Hierarchical Design based on a Calculus of Nets

Bernd Becker, Günter Hotz, Reiner Kolla, Paul Molitor and Hans–Georg Osthof

Fachbereich 10, Universität des Saarlandes
D-6600 Saarbrücken, FRG

Abstract

We present an algebraic approach to hierarchical design of integrated circuits. This approach is based on a "calculus of nets" which includes topological as well as behavioural aspects of integrated circuits. We have developed a hierarchical design system called CADIC which is build around this calculus in much the same way as e.g. Algol is build around numerics. An example for the design of a family of fast adders will demonstrate the power of this calculus. Finally we will give a summary outline on the structure of procedures which automatically transform the design into lower design levels.

1. Introduction

The progress made in the field of microelectronics has been accompanied by a thorough study of design automation methods in order to create systems which permit the design and the synthesis of VLSI-systems within a reasonable amount of time. Hereby a design system should enable the designer to specify the structure of a design in a simple and natural way. On the other hand, the specification of a design should include enough information about its geometrical and behavioural structure in order to make an automatic synthesis feasible without a dramatic loose of performance. Furthermore the specification should have a precise mathematical meaning. This creates the possibility to reason about properties of the design and even to prove it correct rather than simulating it for a certain number of inputs. But this requires, that each step of an automatic synthesis maintains all properties fullfilled by the specification (correctness by construction).

There are many design systems using powerful specification methods or languages based on a more or less abstract design level [3, 10, 2], but these systems are of mainly pragmatic nature, and often they focus only on layout or only on logical structures. On the other hand, algebraic methods are used to describe digital systems [1, 6, 17], but they mainly deal with precise behavioural models whereas geometrical properties are almost completely suppressed.

In this paper we give a short summary of a calculus which contains logical as well as topological properties of integrated circuits. The calculus is based on simple operations and also provides a natural approach to a hierarchical style of design. It is the basis of the design system CADIC [5]. We demonstrate the power of this calculus by an example, namely the design of a family of fast adders, and give a summary outline on experiences with hierarchical algorithms for the synthesis as far they have been implemented todate.

2. A Calculus of Nets

Boolean Algebra is the classical calculus for dealing with logical circuits. This calculus was sufficient as long as the cost of wires were negligible compared with the cost of gates. Since this is no longer the case in integrated circuit design, we need a calculus representing the logical function of the design together with some information about the geometrical arrangement of its components. A first extension of boolean algebra in this direction was given in 1965 with the introduction of the x-category by G. Hotz [4]. In the following we present a generalization of these concepts which are needed for the purpose of VLSI-design.

We consider circuits layed out into a rectangle R. In order to suppress geometrical and physical details of

manufacturing processes and thus to become independent of technology, we forget the width and the layer of wires. In doing so, wires become simple lines which may branch and cross each other. Furthermore we suppose that the circuit is constructed by cells which compute digital values. Assuming that these cells are physically correctly designed, we suppress their internal structure and size and maintain only the order of external connectors on their boundaries. If we consider crossings and branchings of wires also as cells which perform crossings and branchings of signals, this abstraction results in an arrangement of cells in the plane whose interconnections consist of crossing-free non-overlapping lines. (see figure 2.1)

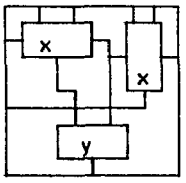

figure 2.1

Now we define for each cell a northern, southern, western and eastern side, on which connectors are placed. (No connector should belong to a corner, i.e. it should belong to exactly one side.) We denote for a cell a the number of the connectors onto the northern (, southern, western, eastern) side by $N(a)$ $(, S(a), W(a), E(a))$. Let the same be defined for the rectangle R. To suppress precise geometrical relations of this abstract layout, we consider two such layouts to be equivalent iff they can be transformed into each other by a sequence of deformations as deformations of wires, translations and stretching of cells, and order preserving translations of connectors along the side of a rectangle. It is important that the deformations do not produce overlappings or crossings, i.e. they have to maintain the planar topological structure of the layout. Figure 2.2 shows two such abstract layouts which we consider to be equivalent. We call the set of all layouts, which can be transformed into each other by a sequence of such deformations, a **logic topological net**. The elements of this set are **topographical representatives** of a logic topological net.

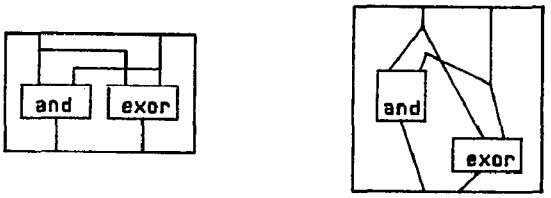

figure 2.2

A logic topological net gives a precise characterization of an integrated circuit, which is sufficiently abstract to suppress geometrical and physical details, and which is sufficiently concrete to control the arrangement of cells and the global routing of wires. A detailed and precise theoretical background is given in [12]. If we relate to each cell its behaviour by a boolean function or a more general model, we also get a precise mathematical characterization of the behaviour of logic topological nets [12], [8].

We will now define operations between logic topological nets. On the geometrical design level abutment is the simplest operation between large objects. Its meaning is obvious and errors are immediatly discovered since operands do not match. From the functional point of view, abutment transforms into a composition of functions, which also has a clear meaning and makes the behavioural part transparent to the designer [12], [8]. The objects of our calculus are logic topological nets and the operations are compositions of nets which we define by abutment of topographical representatives. There are two kinds of compositions, namely the horizontal composition $\ominus$ ("left from") and the vertical composition $\oslash$ ("above"). The composition $N_1 \oslash N_2$ is defined for two nets N_1, N_2 iff there are two topographical representatives of N_1, N_2 so that the southern side of the first representative matches the northern side of the otherone. This operation can be carried out iff $S(N_1) = N(N_2)$, since there do obviously exist two appropriate representatives iff the number of connectors on the participating sides are equal. The result is the net which is represented by abutment of these two representatives. (see fig. 2.3)

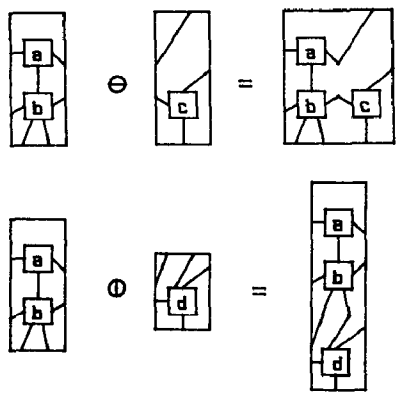

figure 2.3

For the number of connectors on the other sides relations like $N(N_1 \oslash N_2) = N(N_1)$, $W(N_1 \oslash N_2) = W(N_1)+W(N_2),\ldots$ hold. There are also neutral elements for these operations denoted by $1_n^{\oslash}, 1_n^{\ominus}$ which consist of n vertical or horizontal wires. By these operations we get a calculus of nets whose algebraic structure has been studied in [12, 14]. A further important operation is the refinement of nets, i.e. the parallel replacement of each occurrence of a cell by the same net as illustrated in figure 2.4. This operation can be considered as a homomorphic mapping in our calculus and plays an important role in the specification of nets and in the definition of the hierarchical structure of a design.

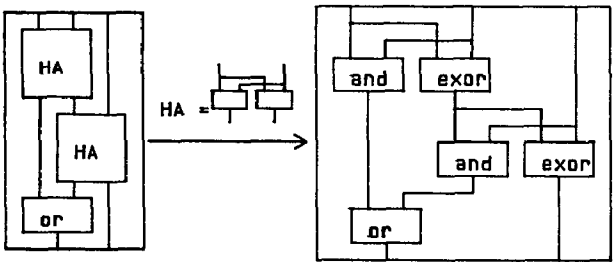

figure 2.4

3. An Example

We will design a fast binary adder using the conditional sum principle [18]. Let

$$a = a_{n-1},\ldots,a_0;\ b = b_{n-1},\ldots,b_0$$

be two n-bit numbers, and let $a^{(1)} := a_{n-1},\ldots,a_{\frac{n}{2}}$ be the number represented by the most significant $\frac{n}{2}$ bits, and $a^{(0)} := a_{\frac{n}{2}-1},\ldots,a_0$ be the number represented by the least significant $\frac{n}{2}$ bits. $b^{(1)}, b^{(0)}$ is defined analogously. Suppose that we have already designed an $\frac{n}{2}$-bit adder computing the sum and the sum plus 1 of its inputs simultanously. Then $a+b, a+b+1$ can be computed by two $\frac{n}{2}$-bit adders and a multiplexer stage as follows:

- Compute $a^{(0)} + b^{(0)}, a^{(0)} + b^{(0)} + 1$ and $a^{(1)} + b^{(1)}, a^{(1)} + b^{(1)} + 1$ and by two $\frac{n}{2}$-bit adders in parallel. After this, the least significant $\frac{n}{2}$ bits of $a+b$ and $a+b+1$ are already computed.
- In order to complete the addition, select the most significant $\frac{n}{2}+1$ bits of $a+b, a+b+1$ by the relation

$$(a+b)_i = \begin{cases} (a^{(1)} + b^{(1)})_{i-\frac{n}{2}} & \text{if } (a^{(0)} + b^{(0)})_{\frac{n}{2}} = 0 \\ (a^{(1)} + b^{(1)} + 1)_{i-\frac{n}{2}} & \text{if } (a^{(0)} + b^{(0)})_{\frac{n}{2}} = 1 \end{cases}$$

for $i = n,\ldots,\frac{n}{2}$

(an analogous relation holds for $(a+b+1)_i$)

This results in a simple recursive structure as illustrated in figure 3.1.

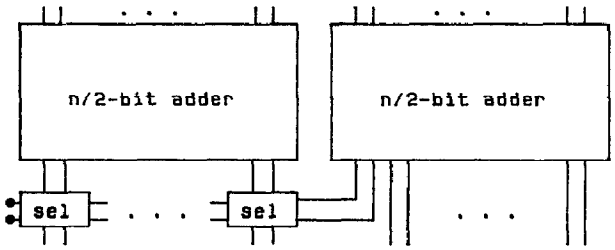

figure 3.1

The CADIC system provides an interpreter for a simple language which consists of recursive equations defining refinements of logic topological nets. Let $adder[k]$ represent a net performing an $n = 2^k$-bit addition by using the above scheme. In our calculus we can specify its structure as follows

$$adder[k] = (adder[k-1] \ominus adder[k-1]) \oslash ((\bullet\!\!- \oslash \bullet\!\!-) \ominus \ominus^{2^{k-1}+1} sel \ominus ((\lrcorner \ominus 1_1^{\oslash}) \oslash \lrcorner) \ominus 1_{2^k}^{\oslash})$$

where we use the composition operations of our calculus. In this formula, we denote cells which define wiring structures by symbols reflecting their geometrical appearence as + for a crossing or $\lrcorner$ for a knee. The part $(\bullet\!\!- \oslash \bullet\!\!-)$ represents two unconnected ends of the preceding multiplexer circuit, and the part $((\lrcorner \ominus 1_1^{\oslash}) \oslash \lrcorner)$ specifies a "double knee". We also use iterative operators. $\ominus^n A$ denotes $\underbrace{A \ominus \cdots \ominus A}_{n-\text{times}}$.

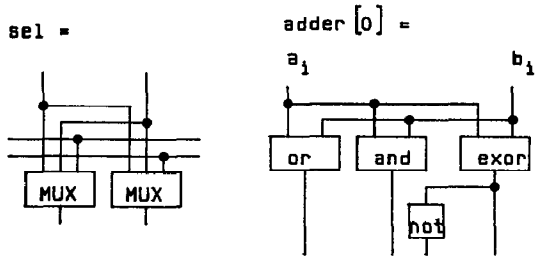

figure 3.2

Refinements of a 1-bit adder and the multiplexer circuit *sel* are given in figure 3.2. (To be brief, we omit the arithmetical expressions representing these circuits, because these are very technical. Refinements can also be specified graphically.)

Thus, the specification of a fast adder consists of one recursive and two simple equations. Such a refinement also defines a very compact hierarchical representation. Since e.g. a 32-bit adder consists of 2 16-bit adders, 4 8-bit adders etc., we can represent it by storing the representation of each part only once. This is shown for a simpler structure in figure 3.3, where we compress a syntax tree which represents an iterative expansion of refinements into a graph in which equal subtrees occur only once.

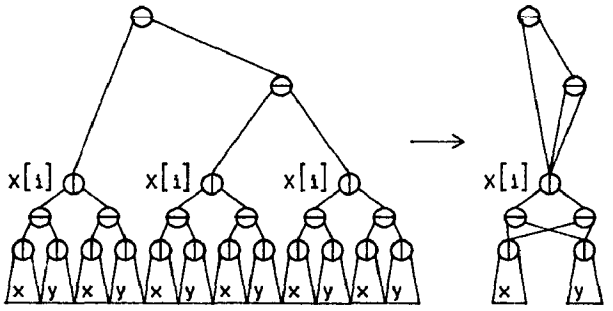

figure 3.3

The use of hierarchy does not only save memory by compact hierarchical representations but also saves CPU-time for synthesis procedures if they work hierarchically. The CADIC system provides such procedures to automatically produce layouts which become more and more concrete starting from the logic topological level of abstraction. These are for example the generating of the wiring for the power supply, layer assignment, transformation into grid structures. All these steps raise optimization problems which are approximated by fast hierarchical algorithms. The layer assignment procedure tries to minimize the number of contacts and is implemented by a probabilistic algorithm [11]. The general structure and complexity of this problem has been investigated in [16], [13]. Figure 3.4 shows the result of the layer assignment and generation of power supply (one layer is represented by grey lines, the layer of the power supply is given by bold lines). The transformation into grid structures tries to minimize the area. The procedure approximates this optimization by locally solving a placement problem for channel routing [7], which is based on a placement algorithm given in [9]. This procedure has produced the representative of figure 3.4.

If we consider hierarchical optimization algorithms there is a trade off between the "degree of hierarchy" (the number and structure of refinement steps) and the quality of the result which can be produced by a hierarchical algorithm preserving the hierarchical structure. This has been studied for hierarchical layer assignment for some examples in [15]. These examples have shown, that minor modifications of the hierarchical structure may dramatically change the number of contacts which are required by the hierarchical structure, but mostly there also exists a compact hierarchical representation which requires only a few more contacts than a non hierarchical representation of the same circuit. Therefore, the CADIC system provides the possibility for the user to manipulate the hierarchical representation explicitly in order to get better results by some optimization procedures. Other synthesis algorithms as e.g. the generation of power supply wires choose an appropriate hierarchical structure by themselves.

References

[1] L.Cardelli:
"An Algebraic Approach to Hardware Description and Verification"
PH.D. Thesis, Edinburgh 1982.

[2] E.Clarke, Y.Feng:
"ESCHER - A geometrical Layout System for Recursively Defined Circuits"
CMU-CS-85-150, Department of Computer Science, Carnegie-Mellon University, Pitsburgh 1985.

[3] E.Hörbst, M.Nett, H.Schwärtzel:
"VENUS - Entwurf von VLSI-Schaltungen"
Springer-Verlag Berlin Heidelberg New-York Tokyo 1986.

[4] G.Hotz:
"Eine Algebraisierung des Syntheseproblems

für Schaltkreise"
EIK 1, 1965, pp.185-205,209-231.

[5] G.Hotz, B.Becker, R.Kolla, P.Molitor:
"Ein logisch-topologischer Kalkül zur Konstruktion von integrierten Schaltkreisen"
Informatik: Forschung und Entwicklung,
Heft 1 und 2, Springer Verlag 1986.

[6] C.D.Kloos:
"Towards a Formalization of Digital Circuit Design"
TUM-I8604, February 1986, Technische Universität München.

[7] R.Kolla:
"Spezifikation und Expansion logisch-topologischer Netze"
Dissertation, Saarbrücken 1986/87.

[8] R.Kolla
"Verification of logic-topological Nets"
will appear as technical report, SFB124, Saarbrücken 1987.

[9] C.E.Leiserson, R.Y.Pinter:
"Optimal placement for river routing"
SAM J. Comput. 12, pp.447-462, 1983.

[10] T.Lengauer, K.Mehlhorn:
"The HILL-System: A Design Environment for the Hierarchical Specification, Compaction and Simulation of Integrated Circuit Layouts"
MIT VLSI Conference 1984, pp.139-149, Artech House.

[11] P.Molitor:
"Layer Assignment by Simulated Annealing"
Microprocessing and Microprogramming 16 (1985), pp.345-350, North-Holland.

[12] P.Molitor:
"Über die Bikategorie der logisch- topologischen Netze und ihre Semantik"
Dissertation, Saarbrücken 1986.

[13] P.Molitor:
"On the contact minimization problem"
Proceedings of the STACS 87, Passau.

[14] P.Molitor:
"Free Net Algebras in VLSI-Theory""
in preparation.

[15] P.Molitor, R.Kolla:
"A note on hierarchical layer-assignment"
TR 8/86, SFB124, Saarbrücken 1986.

[16] R.Y.Pinter:
"Optimal layer assignment for interconnect"
ICCC 1982, pp.398-401.

[17] M.Sheeran:
"μFP - An algebraic VLSI design language"
PH.D. Thesis, University of Oxford, England 1984.

[18] J. Sklansky
"Conditional-sum addition logic"
IRE-EC 9, 226-231 (1960)

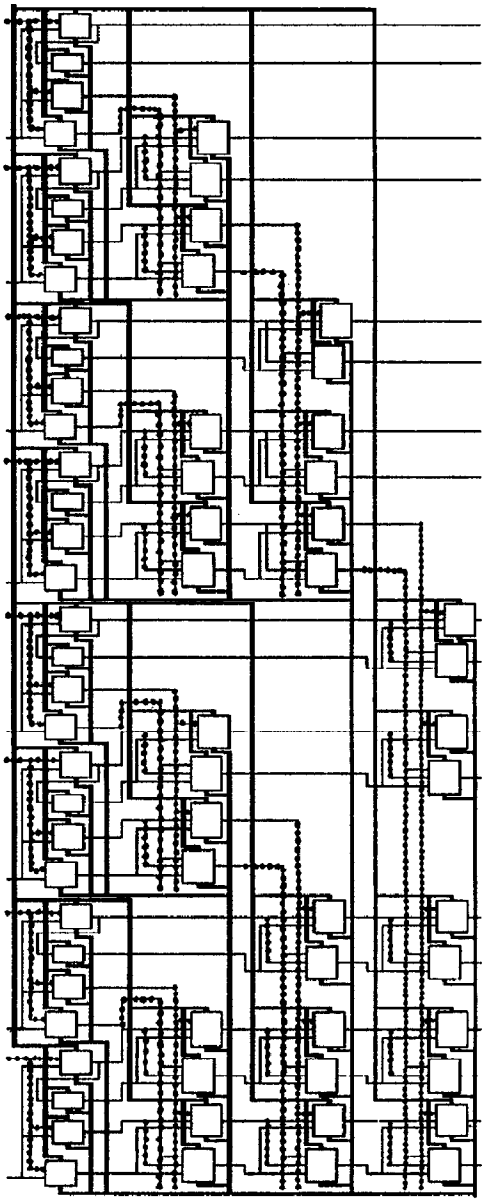

figure 3.4