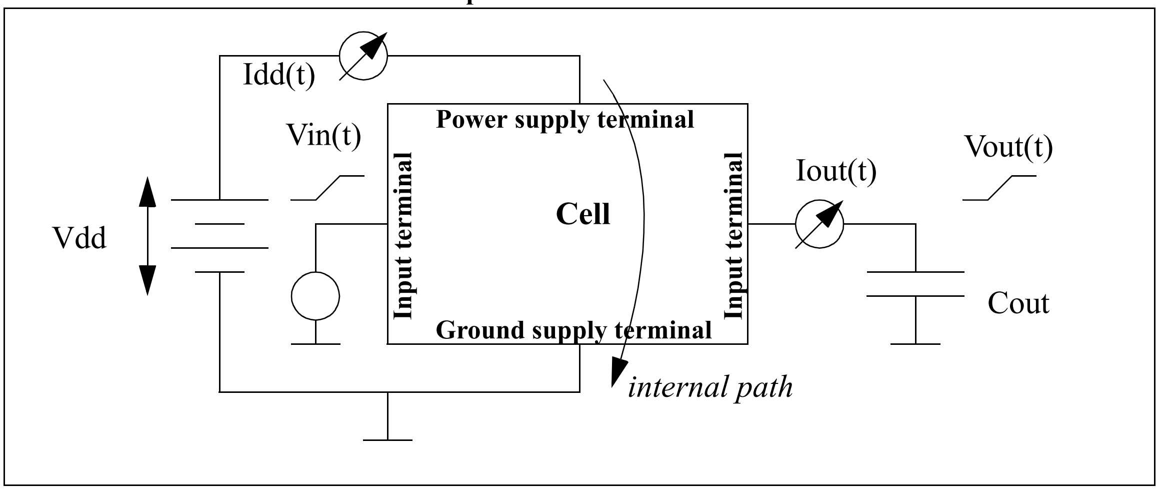
quantity	**calculation**	**output rising**	**output falling**
initial energy stored in Cout	$E_{initial}$ = 1/2*Cout*$Vout^2$(t=0)	0	1/2*Cout*Vdd^2
final energy stored in Cout	E_{final} = 1/2*Cout*$Vout^2$(t=T)	1/2*Cout*Vdd^2	0
energy supplied through cell	E_{supply}	Cout*Vdd^2	0
energy dissipated in the cell	E_{supply} - (E_{final} - $E_{initial}$)	1/2*Cout*Vdd^2	1/2*Cout*Vdd^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 Cout*Vdd^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 Vdd*Idd(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";
      /* 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 PN = Pin Name, RPN = Related Pin Name	Comment
Model Keyword	***Vector Expression***	
power	?! RPN -> ?! PN	model applies for any transition
rise_power	?! RPN -> 01 PN	model applies for rise transition
fall_power	?! RPN -> 10 PN	model applies for fall transition

Question: Is there a way to specify the transition (rise or fall) for the related pin in liberty?

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 $1/2*Cout*Vdd^2$ to every ìpowerî model. Add either $Cout*Vdd^2$ to ìrise_powerî or $1/2*Cout*Vdd^2$ to both ìrise_powerî and ìfall_powerî.
2. Introduce one extra power vector (?! PN) or two extra power vectors (01 PN) and (10 PN) with energy data $1/2*Cout*Vdd^2$ for each.
   Alternatively, add just one extra power vector (01 PN) with energy data $Cout*Vdd^2$.

To convert from ALF to liberty, subtract $1/2*Cout*Vdd^2$ from every power vector featuring the sub-expression (... ?! PN). Subtract $1/2*Cout*Vdd^2$ also from every power vector featuring one of the sub-expressions (... 01 PN) or (10 PN). Alternatively, subtract Cout*Vdd^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.

To do:

- Power with state-dependency (when)
- Power involving a bus and/or a bundle of pins
- Power involving a switching interval, rising|falling|switching_together
- 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 8. 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 9. 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 */
      }
    }
  }
}
```