Hierarchischer Entwurf komplexer Systeme

Günter Hotz und Armin Reichert

Der Aufbau von Rechnern wird in der Regel durch Blockdiagramme erläutert und für einzelne Blöcke dieser Diagramme wie z.B. Prozessoren geschieht das auf gleiche Weise. So tut man dies in schrittweiser Verfeinerung, bis man an die Grenze des Interesses des Lesers stößt. Diese Grenze mag bei den Gattern liegen, sie kann aber auch darüber oder darunter liegen. Beim Chipentwurf interessiert man sich eventuell auch für den Aufbau der Gatter aus Transistoren oder gar für physikalische Eigenschaften der Transistoren.

Dieses Schema zur Erklärung komplexer Systeme finden wir überall. Lehrbücher sind nach diesem Schema aufgebaut, wie die Inhaltsverzeichnisse zeigen, und Theorien sind es auch. Natürlich ist das nicht die einzige Weise sich in großen Systemen zu orientieren, wie das Beispiel von Lexika zeigt.

Hierarchische Strukturen spielen in großen Systemen eine sehr wichtige Rolle. Aus diesem Grund sollte man dort, wo man solche Systeme in Computern modellieren oder repräsentieren will oder die Entwicklung der Zustände solcher Systeme verfolgen will, Werkzeuge bereitstellen, die aus diesen Strukturen Vorteil ziehen.

Die übliche Vorgehensweise komplexe Systeme durch Blockdiagramme zu erläutern, verwendet wesentlich die zwei Dimensionen, die die Tafel zur Verfügung stellt. Das soll heißen, daß man diesen offensichtlichen Vorteil für das Verständnis der Mitteilung durch eine mühsame Umsetzung der zweidimensionalen „Formeln" in lineare Ausdrücke nicht wieder zerstören sollte. Vielmehr sollte diese zweidimensionale Struktur direkt vom Rechner erkannt und umgesetzt werden.

Wir werden hier eine Sprache 2dL vorstellen, die wesentlich zweidimensional angelegt ist. Diese Sprache dient zur Eingabe von Schaltkreisentwürfen in das System CADIC zur Erzeugung des Layouts von Chips, die diese Schaltkreise repräsentieren. Wir werden zunächst die Konzepte dieser Sprache entwickeln und an Beispielen erläutern. Wir kommen dann auf Probleme zu sprechen, die mit der Interpretation dieser Sprache und mit der Erzeugung von Layout verbunden sind. Diese Probleme führen zur Entwicklung von Werkzeugen, die dem Menschen erlauben, in den Layoutprozeß einzugreifen und eventuell auch den Entwurf zu modifizieren.

Im letzten Kapitel kommen wir zurück auf die Frage nach der Übertragbarkeit des im Fall des Hardware-Entwurfs erfolgreichen Systems auf die oben angedeuteten allgemeineren Fälle.

8.1 Die Elemente der Sprache 2dL

Die Elemente der Sprache sind Rechtecke und Polygonzüge in der Ebene. Die Rechtecke und die Polygonzüge tragen Beschriftungen. In der Ebene sind zwei

I. Wegener (ed.), *Highlights aus der Informatik*
© Springer-Verlag Berlin Heidelberg 1996

Richtungen „Norden“ und „Westen“ ausgezeichnet. Im allgemeinen werden diese Richtungen durch die Begrenzung des Bildschirms vorgegeben. Alle Rechtecke sind kantenparallel zu diesen Richtungen und paarweise durchschnittsfremd in die Ebene eingebettet. Auf den Seiten der Rechtecke sind „Anschlüsse“ angebracht. Das sind Punkte, die Markierungen tragen.

Die Beschriftungen der Rechtecke sind Wörter über einem Alphabet wie z.B. $A, B, \mathrm{Ad}, \mathrm{Mult}, \&, \vee, \ldots$ oder mit Parametern versehene Namen wie $\mathrm{Ad}[n]$, Ad_n, $\mathrm{Mult}[2^n]$, $\mathrm{FloatingAd}[n]$.

Die Anschlüsse s auf dem Rand der Rechtecke tragen zwei Markierungen $\omega(s)$ und $\tau(s)$. Die Markierung $\omega(s)$ gibt an, ob s ein „Eingang“ ($\omega(s) = 1$) oder ein „Ausgang“ ($\omega(s) = -1$) oder ohne Orientierung ($\omega(s) = 0$) ist. In der graphischen Eingabe wird ω auch durch Strecken definiert, die in den Anschlüssen auf der Seite senkrecht stehen und die durch einen Pfeil im Falle $\omega(s) \neq 0$ eine Orientierung ausweisen.

Die zweite Markierung $\tau(s)$ gibt den Leitungstyp an. Die Leitungstypen können auch parametrisiert sein. So bezeichnet $\tau(s) = \underline{\mathrm{bool}}^n$ einen Anschluß für n boolesche Leitungen. Es kann auch mehrere Varianten des Typs $\underline{\mathrm{bool}}$ geben, die etwa Leitungen in verschiedenen Schichten repräsentieren. Markierungen können auch variabel sein und ihren Typ erst im Zusammenspiel mit anderen Anschlüssen zugewiesen bekommen. Ein Beispiel eines Bausteins zeigt Abbildung 8.1.

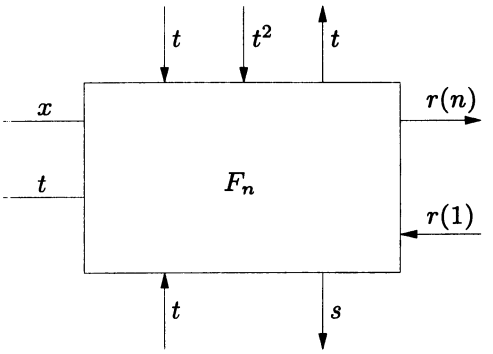

Abbildung 8.1.

Die Rechtecke können als Funktionen oder Relationen interpretiert werden, die Anschlüsse als lokale Variablen, die Werte von außen annehmen oder nach außen abgeben, oder die je nach Situation das eine oder das andere tun.

Die Namen der Rechtecke bestimmen den Baustein eindeutig, sie werden also in der Sprechweise der Programmiersprachen „global“ verwendet. Die Namen der Anschlüsse sind lokal für jeden Baustein zu verstehen. Natürlich macht es auch Sinn, „Lokalität“ bei den Bausteinnamen zu betrachten. Für das Ziel dieses Beitrags ist das aber nur von sekundärem Interesse.