



Enhancements to the `idt()` Operator

Ken Kundert

Version 1a, 6 December 2006 *A proposal on how to fix the Verilog-AMS `idt()` operator.*

Last updated on December 6, 2006. Contact the authors via e-mail at consulting@designers-guide.com.

I propose that the following replace section 4.4.5 of the Verilog-AMS LRM. It clears up the following issues:

1. Previously the reset operation could not have zero duration. This was fixed by defining t_a to be "the time when assert was last nonzero" rather than "the when assert last became 0".
2. Cleaned up the description of how the initial condition is determined if the user does not specify one.
3. Changed the starting point of the integration from 0 to t_0 , which is the start time of the simulation (simulations do not necessarily start at time 0).
4. Defined the behavior of $\text{idt}(\text{expr}, \text{ic}, \text{assert})$ when assert is 0 at t_0 . This was an arbitrary choice on my part. It might make more sense to have the third form of idt behave like the first form at t_0 if assert equals 0 rather than having it behave like the second form, particularly since to provide an absolute tolerance with existing simulators one needs to specify ic and assert even though they might not be wanted.

These changes modify the terminology of the LRM a bit, it would probably be a good idea to update the section on idtmod to conform to this terminology.

1 Time integral operator

The idt operator computes the time-integral of its argument, as shown in Table 1.

TABLE 1 Time integral.

Expression	Comments
$\text{idt}(\text{expr})$	Returns $\int_{t_0}^t x(\tau) d\tau + c$ where $x(\tau)$ is the value of expr at time τ , t_0 is the time the start time of the simulation, t is the current time, and c is the initial starting point as determined by the simulator and is generally the DC value (the value that makes expr equal to zero).
$\text{idt}(\text{expr}, \text{ic})$	Returns $\int_{t_0}^t x(\tau) d\tau + c$ where in this case c is the value of ic at t_0 .
$\text{idt}(\text{expr}, \text{ic}, \text{assert})$	Returns $\int_{t_a}^t x(\tau) d\tau + c$ where c is the value of ic at t_a , which is the time when assert was last nonzero or t_0 if assert was never nonzero.

TABLE 1 *Time integral.*

Expression	Comments
<code>idt(expr, ic, assert, abstol)</code>	Same as above, except the absolute tolerance used to control error in the numerical integration process is specified explicitly.
<code>idt(expr, ic, assert, nature)</code>	Same as above, except the absolute tolerance used to control error in the numerical integration process is taken from the specified nature.

A simple example that demonstrates the first form is a simple model for an opamp.

```

module opamp(out, pin, nin);
output out;
input pin, nin;
voltage out, pin, nin;

analog
    V(out) <+ idt(V(pin,nin));
endmodule

```

Here the opamp is simply modeled as an integrator. In this case the initial condition for the integrator is found by the simulator. Generally the DC operating point is used. For the DC operating point to exist for an integrator that does not have an initial condition explicitly specified the integrator must exist within a negative feedback loop that drives its argument to 0. Forcing the output of the integration operator to be a particular value at start of the simulation using something like

```
V(out) <+ idt(V(pin,nin), 0);
```

avoids this issue.

Using the *assert* argument, the output of the integration operator can be reset to a given value at any time. This feature is demonstrated in the following model, which uses the *idt* operator to generate a periodic ramp waveform:

```

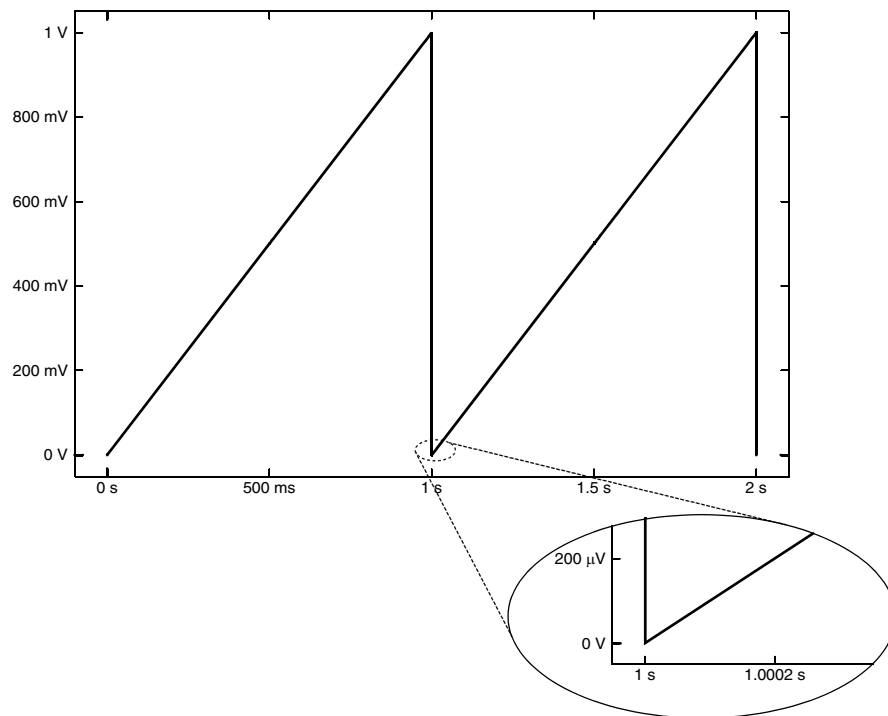
module ramp_generator(out);
output out;
voltage out;
integer reset;

analog begin
    reset = 0;
    @(timer(1, 1))
        reset = 1;
    V(out) <+ idt(1.0, 0, reset);
end
endmodule

```

The output of this model is shown in Figure 1. Notice that in this model the reset occurs instantaneously.

FIGURE 1 The output from the ramp generator.



The optional parameter *abstol* or *nature* is used to derive an absolute tolerance if needed. Whether an absolute tolerance is needed depends on the context where *idt()* is used. (See Section 4.4.3 for more information.) The absolute tolerance applies to the input of the *idt()* operator and is the largest signal level that is considered negligible.