Zeitschrift: Bulletin des Schweizerischen Elektrotechnischen Vereins, des Verbandes Schweizerischer Elektrizitätsunternehmen = Bulletin de l'Association suisse des électriciens, de l'Association des entreprises électriques suisses

Herausgeber: Schweizerischer Elektrotechnischer Verein ; Verband Schweizerischer Elektrizitätsunternehmen

Band: 74 (1983)

Heft: 5

Artikel: CAD für integrierte Schaltungen

Autor: Hottinger, W.

DOI: https://doi.org/10.5169/seals-904773

CAD für integrierte Schaltungen

W. Hottinger

Die stürmische Entwicklung der Technologie für integrierte Schaltungen (IC) führt dazu, dass die bestehenden Möglichkeiten aus Mangel an geeigneten CAD-Werkzeugen kaum mehr ausgeschöpft werden können. Ein CAD-System, welches sich durch Designsicherheit, hohe Interaktivität und eine hierarchische Struktur auszeichnet, kann die Machbarkeit von IC in der Zukunft gewährleisten; allerdings braucht es dazu eine veränderte Arbeitsweise der Schaltungsentwickler sowie eine neue Generation von CAD-Werkzeugen.

Par suite du manque d'outils appropriés pour la conception assistée par ordinateur (CAO), il n'est plus guère possible d'épuiser les possibilités de l'évolution extrêmement rapide de la technologie des circuits intégrés. Un système de CAO qui se distingue par sa sûreté de conception, sa grande interactivité et sa structure hiérarchique, peut assurer désormais la confection de ces circuits. Toutefois, il faut un autre mode de travail des concepteurs ainsi qu'une nouvelle génération d'outils de CAO.

Dieser Aufsatz entspricht dem Vortrag des Autors anlässlich des «Fall Meeting 1982 on Computer Aided Design (CAD)» der IEEE Swiss Section, Chapter on Solid State Devices and Circuits, am 19. Oktober 1982 in Bern.

Adresse des Autors
W. Hottinger, Faselec AG, Räffelstrasse 29, 8045 Zürich.

1. Tendenzen der VLSI-Entwicklung

Die maximale Anzahl Komponenten auf einer integrierten Schaltung hat sich in den letzten 20 Jahren im Durchschnitt pro Jahr beinahe verdoppelt und erreicht heute über 100 000 Transistoren pro Chip [1]. Diese Komplexität wurde einerseits durch immer kleinere Geometrien erreicht (Linienbreiten von 2 bis 4 µm sind heute üblich), anderseits durch immer grössere Chipflächen (40 bis 80 mm²). Die Analyse der technischen Möglichkeiten und Grenzen zeigt, dass eine weitere Steigerung der Integrationsdichte um zwei Zehnerpotenzen im Bereich des Möglichen liegt [2].

Der Entwicklungsaufwand für moderne Mikroprozessoren beträgt zurzeit ein bis einige Dutzend Mannjahre. Extrapolationen ergeben, dass in Zukunft mit einigen Hundert, ja einigen Tausend Mannjahren gerechnet werden muss [3]. Offensichtlich kann eine solche Entwicklung nicht stattfinden, die angestellten Überlegungen machen jedoch klar, dass der Fortschritt in Schaltungsintegration nicht mehr nur durch die Technologie begrenzt ist, sondern vielmehr durch die Produktivität der Entwickler.

2. Kritik der heutigen CAD-Systeme

Die heutigen Methoden zur Entwicklung von integrierten Schaltungen entstanden in der Zeit, wo ein IC einige zehn bis einige hundert Elemente umfasste. VLSI (Very Large Scale Integration) darf aber nicht einfach als «grosser IC» verstanden werden, sondern verlangt eine veränderte Design-Methodik und eine Anpassung an die neuen Randbedingungen, welche notabene durch die Halbleiter geschaffen wurden.

Gegen 1970 entstanden die ersten CAD-Programme für IC Design. Es handelte sich um Programme zur Schaltungsanalyse, um Logiksimulatoren und um Layoutprogramme. Bald darauf folgten Programme, welche geometrische und elektrische Designfehler entdecken konnten (DRC: Design Rule Checking; ERC: Electrical Rule Checking). Sie waren, der Zeit entsprechend, auf optimalen Einsatz der teuren Computer zugeschnitten, aber kaum auf die Bedürfnisse der Benutzer. Die Eingabe geschah mittels Lochkarten, die Verarbeitung erfolgte «Batch»-weise, und die Resultate wurden in Tabellenform dargestellt. So vertauschte der Designer seinen Laborplatz mit einem Kartenstanzer. Mit der Zeit lösten interaktive graphische Editoren die Layoutprogramme ab, alle anderen Programme blieben Batch, die meisten bis heute.

Alle Verbesserungen und Erweiterungen der Programme fielen zudem über kurz oder lang dem exponentiellen Wachstum der Schaltungskomplexität zum Opfer. Die strikte Trennung zwischen elektrischem Design einerseits und Layout anderseits – vor allem in den USA – zementiert die Struktur des CAD. Oft laufen die Programme auf verschiedenen Computern, und falls überhaupt Information von einem Programm zum nächsten weitergegeben werden kann, dann muss diese meistens verändert werden. Nur in wenigen Fällen wurde versucht, ein CAD-System zu schaffen, welches von der Simulation bis zur fertig getesteten Schaltung eine einzige Datenbasis benutzt und kompatible Schnittstellen zwischen den Designschritten hat, um so die Wahrscheinlichkeit von Designfehlern drastisch zu reduzieren.

Bei den meisten heute erhältlichen Systemen [4] kann die Identität von elektrischem und geometrischem Design erst am Layout durch Patternerkennung und Extraktion von elektrischer Information überprüft werden. Die nötigen Korrekturen müssen dann am fertig optimierten Layout erfolgen.

Es scheint, dass CAD für VLSI nicht durch Evolution der bestehenden Programme, sondern nur durch Revolution der Entwicklungsmethoden erreicht werden kann. Wenn man allerdings drei der wichtigsten Neuerungen der letzten Jahre betrachtet, zeigt sich, dass nur eine davon einen wirklichen Durchbruch erreichte.

1. Die Verwendung von regelmässigen Strukturen wie RAM (Random Access Memory), ROM (Read Only Memory) und PLA (Programmable Logic Array) könnte die Entwicklungszeiten in gewissen Grenzen halten, da solche Strukturen verhältnismässig einfach durch Computerprogramme generiert werden können. Obwohl diese Tatsache allgemein anerkannt wird, haben selbst die modernsten und komplexesten Schaltungen einen enttäuschend niedrigen Regularitätsfaktor [5]. (Als Regularitätsfaktor bezeichnet man das Verhältnis der Gesamtzahl der Elemente auf einer Schaltung zur Anzahl der manuell ausgelegten Elemente.)
2. Seit längerer Zeit wird die Automatisierung von Plazierung und Verbindung von einzelnen Layoutteilen untersucht. Obwohl sehr viel grundlegende Arbeit geleistet wurde und obwohl manche Resultate beeindruckend sind, ist das Echo enttäuschend gering. Automatisches Design liefert in den weitaus meisten Fällen wesentlich weniger dichte Layouts als manuelle Arbeit, zudem verschlechtert sich die Qualität stark mit zunehmender Komplexität.
3. Synchrone Logik bildet die Ausnahme in dieser Reihe. Obwohl asynchrone Logik die Zeit besser nutzen kann (ein Ereignis kann zu jeder beliebigen Zeit erfolgen, nicht nur als Folge eines Clocksignals wie in synchronen Systemen), haben die Schwierigkeiten, welche grosse asynchrone Systeme in der Entwicklung, im Austesten und in der Endkontrolle bereiten, dazu geführt, dass das Konzept der synchronen Logik weltweit verwendet wird.

Die wichtigsten Nachteile der gängigsten CAD-Systeme sind also: schlechte «man-machine interfaces»; ungenügende Designsicherheit wegen fehlender oder schlechter Programmkompatibilität; lange Zeiten für Fehlerrückmeldungen. Die ernsteste Beeinträchtigung liegt aber darin, dass die heutigen CAD-Werkzeuge keine angemessene Methodik unterstützen. Der Designer wird geradezu ermutigt, auf der niedrigst möglichen Ebene zu arbeiten: mit grafischen Editoren direkt am geometrischen Layout, mit Simulatoren an Transistoren oder Gattern. Werkzeuge, um abstrakte Darstellungen zu erfassen und zu manipulieren, fehlen weitgehend. Ein CAD-System, welches die Möglichkeit bietet, eine Aufgabe durch Gliederung in Teilaufgaben zu vereinfachen und übersichtlicher zu gestalten, existiert bis heute nicht.

3. Anforderungen an künftige CAD-Systeme

Die Ziele einer neuen Methodik für VLSI-Design müssen sein:

A) Ein möglichst grosses Mass an Designsicherheit wird gewährleistet.
B) Alle Entwicklungsstufen (vom Pflichtenheft bis zur Endkontrolle) sind beinhaltet.
C) Die Möglichkeiten der Technologie werden optimal ausgenützt.
D) Die Produktivität der Designer wird verbessert.
E) Die Befriedigung der Designer wird erhöht.
F) Die Einführung von effizienter CAD-Software ist möglich.

Am einfachsten lassen sich die nötigen Kompromisse erklären, wenn man die zwei Extreme der möglichen CAD-Systeme betrachtet:

3.1 Vollautomatisches CAD (Fig. 1A)

Alle Schritte zwischen den verschiedenen Stadien werden durch Computerprogramme vollautomatisch ausgeführt. Ein solches System (es wird als «Silicon-Compiler» bezeichnet) erfüllt A, B und D sehr gut, C sehr schlecht. Die Auswirkung auf E und F ist stark von der Implementierung abhängig. Silicon-Compiler sind heute erst im Forschungsstadium [6], bis zu ihrem Einsatz dürften noch einige Jahre vergehen.

3.2 Manuelles CAD (Fig. 1B)

Alle Schritte werden manuell ausgeführt, die Richtigkeit des Designs wird durch Programme geprüft. Die Forderungen B und C würden gut, D und E schlecht erfüllt. Die Designsicherheit kann nur durch konsequentes «top-down Design» wirklich garantiert werden.

4. Hierarchie

Die Komplexität der heutigen Schaltungen bewirkt, dass entweder lange Entwicklungszeiten in Kauf genommen werden müssen oder aber die Fläche schlecht genutzt wird. Nur durch Reduktion der Komplexität kann eine signifikante Verbesserung erreicht werden.

Die Struktur eines Systems bestimmt dessen Komplexität. Zur Illustration mag der Vergleich zwischen einer hoch strukturierten Schaltung (z.B. 64 k RAM) und einer unstrukturierten Schaltung (Mikroprozessor der ersten Generation) dienen. Obwohl die Speicherschaltung etwa zehnmal mehr Transistoren enthält, ist die Entwicklung verhältnismässig einfach, da eine einzige Entscheidung 65 536 Transistoren beeinflusst, statt nur einige we-

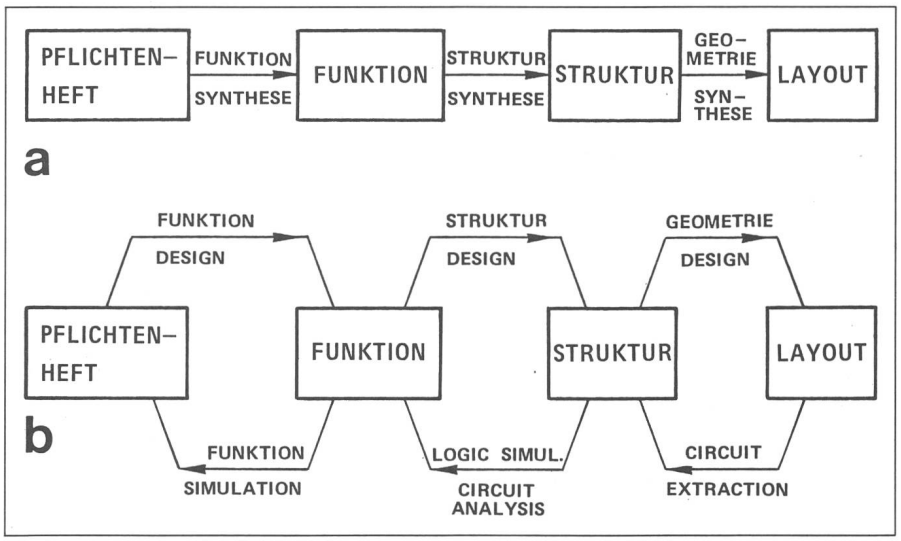

Fig.1 Vollautomatisches (a) und manuelles (b) CAD

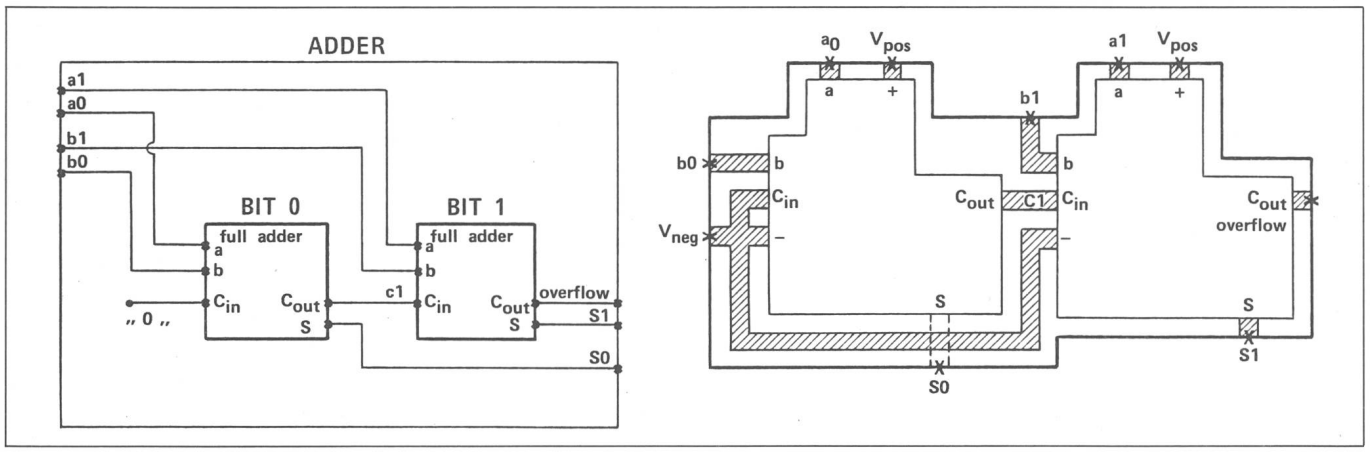

Fig. 2 Layout und Struktur eines hierarchischen Blocks
Adder: 2 Bit-Adder I (A0, A1, B0, B1) 0 (S0, S1, Overflow)

Begin
Bit 0: Full adder I (A0, B0, 0) 0 (S0, C1)
Bit 1: Full adder I (A1, B1, C1) 0 (S1, Overflow)
End

nige. Regelmässige Strukturen vereinfachen also ein System, in vielen Fällen genügen sie aber den Geschwindigkeitsanforderungen nicht.

Eine andere Möglichkeit, die Komplexität zu kontrollieren, ist die Abstraktion, d.h. das Ersetzen eines Elementes durch ein einfacheres, welches nur noch die Wechselwirkung mit der Aussenwelt definiert, aber keinen Inhalt mehr besitzt (Black Box). Abstraktion kann die Datenmenge, welche nötig ist, um ein Element zu beschreiben, um Grössenordnungen reduzieren. Beinahe alle gängigen CAD-Systeme verwenden Abstraktion auf der einen oder anderen Stufe («Stickdiagrams», Gates usw.). Erst durch konsequent wiederholte (hierarchische) Abstraktion ist es aber möglich, beliebig komplexe Systeme in den Griff zu bekommen.

Ein Block im hierarchisch organisierten Design besteht aus einer Anzahl untergeordneter Blöcke, welche durch Leitungen miteinander verbunden sind, und aus Anschlusspunkten für Signale zur Aussenwelt. Die untergeordneten Blöcke sind selbst wieder in derselben Art aufgebaut. Jeder Block im Layout muss definierte Grenzen besitzen, welche nicht durch Leitungen oder andere Blockgrenzen überkreuzt werden dürfen. Auf dieser Grenze müssen auch die Anschlusspunkte für Verbindungen nach der Aussenwelt liegen.

Die Blockgrenzen und die Lage der Anschlüsse werden ihrerseits bestimmt durch das interne Layout, d.h. durch die Anordnung der untergeordneten Blöcke und deren Verbindungen.

Zu jedem Block gehört demnach eine Beschreibung seiner Struktur und eine Beschreibung seiner Geometrie (Fig. 2). Die strukturelle Beschreibung muss enthalten:

- den Namen des Blocks,
- die Namen der Signale zur Aussenwelt,
- die Namen der untergeordneten Blöcke und
- die Beziehung der untergeordneten Blökke (Netzwerk).

Die geometrische Beschreibung muss enthalten:

- die Koordination der Blockgrenzen,
- die Koordination der Anschlusspunkte,
- die geometrische Anordnung der untergeordneten Blöcke und
- die geometrische Anordnung der internen Verbindungen.

Es ist aber durchaus denkbar, dass weitere Information gespeichert werden kann, wie z.B. Leistungsaufnahme, Kommentare, Designstatus usw. In einem hierarchischen System wird immer ein Verlust an Dichte zu erwarten sein, da die Blockgrenzen einen Teil der verfügbaren Fläche beanspruchen. Je kleiner ein Block ist, um so grösser wird auch die «verschwendete» Fläche. Anderseits wird durch die Reduktion der Komplexität das Design stark vereinfacht und dadurch beschleunigt.

5. Interaktivität

Im direkten Dialog mit dem Computer können die Resultate jeder einzelnen Operation unmittelbar überprüft und fehlerhafte Eingaben sofort korrigiert werden. Es ist allerdings wesentlich, dass die Datenmengen, welche verarbeitet werden müssen, möglichst klein sind, andernfalls werden die Antwortzeiten zu lang. Das hierarchische CAD-System erfüllt diese Bedingungen optimal, es ist denkbar, neben der Syntax auch die elektrische und geometrische Richtigkeit des Designs in Echtzeit zu überprüfen.

Für die Verbindungen zwischen einzelnen Blöcken werden im allgemeinen nur zwei, höchstens drei Masken verwendet, dadurch verringert sich die Anzahl der relevanten Layoutregeln drastisch. DRC kann unter Umständen sogar trivial sein!

Für eine Technologie sei die minimal zugelassene Linienbreite $2d$, der minimale Abstand zweier Linien auch $2d$. Durch die Wahl eines Gitters mit der Maschenweite $4d$ kann jede Verletzung von Layoutregeln automatisch ausgeschlossen werden (Fig. 3). Die Verifikation, ob eine Verbindung zwischen zwei Blöcken elektrisch richtig ist, kann durch einen Vergleich der den entsprechenden Anschlusspunkt zugeordneten Signalnamen gemacht werden. Interaktivität kann aber auch dazu benutzt werden, dem Ingenieur seine fast vergessene Umgebung wieder näherzubringen. Es ist nicht einzusehen, weshalb die Skizze eines Schaltschemas in Text umgewandelt werden

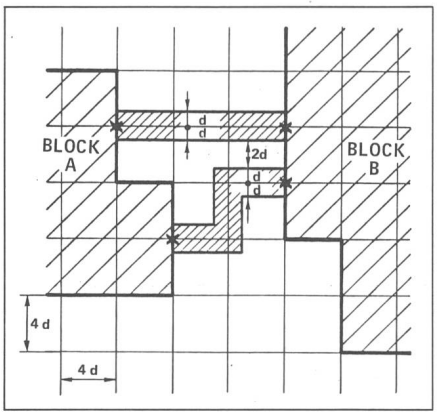

Fig. 3 Einfache Methode zur Einhaltung der Layout-Regeln

muss, um für einen Computer akzeptabel zu sein; mit einem geeigneten grafischen Editor könnte dasselbe Schema am Bildschirm gezeichnet und mit einem Simulationsprogramm direkt evaluiert werden. Die Ausgabe der Resultate könnte dann auch in grafischer Darstellung geschehen, anstatt in Form von Tabellen. Diese Art von Arbeiten, welche man als «graphic breadboarding» bezeichnen könnte, hat in einem hierarchischen System seine besondere Berechtigung, da die Komplexität der einzelnen Blöcke klein gehalten werden kann und damit auch die Rechenzeit der Simulationsprogramme.

Über kurz oder lang wird automatisches Design (DA: Design Automation) eingeführt werden müssen. Auch hier könnte sich hierarchische CAD als optimale Umgebung erweisen.

Wenn ein Block nur aus wenigen, beispielsweise maximal 50 untergeordneten Blöcken aufgebaut ist, wird automatisches Layout realistisch. Interaktive DA-Programme könnten zudem die Geschwindigkeit des Computers kombinieren mit der Fähigkeit des Menschen, Zusammenhänge zu erkennen.

6. Implementation eines hierarchischen VLSI-Layout-Systems

Falls die Steigerung der Integration von VLSI im gleichen Tempo weitergeht wie bisher, dann sind 1990 Schaltungen mit 10 Mio Komponenten zu erwarten. Ein VLSI-System muss diese Datenmenge effizient behandeln können. Die Forderung nach Interaktivität bedeutet, dass farbige Bildschirme mit hoher Auflösung und hoher Bandbreite (Geschwindigkeit) eingesetzt werden müssen. Sobald CAD alle Designstufen umfasst, muss pro Entwickler ein Bildschirm zur Verfügung stehen; dies lässt sich durch dezentralisierte Intelligenz (Workstations) einfacher und kostengünstiger lösen als durch «Monstercomputer». Ein zentraler Computer wird gebraucht für die Speicherverwaltung von Bibliotheken, zur Kommunikation unter den Designern und als Rechner für Batchverarbeitung (z.B. Postprocessor für Maskenherstellung usw.).

Als Eingabe ins Layoutsystem muss eine strukturelle elektrische Beschreibung zur Verfügung stehen. Diese Beschreibung muss schon hierarchisch organisiert sein. Idealerweise wären die verschiedenen hierarchischen Ebenen des elektrischen Netzwerks (Transistor-, Gate-, Register-Transfer- und funktionelle Beschreibung) in einer einzigen Datenbasis vorhanden, heute existiert aber noch kein Simulator, der eine so grosse Spannweite von Beschreibungen akzeptiert [4].

Im Moment können «Mixed Mode»-Simulatoren verwendet werden, wie SPLICE oder DIANA [7]. Diese vereinigen Schaltungsanalyse und logische Simulation in einem Programm, zudem können «Macros» gebildet werden, um die Hierarchie wiederzugeben. Als erstes muss nun die hierarchische Struktur aufgebaut werden (Baumstruktur); dann ist es möglich, die elektrische Beschreibung in ein symbolisches Layout umzusetzen. In dieser Darstellung werden gewisse Vereinfachungen gemacht; Verbindungen werden nur als Linien gezeichnet und Kontakte z.B. als Kreuze dargestellt.

Die Eingabe der Daten erfolgt in einer adäquaten Sprache (SLDL: Symbolic Layout Description Language) oder natürlich grafisch. Es scheint sinnvoll, die Designfreiheit zu beschränken und z.B. nur orthogonale Figuren zuzulassen, dadurch wird die Implementation von Echtzeit-DRC stark vereinfacht. Im symbolischen Editor besteht die Möglichkeit für verschiedene Darstellungen des Layouts; insbesondere wird es möglich sein, Information aus allen tieferen hierarchischen Ebenen einzubeziehen, d.h. das Layout mit mehr und mehr Details zu zeigen (logical zooming).

Die Abgrenzung der Blöcke darf keine zu grossen Flächenverluste verursachen. Dies kann erreicht werden, wenn individuelle Grenzen für verschiedene Masken zugelassen werden und diese aus mehreren unzusammenhängenden Gebieten bestehen dürfen.

Für die Beschreibung der untersten Hierarchiestufe, nämlich des wirklichen geometrischen Layouts, ist SLDL nicht geeignet, weil:

- keine symbolische Darstellung mehr möglich ist;
- geometrische Figuren kein elektrisches Äquivalent haben müssen;
- die Beschränkung auf orthogonale Figuren zu restriktiv ist.

Der geometrische Editor und seine Sprache (GLDL: Geometric Layout Description Language) sollen aus Gründen der Benutzerfreundlichkeit möglichst viel Ähnlichkeit mit SLDL haben, aber mehr Freiheiten zulassen (Polygone mit beliebigen Winkeln, Kreise, Test usw.).

Literatur

[1] *G. E. Moore:* Are we really ready for VLSI? Digest of the IEEE International Solid-State Circuits Conference (ISSCC) 1979, p. 54...55.

[2] *J. D. Meindl* a. o.: Circuit scaling limits for ultra large-scale integration. Digest of the IEEE International Solid-State Circuits Conference (ISSCC) 1981, p. 36...37.

[3] *D. P. Sieworek, D. E. Thomas* and *D. L. Scharfetter:* The use of LSI modules in computer structures: Trends and limitations. Computer 11(1978)7, p. 16...25.

[4] *S. Garcia* and *K. S. Siram:* A survey of IC CAD tools for design, layout and testing. VLSI Design 3(1982)5, p. 68...73.

[5] *D. A. Patterson* and *C. H. Sequin:* RISC I: A reduced instruction set VLSI computer. Eigth Annual International Symposium on Computer Architecture (Comparc) 1981, p. 443...457.

[6] *J. Werner:* The silicon compiler: Panacea, Whisfull Thinking, or Old Hat. VLSI Design 3(1982)5, p. 46...52.

[7] *H. de Man* a. o.: DIANA as a mixed-mode simulator for MOSLSI sampled-data circuits. IEEE International Symposium on Circuits and Systems (ISCAS) 1980, Vol. II, p. 435...438.