4. Simulation algorithms

4.1. Analog engine

The use of modified nodal analysis (MNA) [Ho75] enables separation of the modeling process from the mathematical apparatus used for simulation. Contribution of each analog component to the system of equation is determined as the model “stamp”. In a behavioral simulator, model (i.e. stamp) can be defined by the user. On the other hand, the simulation algorithms are preprogrammed in a simulation engine. Alecsis have SPICE-like simulation algorithms implemented. In particular, we have used Gear methods for numerical integration of ordinary differential equations [Gear71], Newton-Raphson method for nonlinear equations, and modified Berry’s algorithm for solving system of equations characterized by sparse matrices [Berr71].

The user should not be aware of particular algorithms in the simulation engine when describing the model. However, the numerical integration of differential equations and the linearization of nonlinear equations are performed in the stamp description. Therefore, the modeling procedure cannot be quite independent from the algorithms in the simulation engine. However, we have tried to hide the algorithms of the simulation engine from the model designer. For instance, user can invoke operators `ddt` and `d2dt2` when he wants to describe differential equations. The method for numerical integration would be then invoked automatically. The same concept can be found in other analog hardware description languages [Getr89, Kazm92, Pabs95].

We will describe here shortly the algorithm of time advancing in the analog engine. This is important, since this algorithm has to be synchronized with the time-control of a discrete-event simulator. The analog simulation time is driven by the local truncation error (LTE) of the numerical integration method. This LTE is calculated after obtaining the convergence for one time instant. When the LTE is smaller than the allowed limit, the analog solution is accepted and the relation between LTE and the allowed limit is used to determine the new integration step (new simulation time). If the LTE is larger than the limit, the solution is discarded, and the simulation is repeated for a shorter time-step (backtracking in time). That shorter time step is again calculated on the basis of LTE value. Therefore, backtracking in the analog simulator is possible, but not more than for one time step.

4.2. Discrete-event engine

The logic simulation engine of Alecsis simulator uses the data structures built up by the compiler to propagate events through the digital nets - signals. An event represents a change in the value of a signal. When an event on a signal occurs, processes that are sensitive to that signal are activated. Each process that assigns states to a signal creates a driver for that signal. The driver is a time-ordered linked list that contains information of future events on a signal in the form of (time, state) pairs. Well-known inertial and transport delay mechanisms [Lips89] are used in assigning the future events to the driver. A signal can have more than one driver in which case the resolution function is needed to resolve the signal state. It is user's responsibility to provide the resolution function (applying preferred resolution strategy) and attach it to the signal that has more than one driver. In addition to events that convey updates in signal values, the logic simulation machine of Alecsis simulator also processes the control events imposed by the synchronization signals (`initial`, `per_moment`, `final`, ...) at appropriate moments during the simulation.

All events scheduled to occur in the future (relative to the current simulation time) are said to be pending and are maintained in a time-ordered linked list called the global event list (GEL). Besides the time of the event, an entry to the GEL contains the pointer to the signal driver that store the particular event information. GEL is used by the algorithm for advancing the time. The current time is the time of the first GEL entry. Simulation proceeds by processing all events queued at the current time (deleting them from the queue after processing) and then moving to process events at the next time in the GEL. Simulation ends when the GEL is empty or when user specified simulation duration is exceeded. When the process execution results in scheduling an event with zero propagation delay, the "delta delay" is used - this event is processed in the next simulation cycle, but the simulation time remains the same.

The changes in the values of the primary input signals are defined using `initial` processes. All the events defining the applied stimuli are inserted in the global event list and primary input signals drivers before the simulation. Clock generators can be defined that periodically insert events during the simulation.

Logic simulation algorithm consists of initialization and simulation phases. Initialization phase assumes activation of all processes and evaluation of states of all signals. Process activation is automatic, as well as the assigning the default or user defined values to the signals, but further initialization procedure must be built in the circuit model by the designer. The simulation phase assumes propagation of events from primary inputs and internal clock generators to the outputs of the circuit. Both phases use the same event handling procedures that can be divided into two basic operations:

1) evaluation of the new logic state of the signal and

2) execution of the activated process.

Before assigning the new state to the signal, the simulation machine applies the resolution function if the particular signal has more than one driver. If there is a change in signal state, the signal state is updated and its fan-out list is followed to determine the activated processes. When executed, in the next step, processes assign the new events to some signals. Those events are scheduled in drivers and GEL, and simulation proceeds by taking the next event from the GEL.

4.3. Hybrid simulation

For the mixed-signal simulation, analog and discrete-event engines should be connected into one simulation engine. There are two aspects of this connection:

- signal conversion (A/D or D/A) (mixed-signal simulation) and
- time synchronization of analog and discrete-event engines (mixed-mode simulation)..

4.3.1. A/D and D/A interfaces

A net to which both analog and digital components are connected is called hybrid net. After the compiler had built all the data structures representing the simulated circuit, Alecsis detects and eliminates hybrid nets. In order to make distinction between analog and digital portion of the circuit, simulator automatically inserts interface circuits called analogue to digital (A/D) and digital to analogue (D/A) converters. Models of interface circuits are part of system model and are closely related to the digital modules as described earlier.

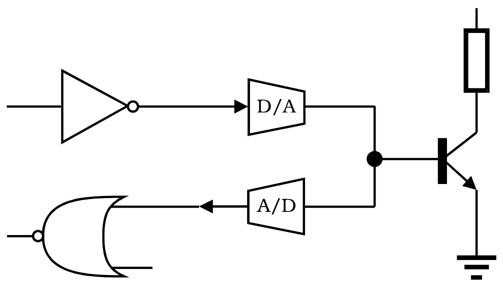

Fig. 4.1: Automatic A/D and D/A converter insertion for hybrid net to which both inputs and outputs of digital modules are connected.

A/D converter is inserted where an input of a digital module is connected to a hybrid node. D/A converter module is inserted where an output of a digital module is connected to a hybrid node. If there are both digital modules inputs and outputs connected to a hybrid node, both types of converters are inserted as illustrated in Fig. 4.1 [Nich92]. A possible A/D module structure for CMOS logic gates is shown in Fig. 4.2. The gate input is seen from the analog portion of the circuit as an input capacitance, while module `cmos_a2d` assigns new states to the digital output signal when the value of analog node crosses the defined thresholds. A/D converter module is parametrized in terms of threshold voltages and input capacitance. A simple model of D/A module for CMOS logic gate output is shown in Fig. 4.3. The gate output impedance is modeled by an RC circuit, while the signal transfer is expressed by a current source controlled by the logic states at the digital output. The interface is parametrized in terms of the output impedances and transition times. An 'X' (unknown) state from the logic circuit cannot be mapped directly into voltages, currents or impedances. The D/A converter module can be parametrized to convert an 'X' to '0', '1' or the previous determinate state.

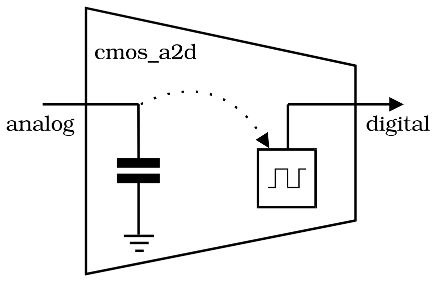

Fig. 4.2: Simple A/D converter model for CMOS gates. Input capacitance of CMOS gate is inserted at the analog side, and digital events are transmitted to the digital side when analog variable crosses the specified thresholds.

In order to produce a smooth ramp from the previous logic state to the new one, the converter should start generating the ramp *before* the actual event occurs. Since it is not possible in logic simulation machine to backtrack the current simulation time, digital modules are bound to assign events to the hybrid nets 1/2 of the ramp duration time (transition time) before it actually occurs. Therefore, it is necessary that the transition time (parameter of the converter) is a member of the model card of the digital module. This advocates the used model class inheritance concept described earlier. Detection if the net has a hybrid character is provided by the signal attribute `hybrid`.

```
      if (y->hybrid)
            y <- a&b after this->delay(a&b, y) -
                        this->get_trans_time() / 2.0;
      else
            y <- a&b after delay;
```

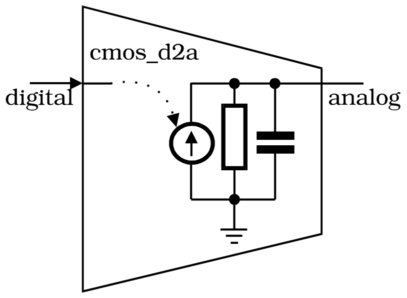

Fig. 4.3: Simple D/A converter model for CMOS gates. The output of CMOS gate is modeled with output resistance and capacitance, while its driving capability is modeled with current source controlled by the gate output logic state.

After inserting converter modules, digital and analog portions of the circuit can be separately simulated.

4.3.2. Initialization

The initialization phase in the mixed-signal circuit concerns both digital and analog part. Simulation of analog circuits requires a consistent initial state, otherwise the whole analysis may fail. Since analog simulation cannot cope with unknown state, the simulator performs initialization in the following manner:

1) Firstly, initialize the digital portion, i.e. compute the initial values of signals.

2) Than, execute a DC operation point computation in analog portion.

Thanks to modeling capabilities of AleC++, it is possible to include into digital modules the initialization mode as well as the normal operational mode. Initialization of digital part of the circuit is performed by the zero-delay logic simulation. To assure that zero delay is used while digital part settles in a stable state, user can check current simulation time (control parameter now), as in the next example.

```
if (now==0.0s) {  //  initialization mode
  out <- a & b after 0.0s;
}
else {  //  normal operational mode
  out <- a & b after delay_func(out, a & b, trise, tfall);
}
```

If digital simulation agent fails to initialize some signals that are needed for analog part initialization due to unknown input states that comes from analog part of the circuit, different initialization strategies can be applied, but those strategies depend on modeler and cannot be automated.

4.3.3. Time control mechanism

As we have described above, analog and logic simulation engines have completely different mechanisms of advancing the time. When a mixed-signal simulator is implemented, these two algorithms should be unified. In our implementation, the analog engine is the main one, it behaves also as a synchronization agent. When it is necessary, the analog engine invokes the discrete-event engine (Fig.2.1). The usual LTE-driven time algorithm of the analog engine is modified to allow for synchronization with the discrete-event engine:

- If the logic event is waiting to be executed in a particular time in the event list, analog simulation time cannot advance over that time. That means, analog time step would be shortened to match the time instant when the logic event happens. In that time instant, both engines are active (iteratively, if necessary, until all logic events for that time instant are executed).
- When the analog engine solves given time instant, that solution can give conditions for generating a logic event. However, the actual time instant when the analog value passes the threshold is not known precisely, since the analog time step is finite [Brow91]. For that reason, this analog time step can be discarded and the time step shortened. In this way, time-instant when the given analog value passes the threshold, i.e. when the logic event is generated on A/D node can be found with better accuracy.

Therefore, in mixed-signal simulation, logic engine is not responsible for advancing the time. When necessary, the analog engine invokes the logic engine to execute one delta-cycle of events in a given time-instant.