Simulation digitaler Systeme auf verschiedenen Abstraktionsebenen

Franz J. Rammig
Universität-Gesamthochschule-Paderborn

1. Zusammenfassung

Es werden Techniken zur Simulation digitaler Systeme vorgestellt. Zunächst wird auf verschiedene Abstraktionsebenen der Modellierung eingegangen. Für die Modellierung von Einzelkomponenten werden Beispiele auf verschiedenen Abstraktionsebenen genannt. Auf dieser Basis werden Kompositionstechniken und Simulationsprototypen angegeben. Auf die Simulation der Schalterebene in diesem Kontext wird besonders eingegangen. Als Beispiel für einen "mixed level"-Simulator wird der Simulator für die Hardware Beschreibungssprache CAP/DSDL vorgestellt.

2. Abstraktionsebenen

Es liegt nahe, hochintegrierte Schaltkreise als Objekte sehr hoher Komplexität auf verschiedenen Abstraktionsstufen zu beschreiben. Dabei konstituiert eine Einteilung in Abstraktionsebenen nicht nur eine Hierarchie von Objekten verschiedener Komplexität sondern in der Regel auch ein System unterschiedlicher Systemsichten. Es ist hauptsächlich eine gemeinsame Systemsicht (Modellvorstellung), die eine Abstraktionsebene konstituiert, innerhalb einer Abstraktionsebene sind dann sehr wohl hierarchische Beschreibungen möglich (und sinnvoll).

Es gibt kein "genormtes" Ebenensystem, doch kann das im Folgenden vorgestellte System als repräsentativ betrachtet werden.

Ebene 7: Systemebene

Die Modellvorstellung auf dieser Ebene ist die kooperierender semiautonomer Module (z.B. Prozessoren, DMA-Kanäle u.ä.). Typische Elementarbausteine sind abstrakte Datentypen. Das Zeitmodell reduziert sich in der Regel auf eine Kausalitätsstruktur. Die beobachtbaren Werte an Meßpunkten sind Werte aus einem frei definierbaren endlichen Wertebereich. Als typische dedizierte Sprachen für diese Ebene sind SIMULA /SIM/ oder OCCAM /IN1/ zu nennen.

Ebene 6: Algorithmische Ebene

Die Moduln der Systemebene kann man als Prozessoren auffassen, deren Wirkungsweise in Form eines ADT definiert ist. Für den Instruktionssatz eines Prozessors ist nun ein Interpretationsalgorithmus anzugeben. Damit ist man auf der algorithmischen Ebene angelangt. Die Modellvorstellung ist die eines (in der Regel hochgradig nebenläufigen) Algorithmus (z.B. interpretiertes Petri Netz), wobei auf Datenobjekten operiert wird, die relativ hardware-nahe definiert sind. Man hat also manipulative Elementarobjekte wie Register, Busse, Operatoren und algorithmische Elementarobjekte wie sequentielle Ausführung, Parallelausführung, Fallunterscheidung,Schleifen. Bezüglich des Zeitmodells beschränkt man sich oft weiterhin auf eine Kausalitätsstruktur. Allerdings erscheint es manchmal auch sinnvoll zu sein, bereits auf dieser Ebene eine erste Taktstruktur (Zeit als Zählung von Taktsignalen) einzuführen bzw. erste Realzeitaussagen zu machen. Die beobachtbaren Werte sind auf dieser Ebene meist Bitketten mit Interpretation. Als typische dedizierte Sprache für diese Ebene ist wohl ISPS /BA1/ zu nennen.

Ebene 5: Register Transfer Ebene

Die RT-Ebene ergibt sich aus der algorithmischen Ebene durch Invertierung: Das System wird nicht mehr aus der Gesamtsicht des Algorithmus gesehen (imperative Sicht) sondern aus Sicht der gesteuerten Objekte (reaktive Sicht). Wichtig ist die Separation zwischen steuerndem Automaten und darauf reagierendem Datenmanipulationsteil, wobei der Steuerungsteil natürlich sehr wohl auch aus reaktiver Sicht angegeben werden kann. Als Zeitmodell hat man in der Regel ein Taktschema zusammen mit einer Realzeitvorstellung. Elementarbausteine sind Register, Busse, Operationswerke. Beobachtbare Werte sind Bitketten ohne Interpretation. CDL /CH1/ mag als klassisches Beispiel einer dedizierten Sprache für diese Ebene gelten.

Ebene 4: Gatterebene

Expandiert man die Elementarobjekte der RT-Ebene in Schaltwerke aus logischen Gattern, so erhält man die Gatterebene. Die Separation zwischen Datensignalen und Kontrollsignalen geht hierbei natürlich verloren. Als Modellvorstellung hat man nun ein System Boolescher Gleichungen. Das Zeitmodell wird durch die reellen Zahlen dargestellt. Elementarobjekte sind logische Gatter und Verbindungsleitungen. Beobachtbare Werte sind "Bits" in einer 2 - 4 wertigen Logik. Als typische dedizierte Beschreibungssprache mag DISIM /JÄ1/ dienen.

Ebene 3: Schalter-Ebene

Die Schalter-Ebene ergibt sich einerseits durch Expansion von logischen Gattern zu Transistorschaltungen, zum anderen aber auch durch Schaltungen, die kein Äquivalent in der Gatterebene haben (z.B. Ausnutzung kapazitärer Effekte zur Speicherung). Die Modellvorstellung ist die eines endlichen Automaten, dessen Zustand im Wesentlichen durch die Ladungsverteilungen auf den Kapazitäten gegeben ist. Das Zeitmodell wird durch die reellen Zahlen dargestellt. Elementarobjekte sind Transistoren verschiedenen Typs und "Knoten" (Kapazitäten). Beobachtbare Werte sind Kapazitäten mit Interpretation, meist in einem diskretisierten Wertebereich. Als typische dedizierte Beschreibunssprache mag MOSSIM /BR1/ gelten.

Ebene 2: Symbolisches Layout

Diese Ebene unterscheidet sich aus Verhaltenssicht nicht von der Schalterebene. Es wird nur Strukturinformation hinzugefügt: Relative Lage der Transistoren zueinander, Datierungstyp der Verbindungswege. Als typische dedizierte Sprache für diese Ebene mag HILL /LM1/ dienen.

Ebene 1: Layout

Diese Ebene entspricht aus Verhaltenssicht der elektrischen Ebene. Die Prozessinformation wird als globale Information betrachtet, sodaß nur die Geometrieinformation angegeben werden muß. Als typische dedizierte Sprache für diese Ebene mag CIF dienen.

Ebene 0: Elektrische Ebene

Auf dieser Ebene ist die digitale Interpretation eines Schaltkreises auf die analoge Wirkungsweise zurückgeführt. Die Modellvorstellung ist die eines Systems von Differentialgleichungen mit einer kontinuierlichen Vorstellung von Zeit und beobachtbaren Werten. Die Elementarobjekte sind Induktivitäten, Kapazitäten, Resistoren. Als typische dedizierte Sprache für diese Ebene mag SPICE /VL1/ dienen.

3. Modellierung von Einzelkomponenten

Eine weit verbreitete Technik, das dynamische Verhalten von Systemkomponenten zu beschreiben, ist die Modellierung durch endliche Automaten. Diese Technik wird in verschiedenen Ausprägungen auf der elektrischen Ebene (z.B. DOMOS /SI1/, MOTIS /CGK/), auf der Schalterebene (z.B. MOSSIM /BR1/, CAP/DSDL /RA1/) und auf den höheren Ebenen benutzt. Natürlich werden sehr unterschiedliche Bestimmungsgrößen für den jeweiligen Zustand und sehr unterschiedliche Überführungsfunktionen benutzt. Dies soll an drei Beispielen erläutert werden:

3.1 Ein Modell für die Gatter-Ebene: QRBF

Quasi Reale Boolesche Funktionen (QRBF) wurden vom Autor vor etwa 10 Jahren als hinreichend exaktes Modell für das zeitlich-logische Verhatlen logischer Gatter entwickelt.Später wurde noch eine fünfwertige Variante /RA2/ vorgestellt.

Sei V der Wertebereich des Modells an Meßpunkten. Die Grundidee der QRBF ist es, ein Gatter nicht durch eine Funktion $f : V^n \to V$ zu modellieren sondern durch eine Funktion $f' : (V^n)^m \to V$, $f'(a_{1,t-m}, \ldots a_{m,t-m}, a_{1,t-m+1}, \ldots, a_{n,t}) = e_t$. D.h. ein Ergebnis zum Zeitpunkt t wird auf der Basis der Argumentwerte über einen gewissen Zeitraum der Vergangenheit berechnet. Dabei wird die Zeit in diskreten Schritten modelliert. Je nach zugrundegelegtem Wertebereich lassen sich nun reine Verzögerung, träge Verzögerung, Schalt- und Laufzeiten, Flankenverschleifungen u.ä. modellieren. Aus technischen Gründen setzt man die Funktion f' dabei aus mehreren Subfunktionen zusammen.

Statt auf die Argumentwerte der letzten m Zeitschritte zuzugreifen, kann man diese Werte natürlich auch als internen Zustand des Gattermodells speichern. Man erhält damit einen endlichen Automaten. Die oben genannten zeitlichen Effekte werden dann durch eine geeignete Zustandsüberführungsfunktion dargestellt. Ein dem QRBF-Modell ähnliches Modell des Zeitverhaltens liegt der CONLAN-Sprachfamilie /CN1/ zugrunde.

3.2 Ein Modell für die Schalterebene: MOSSIM

In MOSSIM /BR1, BR2/ wird von einem Netzwerk ausgegangen, das aus Knoten und Transistoren besteht. Jedem Knoten kann eine Größe zugewiesen werden, um seine Speicherfähigkeit für Ladung zu beschreiben. Als mögliche Zustände werden 0,1 und x vorgesehen, wobei hier x als eigenständiger Wert betrachtet wird. Transistoren werden als Schalter für bidirektionale Leitungen interpretiert. Je nach Typ und Wert des mit dem "gate"-Eingang verbundenen Knotens ist ein Transistor im Zustand "offen", "geschlossen" oder "unbekannt". Jedem Transistor kann eine Stärke zugewiesen werden, um seine Leitfähigkeit zu modellieren. Mit diesen Annahmen kann man aus jedem Zustand den Folgezustand errechnen, wenn man annimmt, daß bis zum Erreichen dieses Folgezustands sich die Transistorzustände nicht ändern ("unit delay"-Modell). Die Übergangsfunktion läßt sich recht elegant aufschreiben, wenn man die Menge der Werte an Knoten als Elemente eines Verbandes auffaßt, der nach Signalstärken geordnet ist. Bryant leitet dabei die Signalstärke bei Knoten, die aktuell nicht mit einem Eingang verbunden sind, von der Knotengröße ab, bei solchen Knoten, die aktuell mit einem Eingang verbunden sind, durch den maximalen Transistorwiderstand auf dem Weg vom Eingang zum Knoten.

3.3 Ein Modell für die elektrische Ebene: DOMOS

DOMOS /SI1/ ist ein Simulationssystem für MOS-Schaltungen, das von der Zustandsvariablen-Technik /KR1/ Gebrauch macht. Die Elemente des Zustandsvektors zu einem Zeitpunkt t sind die Spannungen (oder die Ladungen) der Kapazitäten und die Ströme (oder die magnetischen Flüsse) der Induktivitäten. Bei bekannten Schaltungsparametern können daraus dann mit einem algebraischen Gleichungssystem die Spannungen und Ströme durch alle nicht speichernden Bauelemente berechnet werden. Hat man all diese Werte, so lassen sich die Kapazitäten und die Spannungen an den Induktivitäten berechnen. Integriert man diese Werte über einen Zeitschritt, so erhält man die Änderung des Zustandsvektors, mit der dann auf der Basis des alten Zustandsvektors der neue berechnet werden kann.

Für die Integration könnte man sich auf eine einfache "Forward Euler" Integration beschränken, d.h. einfach die Anfangswerte über den Zeitraum integrieren. DOMOS benutzt etwas kompliziertere Verfahren, um den Simulationsfehler geringer zu halten.

Die simulierte Schaltung muß man sich um ideale Spannungsquellen erweitert denken, die die " Eingaben" in das zu simulierende Netzwerk darstellen (Anregungsvektor).

Die Übergangsgleichungen von DOMOS sollen hier nicht behandelt werden. Sie sind z.B. in /SI1/ nachzulesen. Einen Überblick über alternative Simulationsverfahren auf der elektrischen Ebene gibt /SI2/.

4. Kompositionstechniken

Zur Modellierung von Einzelkomponenten ist das Modell des endlichen Automaten gut geeignet, wie oben anhand dieser Beispiele demonstriert wurde. Die einfachste Methode, aus Komponentenmodellen ein Globalmodell zu erstellen, ist die Konstruktion eines Produktautomaten aus den Einzelmodellen anhand der Verschaltung des zu modellierenden Systems.

Dazu muß nur das Kreuzprodukt der einzelnen Zustandsvektoren als globaler Zustandsvektor benutzt werden und die Überführungsfunktion δ muß aus den einfachen Überführungsfunktionen unter Berücksichtigung der Verschaltung konstruiert werden. Diese Konstruktion ist problemlos und soll hier nicht näher erläutert werden.

Da diese Konstruktion so einfach ist, wurde sie früher für Simulatoren auf der Gatterebene gern benutzt. Auf anderen Abstraktionsebenen wird sie bis heute eingesetzt. DOMOS und MOSSIM mögen als Beispiele für Simulatoren, die auf dieser Technik basieren, dienen.

Aus zwei Gründen ist dieser Ansatz jedoch problematisch:

a) Es muß unterstellt werden, daß für alle Komponenten des zu simulierenden Systems im modellierenden Automaten eine Zustandsfortschaltung im selben "Zeitraster" erfolgt (erfolgen kann) (Synchronieproblematik).
b) In der Regel wird zu einem Zeitpunkt ein System nur punktuell angeregt (wenige Komponenten des Eingabevektors werden verändert). Diese punktuelle Anregung hat in der Regel auch nur eine Folge von punktuellen Wertänderungen im Zustandsvektor zur Folge. Da jedoch stets die gesamte Zustandsüberführungsfunktion berechnet wird, wird bei den meisten Iterationen für die meisten Komponenten des Zustandsvektors die Identität als Überführungsfunktion berechnet (Lokalitätsproblematik).

Die Synchronieproblematik läßt sich relativ leicht in den Griff bekommen, wenn man nur Objekte verschaltet, für die eine gemeinsame Auflösung der Beobachtung angemessen erscheint. Dies führt natürlich zu Simulatoren, die dediziert für eine spezielle Abstraktionsebene sind. Es kann darüber hinaus zu spezifischen Einschränkungen bzgl. der modellierbaren Systeme führen. Ein Musterbeispiel für derartige Einschränkungen ist die Beschränkung vieler RT-Simulatoren auf getaktete Systeme. Hier liefert der Systemtakt des zu simulierenden Systems die globale Synchronisation des Modells.

Eine Steigerung der Effizienz erhält man auch dadurch, daß man die Zeitfortschaltung nicht in festen Inkrementen vornimmt, sondern situationsabhängig. So müssen beispielsweise lineare Abhängigkeiten nicht mehrfach berechnet werden. Dient eine lineare Abhängigkeit zur Approximation einer höherwertigen, so kann als jeweiliger Zeitschritt der größte gewählt werden, bei dem ein vereinbarter Fehler nicht überschritten wird. Da Fehlerabschätzungen in diesem Zusammenhang relativ einfach sind, wird das Konzept der dynamsichen Schrittweitenberechnung bei Simulatoren der elektrischen Ebene häufig eingesetzt (z.B. DOMOS /SI1/). Die Lokalitätsproblematik läßt sich im Rahmen des Produktautomatenansatzes kaum vernünftig lösen. Man kann nur "hoffen", daß die zu simulierende Schaltung derart vermascht ist, daß sich Anregungen relativ breit verteilen. Sinnvoll ist dieser Ansatz daher m.E. auch nur dort, wo dies noch am ehesten gewährleistet ist und wo sich auch die Synchronieproblematik noch in vertretbarer Weise lösen läßt, d.h. auf der Ebene der elektrischen Simulation.

Beide genannten Problematiken lassen sich lösen, wenn man den Zustand als gespeicherte Basisinformation ersetzt durch das Ereignis, d.h. eine Zustandsänderung. Ausgehend von der Schaltungsbeschreibung lassen sich für jedes Ereignis seine auslösenden Komponenten ("influencer") und unmittelbar beeinflußte Komponenten ("influencees") bestimmen.

Anstelle einer strikten Kopplung von Einzelmodellen durch Integration in ein Globalmodell hat man nun eine lose Kopplung von Einzelmodellen über die Influencer/Influencee-Kopplung. Eine stabile Komponente bleibt passiv bis sie durch ein Ereignis (Wertänderung an mindestens einer Komponente des Eingabevektors) angeregt wird. Diese Anregung kann eine oder mehrere Wertänderungen an Komponenten des lokalen Zustandsvektors zur Folge haben. Diese Zustandsüberführungen können zu Zeitpunkten einer lokalen Zeitfortschaltung stattfinden und Wertänderungen an Komponenten des Ausgabevektors zur Folge haben. Für derartige Komponenten mit Wertänderungen werden aus der Schaltungsbeschreibung die Indluencees bestimmt und für diese zum entsprechenden Zeitpunkt Ereignisse eingeplant.

Man sieht, daß sowohl die Synchronieproblematik wie auch die Lokalitätsproblematik vollständig gelöst sind. Jede Komponente kann mit ihrer lokalen Zeitfortschaltung arbeiten und an die Stelle der globalen dynamischen Schrittweitenanpassung tritt nun die lokale dynamische Anpassung. Da für jedes Ereignis die betroffenen Influencees bestimmt werden und nur diese aktiviert werden, wird die Lokalitätsproblematik optimal gelöst. Man zahlt durch einen höheren Basisaufwand, der sich im wesentlichen dadurch ergibt, daß man neue zukünftige Ereignisse in die nach Zeitpunkten vollständig sortierte Liste der zukünftigen Ereignisse einsortieren muß. Als einmaliger Aufwand ist auch noch die Bestimmung der Influencer/Influencee-Beziehungen, die sich jedoch unmittelbar aus der Schaltungsbeschreibung ableiten läßt, zu nennen.

Ein typischer Vertreter dieser Art von Simulatoren ("event scheduling") ist auf der Gatterebene TEGAS /SZ1/. Als "multilevel" Simulator (und nur über "event scheduling" lassen sich .E. "multilevel" Simulatoren effizient implementieren) ist CAP/DSDL als Beispiel für diese Klasse von Simulatoren zu nennen.

5. Simulationsprototypen

Die Wirkungsweise der beiden vorgestellten Simulationstechniken soll anhand von Simulator-Prototypen verdeutlicht werden. (Einen umfassenden Einblick in Simulationstechniken vermittelt /ZE1/.)

Sei X_t der Zustand des Simulationsmodells zum (simulierten) Zeitpunkt t, gegeben durch die Werte seiner Komponenten (Darstellungsvariable) $x_{1,t}$ ..., $x_{m,t}$. Sei h die (hier als fest angenommene) Schrittweite, angegeben in zu simulierender Zeit, t_o der Startzeitpunkt, $t_f = t_{o+nh}$ der Endezeitpunkt.

Mit X_{t_o} wird der Anfangszustand bezeichnet. Faßt man die Eingabemuster als Beschreibung des Moduls "Ambiente" auf und geht davon aus, daß der gesamte Zustandsvektor beobachtet wird, die Ausgabefunktion also einfach die Identität auf dem Zustand ist, so erhält man einen sehr einfachen Simulationsprototyp.

```
while t ≤ tf do
   begin
      for all xi do
         xi' := fi(X)
      t := t + h ;
      for all xi do
         xi := xi'
   end
```

Man beachte, daß die Wertzuweisungen an die Darstellungsvariablen über einen Zwischenpuffer geschehen. Dadurch ist sichergestellt, daß nur Werte zum Zeitpunkt t für die Berechnung zum Zeitpunkt $t + h$ eingehen. Der Algorithmus wird damit auch unabhängig von der Anwendungsreihenfolge der f_i.

Will man noch eine dynamische Schrittweitenberechnung vornehmen, so erweitert sich der Algorithmus zu:

```
while t ≤ tf do
   begin
      h := step (error) ;
      for all xi do
         xi' := fi(X,h) ;
      t := t + h ;
      error := errorfkt (X,X',h) ;
      for all xi do
         xi := xi'
   end
```

Hier muß natürlich ein initialer Fehler vorgegeben werden, z.B. $\emptyset$.

Für den Simulatorprototyp für die ereignisgesteuerte Simulation sei angenommen, daß Ereignisse zeitlich sortiert in einer Warteschlange stehen. Jedes Ereignis ist von der Form $E_{K,t} = (K,t,V)$, d.h. die Komponente K bekommt zum Zeitpunkt t den Wert V zugewiesen. Für jede Komponente K des Zustandsvektors wird folgende Information gespeichert.

Influencees$_K$

$B_k = \beta_k(bx_{k_1}, \ldots, bx_{k_m})$ Ausführbarkeitsbedingung
$A_k = \alpha_k(ax_{k_1}, \ldots, ax_{k_n})$ lokale Zustandüberführung
$D_k = \delta_k(dx_{k_1}, \ldots, dx_{k_l})$ lokale Verzögerung

Mit diesen Annahmen läßt sich ein Simulatorprototyp für ereignisgesteuerte Simulation wie folgt aufschreiben:

```
begin
   t := ∅
   while t ≤ tf do
       begin
         extract (ke,te,ve) with t minimal from queue ;
         t := te ;
         xk:= ve ;
         for all k' elem influenceesk do
            if Bk' satisfied then
               begin
                  ve' := Ak' ;
                  te' := t + Dk' ;
                  insert (k'e ,te',ve') into queue
               end
       end
end
```

Auch dieser Basisalgorithmus läßt sich noch verfeinern, wobei hauptsächlich darauf geachtet wird, keine "überflüssigen" Ereignisse zu produzieren (z.B. Wertzuweisungen, die keine Änderung bedeuten).

6. Simulation der Schalterebene mit ereignisgesteuerter Simulation

Die Simulationstechnik, die hier vorgestellt werden soll, arbeitet mit einem Potenzmengenmodell für die beobachtbaren Werte an Netzwerkknoten. Dieses Wertemodell ist für die eigentliche Simulationstechnik nicht zwingend, erscheint aber sehr adäquat.

Das Potenzmengenmodell geht von einer endlichen Menge eigentlicher Werte aus, unbestimmte Werte werden durch Angabe aller möglichen Werte dargestellt. Alle Operationen werden nun auf derartigen Teilmengen (die im Falle von eigentlichen Werten einelementig sind) definiert. Bei der üblichen Mengendarstellung durch Bitvektoren lassen sich alle diese Operationen auf logisches Und und Oder zurückführen, sodaß alle Alternativen ohne zeitlichen Zusatzaufwand simultan durchgerechnet werden (und beim üblichen Instruktionsvorrat auch sehr effizient). In /LR1/ wird von folgender Menge V eigentlicher Werte ausgegangen:

L0 (niederohmig Null)	L1 (niederohmig Eins)
M0 (abgeschwächte Null)	M1 (abgeschwächte Eins)
H0 (hochohmige Null)	H1 (hochohmige Eins)
Z (hochohmig ohne Wertinterpretation)	

Auf V wird die offensichtliche Halbordnung definiert. Mit dieser Ordnung lassen sich nun die beiden Basisoperationen der Modellierung definieren:

a) Sup-Funktion (Basisfunktion für Werteverhalten an Knoten)

$$\forall a, b \in V : \sup(a,b) = \text{if } a \le b \text{ then } b \text{ else if } b \le a \text{ then } a \text{ else } \{a,b\}$$

$$\forall A, B \in 2^V : \sup(A,B) = \bigcup_{\substack{a\in A \\ b\in B}} \sup(a,b)$$

b) $\dagger$-Funktion (Basisfunktion für Verhalten sperrender Transistoren)

$$\forall a \in V : \dagger(a) = \text{if } a \in \{H1,M1,L1\} \text{ then } H1 \text{ else if } a \in \{H0,M0,L0\} \text{ then } H0 \text{ else } Z$$

$$\forall A \in 2^V : \dagger(A) = \bigcup_{a\in A} \dagger(a)$$

Ein Feldeffekttransistor als bidirektionales Schaltelement ist zunächst in einen unidirektionalen Simulationsalgorithmus schwer einzubinden. Dies läßt sich aber relativ einfach lösen, wenn man die bidirektionalen Anschlüsse jeweils durch Paare (Eingang, Ausgang) modeliert. Aus technischen Gründen ist es zudem notwendig, zwischen den Werten von benachbarten Knoten einschließlich des Eigenanteils a.in.me, b.in.me und ohne den Eigenanteil a.in.oe, b.in.oe zu unterscheiden.

Man erhält damit beispielsweise für einen positiven Schalter PS (z.B. n-Kanal MOSFET) folgendes Modell:

$PS : (2^V)^5 \rightarrow (2^V)^2$
PS (gate,a.in.oe, a.in.me, b.in.oe, b.in.me) = (a.out, b.out)
= if gate $\in$ {LO,MO,HO,Z} then ($\downarrow$(b.in.me), $\downarrow$(b.in.in.me))
else (b.inoe, a.in.oe)

Ein Knoten nd mit Ladungszerfallszeit n kann modelliert werden durch:

$nd : ((2^V)^2)^n \rightarrow 2^V$
$nd(a_{t-n}, b_{t-n}; a_{t-n+1}, b_{t-n+1}; \ldots; a_t, b_t)$
= if(for all $t' \in [t-n+1, t]$: sup $(a_{t'}, b_{t'}) \in$ {H1,HO,Z}
then Z else sup(a_t, b_t)

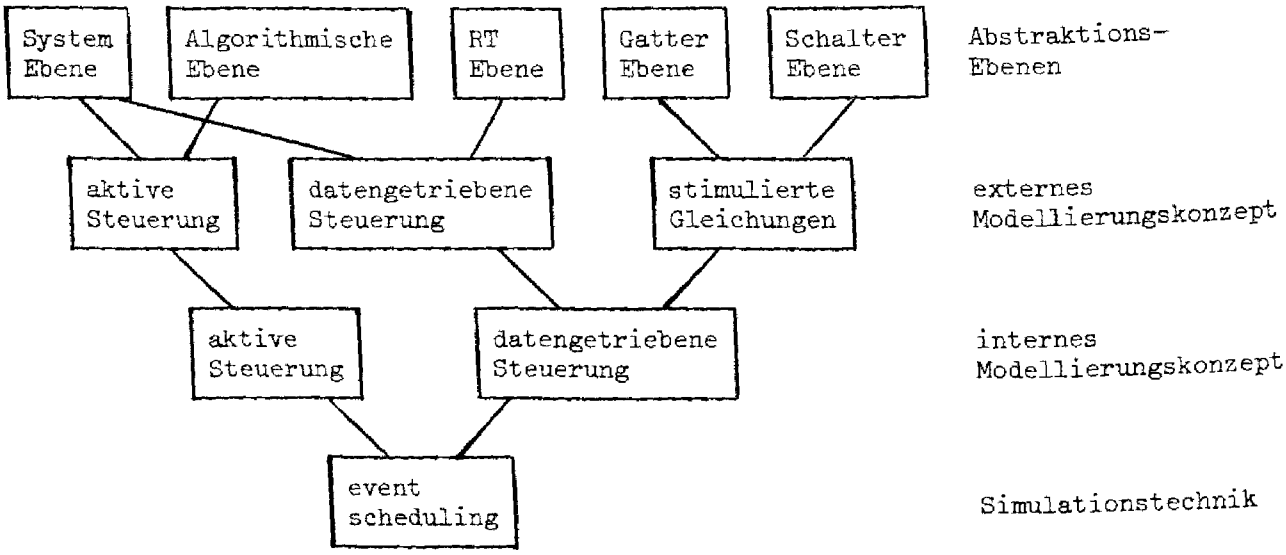
Diese Modelle lassen sich sehr einfach mit ereignisgesteuerter Simulation behandeln. Somit lassen sich sowohl sehr effiziente Schalterebenen-Simulatoren bauen wie auch diese Abstraktionsebene in Mehrebenen-Simulatoren integrieren.

Nicht ganz trivial ist die Umsetzung einer Transistorschaltung in eine "Schaltung" (Strukturbeschreibung) aus den modellierenden Objekten. In /MU1/ wird dieser Transformationsalgorithmus mit Hilfe einer Graph Grammatik spezifiziert.

7. Ein Beispiel für einen "multilevel" Simulator: CAP/DSDL

CAP/DSDL soll hier als Beispiel für einen komplexen "multilevel" Simulator dienen. (In Rahmen dieser Abhandlung wird nur auf den Simulationsaspekt von CAP/DSDL eingegangen, andere Systemkomponenten werden bewußt ausgeklammert.)

Die Hardware Beschreibungs Sprache CAP/DSDL überdeckt den Bereich von Systemebene bis Schalterebene. Somit ist der CAP-Simulator als Laufzeitsystem für diese Sprache ein "multilevel" und "mixed level" Simulator. Die für das genannte Ebenenspektrum genannten externen Modellierungskonzepte "aktive Steuerung", "datengetriebene Steuerung" und "stimulierte Gleichungen" werden auf zwei interne Modellierungskonzepte "aktive Steuerung" und "datengetriebene Steuerung" zurückgeführt. Diese beiden internen Konzepte schließlich werden abgebildet auf ereignisgesteuerte Simulation als Simulationstechnik. Man erhält somit den folgenden konzeptionellen Aufbau des CAP-Simulators:



Die Abbildung einer datengetriebenen Steuerung auf ereignisorientierte Simulation ist eine wohlbekannte Technik, zumindestens solange man sich auf der Gatterebene bewegt. Die Schalterebene wird integriert, indem man das in Kapitel 6 beschriebene Verfahren wählt.

Das Simulationssystem ist in mehreren Schalen aufgebaut. Den Kern bildet die Modellierung atomarer Ereignisse (z.B. Abarbeitung von Ausdrücken). Dies wird durch direkt ausführbaren Code geleistet. Darauf sitzt ein ereignisgesteuerter Simulator für die datengetriebene Steuerung. Dieser wird überlagert von einem weiteren ereignisgesteuerten Simulator für die aktive Steuerung. Der Gesamtalgorithmus ist selbstadaptierend, d.h. die aktuell benötigten Systemteile werden aktiviert.

Literatur

/BA1/ M.R. Barbacci et al.
The ISPS Computer Description Language
Technical Report, Dept. of Computer Science, Carnegie Mellon University, 1977

/BR1/ R.E. Bryant
MOSSIM: A Switch-Level Simulator for MOS-LSI
Proceedings 18th Design Automation Conference, 1981

/BR2/ R.E. Bryant
A Switch-Level Model of MOS-Logic Circuits
Proceedings VLSI 81, Edinburgh 1981

/CH1/ Y. Chu
Introducing CDL
IEEE Computer, Dec. 1979

/CGK/ B.R. Chawla, H.K. Gummel, P. Kozak
MOTIS - A MOS Timing Simulator
IEEE T-CAS-22, Dec. 1975

/CD1/ R. Piloty, M. Barbacci, D. Borrione, D. Dietmeyer, F. Hill, P. Shelly
CONLAN Report
Springer, 1983

/IN1/ Inmos
Occam Programming Manual
Prentice Hall, 1984

/JÄ1/ U. Jäger
Logik- und Fehlersimulation mit dem Programmsystem DISIM
Seminarunterlagen Praxis der Großintegration, Abt. Elektrotechnik, Universität Dortmund, 1983

/KR1/ E.S. Kuh, R.A. Rohrer
The State-Variable Approach to Network Analysis
Proc. IEEE, Vol. 53, July 1965

/LM1/ T. Lengauer, K. Mehlhorn
The HILL System: A Design Environment for the Hierarchical Specification, Compaction, and Simulation of Integrated Circuit Layouts
TR A 83/04 , SFB 124, B2 Univ. des Saarlandes, Saarbrücken, 1983

/LR1/ K.-D. Lewke, F.R. Rammig
Description and Simulation of MOS Devices in Register Transfer Languages
Proceedings VLSI '83, Trondheim, 1983

/RA1/ F.J. Rammig
CAP/DSDL: Preliminary Language Reference Manual
Forschungsbericht Nr. 129, Abt. Informatik, Universität Dortmund, 1980

/RA2/ F.J. Rammig
Five Valued Quasi Real Boolean Functions
Proceedings 5th European Meeting on Cybernetics and System Research, Wien, 1980

/SI1/ H. Sibbert
Modellierung und Netzwerkanalyseprogramm für MOS-Schaltungen mit hoher Leistungsfähigkeit
Diss. Univ. Dortmund, 1977

/SI2/ H. Sibbert
Verfahren und Programme für die Schatlungs- und Timing-Simulation
Seminarunterlagen Praxis der Großintegration, Abt. Elektrotechnik, Univers. Dortmund, 1983

/SIM/ O. Belsnes
The Use of SIMULA for Real-Time System Implementation
Norwegian Computing Center, Oslo, 1978

/SZ1/ S.A. Szygenda
TEGAS 2 - Anatomy of a general purpose generation test generation and simulation system for digital logic
Proceedings Design Automation Workshop, 1972

/VL1/ A. Vladimirescu, S. Liu
The Simulation of MOS Integrated Circuits
Using SPICE 2
Memo UCB/ERLM 80/7, Univ. of Calif., Berkley, 1980

/ZE1/ B. Zeigler
Theory of Modelling and Simulation
Wiley & Sons (1976)

/MU1/ E. Muliyanto
Logiksimulation für MOS-Schaltungen
Diplomarbeit Universität Dortmund, 1984