The HILL System: A Design Environment for the Hierarchical Specification, Compaction, and Simulation of Integrated Circuit Layouts

A83/04

Thomas Lengauer
Kurt Mehlhorn +)

Fachbereich 10
Universität des Saarlandes
6600 Saarbrücken
West Germany

+) Research supported by: Deutsche Forschungsgemeinschaft
Grant: SFB 124, Teilprojekt B2

1. Introduction

The HILL (Hierarchical Layout Language) system is currently under development at the University of the Saarland. The development of HILL is one of several research projects that are funded by Deutsche Forschungsgemeinschaft, Grant SFB 124, and are under way at the universities in Kaiserslautern and Saabrücken, West Germany. The research projects include the development of CAD systems for VLSI circuit development, such as HILL and CADIC as well as work on parallel machine architectures and their software.

HILL aims at supporting symbolic layout design on the stick diagram level. For this purpise the HILL system provides a hierarchical layout specification language in conjunction with a graphis editor, a switch level MOS simulator, and a compacter.

The simulator is based on a simple MOS model that includes no timing. Transistors are modeled as switches that can be in one of three states: 0,1, and X modelling off, on, and undefined. In [MNN82] three things are proved:

1. The correctness of the simulation w.r.t. the model
2. The relationship between the model and the reality of RC-networks
3. The efficiency of the simulation

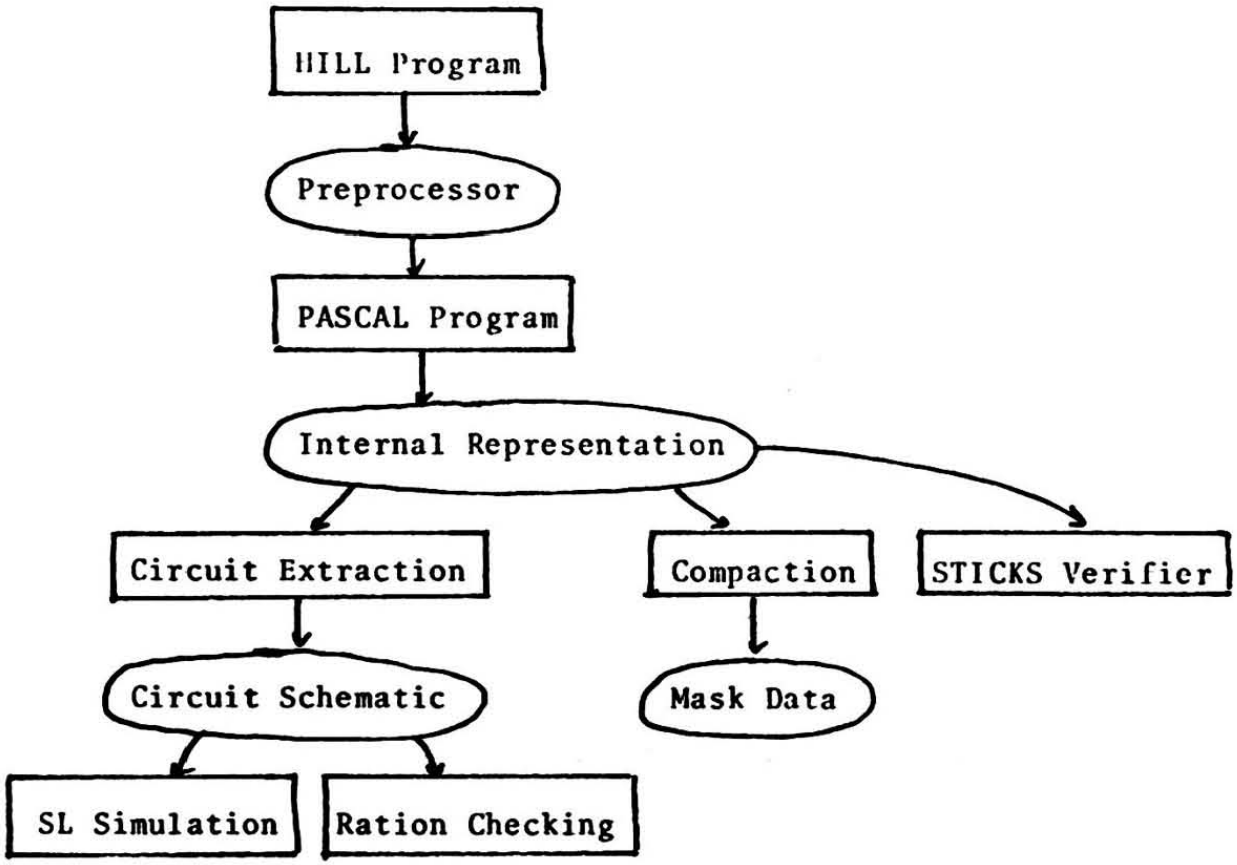
The simulator runs in time O(n) on a large class of networks, where n is the number of transistors in the network that change state. Experimental evidence shows that 0.2 msec are spent per transistors on a SIEMENS 7760.

Since the simulator simulates the electrical network underlying symbolic layouts it has no access to delay information. Currently simulation algorithms are being developed which ensure the functioning of a circuit independently of the delays associated with gates and wires.

The compacter is based on the graph-theoretic approach for gridless compaction pursued by [CABBAGE,FLOSS,STICKS]. The algorithms it uses are described in [L82a,L83]. It allows previously unachieved topological flexibility. Specifically it can switch the positions of adjacent circuit structures that lie on the same mask layer, if no design rules are violated during this process, and if doing so reduces the area of the layout. The compaction algorithm is proved to run in O(n log n) time, and is expected to perform very well in practice.

A less efficient and more complicated compacter has been implemented as part of [HILL82], a predecessor of the HILL system. Nevertheless, even in HILL82 compaction is quite fast. For example, the adder using the carry chain below consists of (262,564) transistors for (8 bit, 16 bit) numbers, and is compacted in about (28.8, 61.4) seconds on a SIEMENS 7760.

This paper cannot go into the details of the simulator and the compacter. Here we intend to give an overview of the specification language used in HILL. It is described in detail in [HILL83].The implementation of HILL83 is under way and will be completed by the end of 1983. A predecessor [HILL82] is running and is being used successfully by us and students. HILL82 has a less flexible cell concept, a less efficient (running time and quality of output) compacter and a less convenient user interface. (Compilation from HILL to PASCAL had to be done by hand.) The following diagram gives an overview of the HILL system.


HILL Program
Preprocessor
PASCAL Program
Internal Representation
Circuit Extraction
Compaction
STICKS Verifier
Circuit Schematic
Mask Data
SL Simulation
Ration Checking

2. Overview of the HILL Specification Language

HILL is a tool for single chip development. The main focus of HILL is layout generation and verification. HILL aims at supporting the designer who has a comprehensive global image of his circuit. HILL provides a convenient way of describing a layout symbolically either by a HILL program or during an interactive session at the graphics terminal. Even though the layout is specified in a symbolic manner the designer has many means of exerting direct influence on the quality of the resulting mask data.

HILL is a system which combines convenient circuit description with efficiency of the implementation. We aim for efficiency in three respects: Human design time, chip area and delay, and computational resources.

Human Resources: In HILL, integrated circuits are described at the level of stick diagrams enhanced by extensive means for structuring a design hierarchically. We have chosen the level of stick diagrams because on the one hand it still allows the designer to express his insights about the topology of the circuit and on the other hand it frees the designer from the tedious and error-prone task of specifying his circuit at the mask level. The stick diagram level has been used successfully in systems like [CABBAGE,STICKS,MULGA].

Even though it certainly is no good practice to specify a whole large scale circuit with one giant stick diagram, if enhanced by extensive means for hierarchical structuring, especially with a powerful cell concept, stick diagrams become a convenient symbolic representation of even large scale layouts. In the HILL report [HILL83] this thesis is exemplified by a number of examples. We will give only a small example below.

Most existing stick diagram systems are totally or almost totally graphics oriented. Graphics is indeed an ideal tool for describing small irregular parts of a layout. The graphics editor in HILL is similar to existing ones and is not described here. It is currently being implemented. Graphics has its limitations when it comes to building up hierarchies because it can support only limited mechanisms for structuring layouts. Essentially, graphics can only supports composition and primitive forms of iteration (duplication).

However, graphics cannot support regularity in its full generality. A (large) object is regular if it allows for a short description. Only a universal programming language can support regularity in its full sense. Therefore HILL is designed as a PASCAL extension which interacts gracefully with a graphics editor. In particular, HILL supports recursion, the full power of iteration, and parameter passing. Recursion and iteration are central to (software) algorithm design, and they have already proven to be powerful concepts in hardware design (e.g. [GV82,MC80 Chapter 8]). Recursion corresponds to trees, iteration to array-like structures, and programming in general allows to specify general regular networks.

A good example, although too long to be included here, is the multiplier described in [LM83]. This multiplier is based on the divide-and-conquer method using three multiplications of numbers of half length. The layout given there is regular, however, the regularity can certainly not be captured within a pure graphics system. Rather, powerful descriptive tools, such as recursion, iteration, and parameter passing are needed to capture the regularity.

Chip area and delay: Experience with existing systems [CABBAGE, STICKS,MULGA,HILL82] suggests that automatic compaction can yield small layouts which come close to hand-compacted layouts. The stick diagram level is close enough to the silicon to allow the designer to incorporate performance aspects into his speci-

fication, and the compacter supports chip performance with his knowledge of the fabrication process. Finally, the ease of circuit description and the feasibility of the algorithms used in the system allow the designer to try several approaches to his circuit and select the one he likes best.

Computational resources: Most existing systems have definite shortcomings in this respect. In most cases, only sketchy theoretical analyses of the running time are given, often algorithmic concepts enter the system only scantily. Computational experience with the systems suggests that the running time is highly non-linear. For example, it is reported that CABBAGE takes time $O(n^{1.2})$ to compact a circuit with n transistors and wires. In HILL82 we improved upon this and implemented an O(n log n) algorithm. A more elegant and flexible O(n log n) algorithm is described in [L82a,L83] and will be used in HILL83. However, even this algorithm will not do for large scale circuits, because of its $\Omega(\sqrt{n})$ space requirement. The solution to this problem is to compact hierarchically, see [L82b].

The main structuring device in HILL is that of a "cell". A cell is the specification of a subcircuit of the chip to be designed. This subcircuit will in general function as a module in the chip that communicates with its surroundings through relatively few connections (pins) and performs a specific subfunction of the chip function. It is rectangular in shape with the pins arranged on its boundary. It is very much reminiscent of a function in a sequential programming language like PASCAL. As procedure parameters form the (up to side effects exclusive) interface between the function and its call environment, so the pins of a cell facilitate the interface between the cell and the circuitry around its location of placement on the chip. The only way to contact to a cell is through one of its pins. Like procedures cells can be compiled separately and defined externally. For "instantiating" a cell only a description of its rectangular boundary, its so-called "template" has to be given. The template

contains no (electrical or topological) information about any of the inner workings of the cell. However, it contains both electrical and topological information about the cell boundary. Pins can be related to each other electrically in the template, and they have to be "placed" in order to specify the order in which they appear around the cell boundary.

In addition to specifying in which order the pins occur on each side of the cell, HILL also requires to specify the relative positions of pins on opposite sides of a cell. This is done by giving two partial orders, one for the x-direction and one for the y-direction. The system checks whether the symbolic layout given for a cell can be distorted such that it conforms to any linear extension of the partial orders. An example of this is given in the next section. The distortion mechanism implemented in HILL83 is one-dimensional for reasons of efficiency, and therefore one of the partial orders mentioned above has to be a linear order. Even with this limitation HILL cells are quite flexible and hence adapt easily to changes in the environment of their placement. On the other hand, the cell template captures enough information about the topology of a cell for the designer to keep track of the relative placement of his circuit components while stepping through the design hierarchy. We believe that this is a must if the designer wants to keep control of his layout while working on it hierarchically.

3. The HILL Specification of a Recursive Carry Look-Ahead Adder

We now give an example that highlights many of the features of the HILL specification language. We design the carry chain for an adder with complete carry look-ahead according to the construction given in [GV82] which is particulcarly well suited for NMOS technologies. The fabrication process for which we design the circuit is described in [MC80]. This circuit accepts the two numbers to be added not in binary notation $a=a_{n-1}\ldots a_0$, $b=b_{n-1}\ldots b_0$ but encoded as two strings of carry generate bits $g=g_{n-1}\ldots g_0$ and carry propagate bits $p=p_{n-1}\ldots p_0$. Here

$$g_i = a_i \wedge b_i \qquad p_i = a_i \oplus b_i \qquad i=0,\ldots,n-1$$

The bit pairs (g_i,p_i) $i=0,\ldots,n-1$ are then used to compute the carry bits c_i out of each bit position $i=0,\ldots,n-1$. Here an associative operation o on bit pairs is used that is defined as follows:

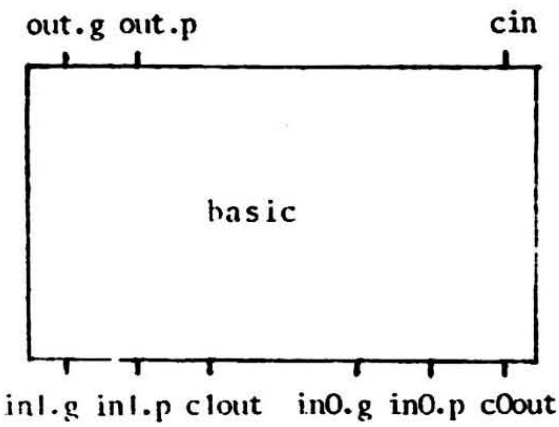
$$(g,p)\ o\ (g',p') = (g \vee (p \wedge g'), p \wedge p')$$

The carry chain computes the carry bit $c_i=a_i \oplus b_i \oplus c_{i-1}$ $i=0,\ldots,n$ $(c_{-1}=0)$ out of each bit position recursively using the o-operation. Specifically, it computes

$$(G_i,P_i) = (g_i,p_i)o(g_{i-1},p_{i-1})o\ldots o(g_0,p_0) \qquad i=0,\ldots,n-1$$

Note that $G_i=c_i$.

The circuit for the carry chain is defined recursively. Its bottom level element is the following basic cell.

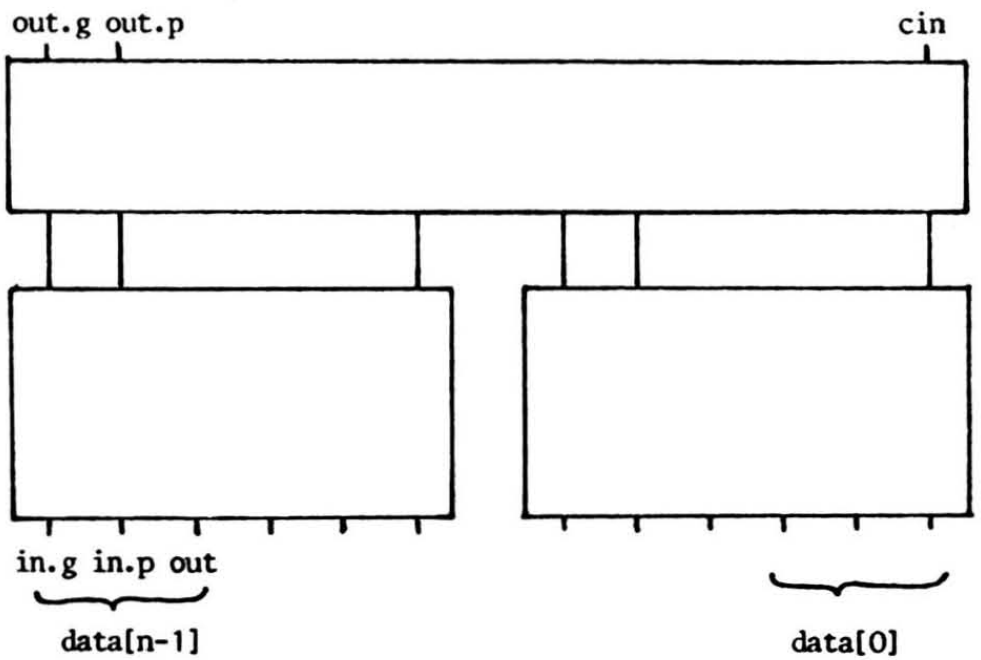


Note that in0, in1, and out denote pairs of pins. The cell performs the following function.

$$out = in1 \circ in0$$
$$c0out = cin$$
$$c1out = in0.g \vee (cin \wedge in0.p)$$

Thus, if we set cin=0 the basic cell computes the carry bits of the sum of two 2-bit numbers and provides them at the pins c0out and c1out.

For $n=2^i$, $i>1$, the carry chain is built up out of two carry chains for n/2-bit numbers according to the following diagram.

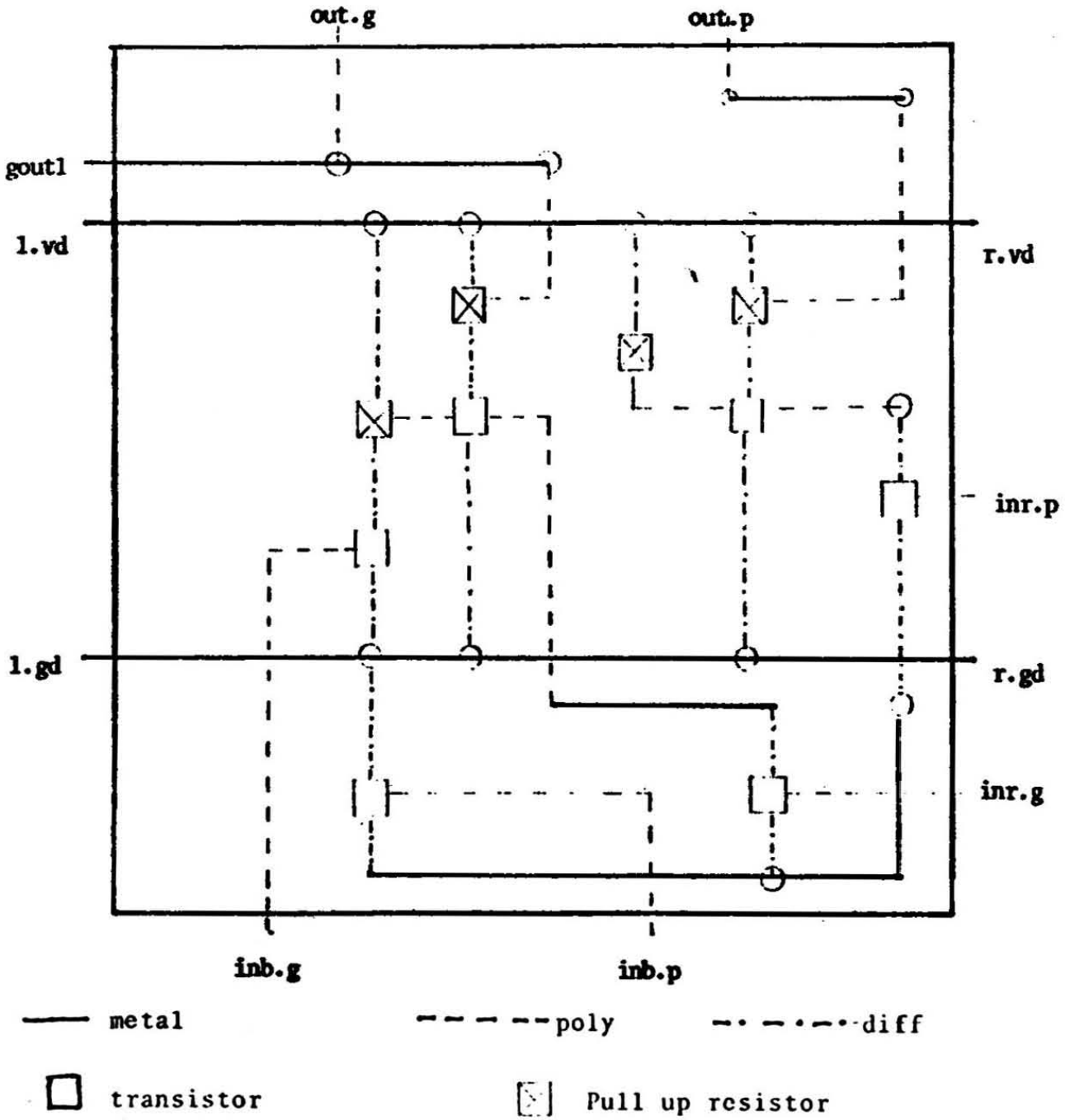


It is shown in [GV82] that this construction indeed computes the carry bit c_i and provides it at the pin data[i].out.

We will now describe these cells in HILL. We need three types of cells. The first cell has the name op and implements the o-operation. Its definition in HILL looks as follows:

```
aggregate   cbits = record g,p: poly end;
            ps = record vd: metal sig=vdd;
                        gd: metal sig=gnd
                 end;

cell op (z: int);
temp pins out, inb, inr: cbits;
          gout: metal;
          l,r: ps;
     order implicit y
     sides top out;
           bottom in;
           right r.vd at 2, inr.p, r.gd, inr.g;
           left goutl, l.vd, l.gd at 4
pmet
external;
```

The above text only describes the rectangular boundary of the cell. Its interior, i.e., its layout is input with the interactive graphics editor. This results in the following picture.



It can be easily checked that this cell realizes the following function:

$$goutl = out.g = ing.g \vee (inb.p \wedge inr.g)$$
$$out.p = inb.p \quad inr.p$$

The cell has an integer parameter z. This parameter determines the strength of the transistors in the cell, which has to be different in different levels of the recursive hierarchy of the chip specification (see [GV82]).

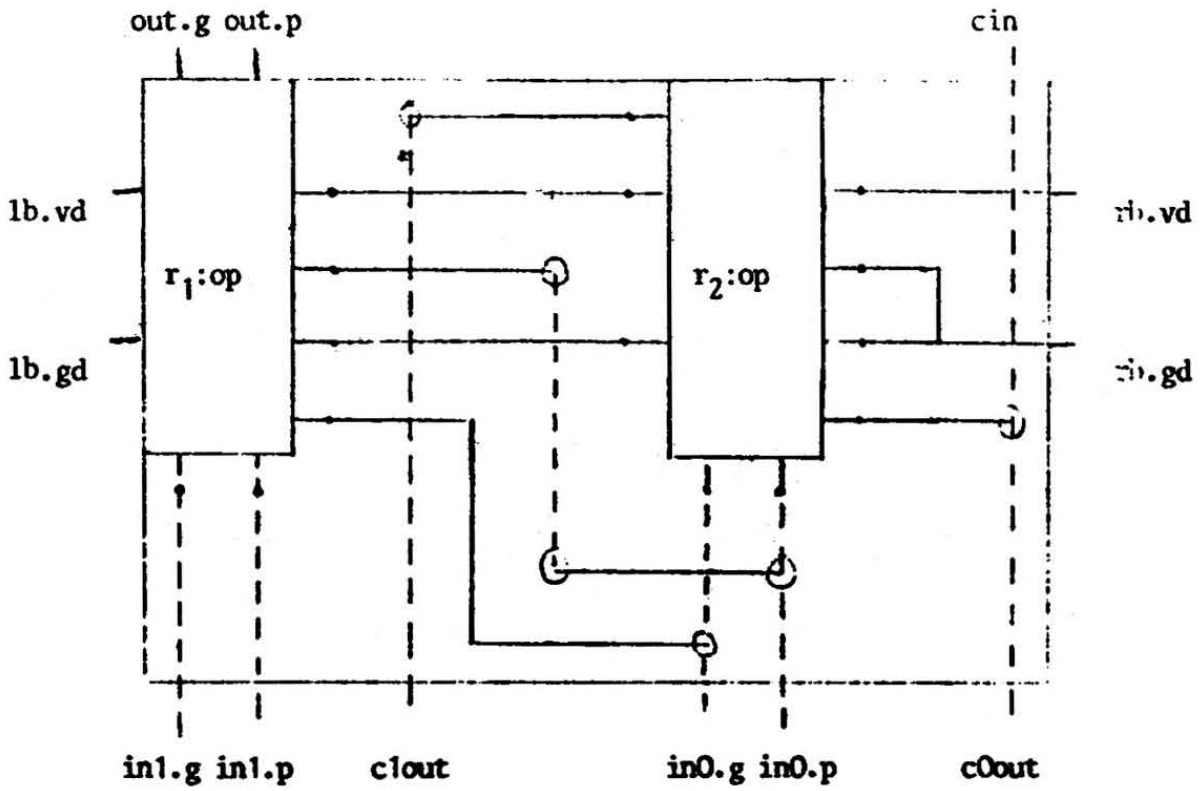
The specification of cell op is preceded by an aggregate-statement that forms groups of pins. Here corresponding carry generate and carry propagate lines are paired up, and a power supply bus is defined.

We now describe the specification of cell op. The header containing the numerical parameter is followed by a so-called template enclosed in the parentheses temp and pmet. The template gives information about the rectangular boundary of the cell op. It is followed by a definition of the layout of the interior of the cell. For graphically defined cells the symbol external is substituted for this part of the specification.

The template lists all pins of the cell in the pins section. Each pin has a sort that identifies the layers it exists upon and can be accompanied by some specification of the electrical signal it carries, e.g., gnd or vdd. Each pin also has a location on the boundary of the cell. In the sides section each pin is assigned a side of the template and the ordering of the pins on each side is determined by enumeration. Pins on opposite sides can also be related to each other w.r.t their positions. This can happen in two ways. In the above example the pins on the vertical sides (the y-pins) are paired up implicitly by assigning coordinate values to them. These values are assigned according to the sequence of enumeration except for explicit changes (e.g. r.vd is assigned the value 2). Thus pins 1.vd and r.vd as well as pins 1.gd and r.gd are paired, i.e., they receive the same y-coordinate in each symbolic layout of the cell. The locations of the x-pins of the cell op are not constrained w.r.t. each other. This reflects the fact that out.g and out.p are free to slide along the top side to the position required by the specific surroundings in which the cell is placed. Only limited use is made of this freedom in our example. When using cell op we could have always aligned out.p with inb.p and out.g with inb.g. However the freedom is carried out to the boundary of the cell containing the carry chain, where it may be used in a placement of the whole carry chain circuit higher up in a chip specification hierarchy.

This flexibility is a critical part of the cell concept underlying HILL. It allows for detailed information about the topology of the cell layout as we step through the chip specification hierarchy.

We use cell op to define cell basic. Cell basic can also (and should) be defined graphically, but we will specify it with a HILL program to exemplify the explicit layout mode provided in HILL.




```
cell basic (z:int);
temp pins out, in0, in1: cbits;
          cin, c0out, c1out: poly
          lb, rb: ps;
```

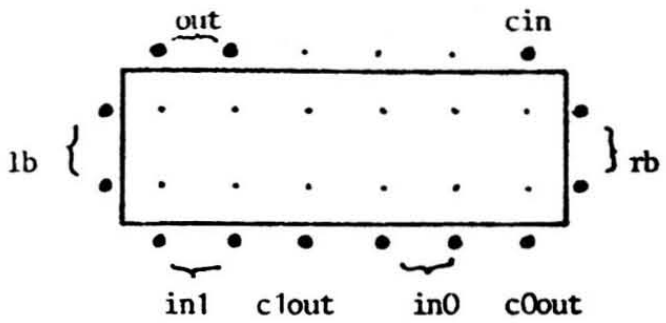
```
      order implicit y
      sides top out, cin;
            bottom in1, c1out, in0, c0out;
            left lb;
            right rb;
      constraints begin out.p leftof c1out;
                        cin above c0out
                  end
pmet

layout

extend order to top out, cin after 3;
                  bottom in1, c1out, in0, c0out;

components r1,r2: op(z) extend order to top out; bottom in;

begin
      makegrid(0,0,1,3,0,2,0,1,1,4);
      place r1 on (allx 0,3 ,ally 0,5 );
      place r2 on (allx 7,10 ,ally 0,5 );
      route in1.g to r1.inb.g;
      route in1.p to r1.inb.p;
      route r1.r.vd to r2.1.vd;
      route r1.r.gd to r2.1.gd;
      route r2.r.vd to rb.vd;
      route r2.r.gd to rb.gd;

      route c1out up layer metal right to r2.gout1;
      route r1.inr.p right 3 layer poly down 3 layer metal
            to in0.p;
      route layer metal r2.inr.g to c0out;
      route layer metal r2.inr.p right 1 down to rb.g
end
```

This specification contains a lot of information about cell basic. Let us begin with the template. Again the y-pins are matched up against each other, but this time the x-pins on the top and bottom sides are also partially constrained. As can be seen from the figure cin must have the same x-coordinate as cOout and out.p cannot come to lie to the right of c1out. This is stated in the constraints section of the template. If this specification were not given the processing of the cell basic would terminate with an error, because the system checks whether all the freedom among pins allowed by the template can be realized by the cell layout. This is important since this freedom may be exploited higher up in the chip hierarchy. In this way we assure that the layouts specified hierarchically really resemble the planar chip layout after all cells are expanded. Thus the designer does not get trapped into forming faulty images of what his layout looks like after expansion.

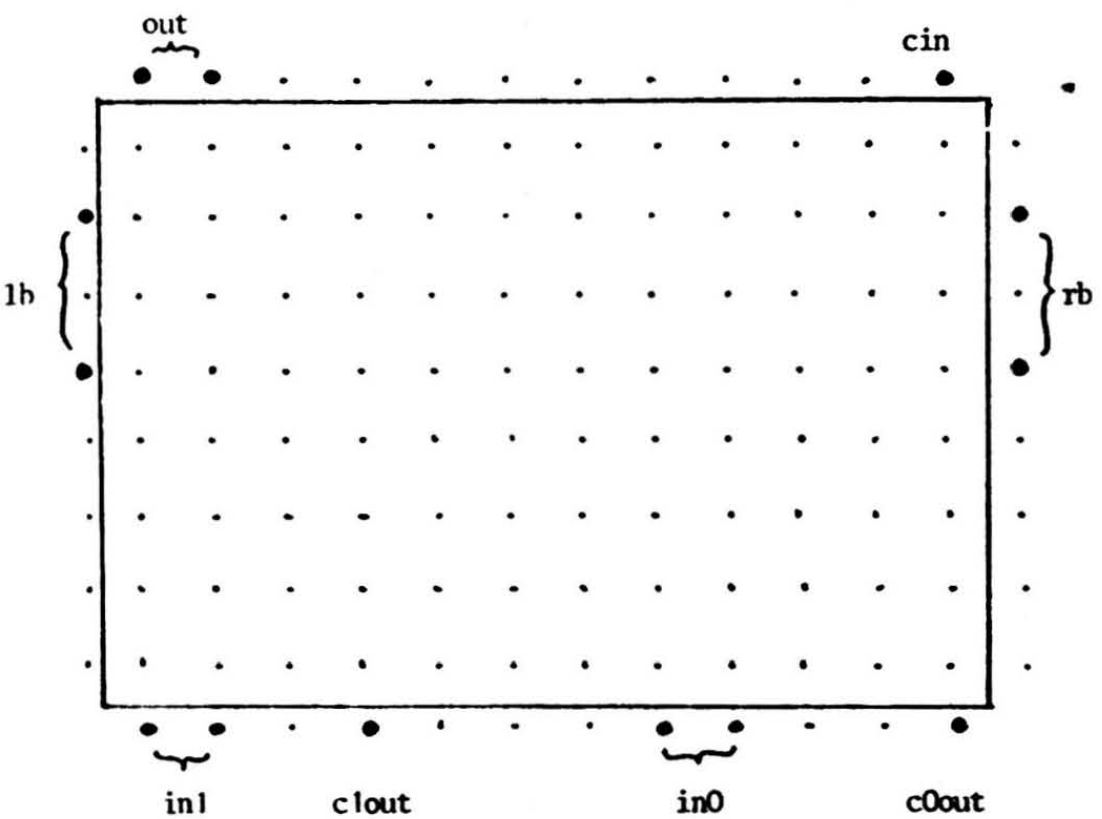
The constraints on the pins allow several layouts of the cell interior. These layouts can be transformed into each other by a process called non-uniform stretching. In order to keep this transformation feasible HILL restricts cells to allowing this freedom of pin placement only in one dimension. In the other dimension the pins have to be ordered linearly; no freedom of relative placement exists.

The layout specification following the template gives an example layout for the cell. This layout is assembled on a square grid. Here pins always come to lie on grid points and wires always come to lie on grid lines. First the pins are placed on grid points by the same mechanism that is used for matching up pins on opposite sidea of a template. This happens in the section following the keyword extend order to.

After processing this section the system provides the following grid for placement:



Now the components to be placed on the grid are declared. They are two cells r1 and r2 of type op(z), with their pins matched as given by the following extend order to section. (A consistency check is made here.) Then the placement begins. First additional grid lines are created to expand the grid to the required size. (This can also happen dynamically during the placement.) This yields the following enlarged grid.



Then the cells r1 and r2 are placed by specifying the grid lines that are covered by their templates. (Here these grid lines are simply given as sublists of the list allx off all vertical grid lines resp. the list ally of all horizontal grid lines. However, one can also form lists of non-adjacent grid lines, thus stretching the cell before placement.) A quite elaborate routing statement allows to route wires flexibly and to create contacts between layers implicitly where needed.

Finally we have to recursively put the cells of type basic together. This is done completely differently from the specification of cell basic. We use composition mode here. In composition mode no grid is used, but existing cells are copied and plugged together in x-direction using the xx-operator or in y-direction using the yy-operator. For this operation to succeed the corresponding sides of the cells have to match.

The recursive specification looks as follows:

```
cell recadd (n: int);
temp over: cbits;
     cin: poly;
     data: array [0..2^n-1] of record in: cbits; out: poly end;
     psl, psr: array [1..n] of ps;
     order  implicit y;
     sides top out, cin;
           bottom data reverse;
           left psl;
           right psr;
     constraints begin cin above data 0 .out;
                       out.p leftof data[2^n-1-1].out
                 end
pmet
```

```
composed
begin if n≥2 then
        recadd := basic(n) yy (recadd(n-1) xx recadd(n-1))
      else
        recadd := basic(n)
end
```

4. Additonal features of the HILL Specification Language

The HILL specification language entails a number of features that have not been used in the above example. These will be described in the following.

4.1 Maintenance of Electrical Information

HILL has the concept of an electrical signal. Electrical signals are associated with points in the circuit and grouped into classes by the link-statement to form electrical networks. The designer can build up completely or in part the electrical network he wants to realize with a layout. The HILL system encompasses tools for checking whether the network realized by the layout is the same as the intended network. HILL is also designed to be augmented with automatic layout routines that realize an intended electrical network with an automatically genrated layout. Only a PLA generator exists at this point. Global signal names allow to implement naming conventions for signals that pervade the whole specification hierarchy (like power supply and clock signals).

4.2 Control over the Proportions of the Mask Layout

The cell concept used in HILL ensures complete control over the topology of the symbolic layout, even when specifying a chip hierarchically. However, control over the mask layout is only indirect, since an automatic compacter determines the final proportions of the mask layout, and the sizes of cells in the mask layout are not known when specifying a chip hierarchically in a top-down fashion. However, floor planning and performance optimization of a layout require direct influence on a subset of the mask data. For this purpose HILL provides the keep statement. The keep statement formulates explicit constraints that

have to be obeyed during compaction. The existence of the keep statment also allows to include pre-compacted cells into a symbolic layout specification. This is the basis for compatibility with cell libraries generated with other layout systems and for hierarchical compaction to be implemented in the future.

4.3 Top-Down Design

HILL supports top-down design of chips in several ways. One of them has been mentioned in 4.2. Another is the strict separation of the template and the interior layout of a cell. This allows to use subcells in a hierarchical specification, even if they have not been designed yet. The detailed information given in a template serves as a guide both to the user of the subcell and to the designer who has to fill in the layout of the subcell independently. A fill statement is used in HILL to fill the empty template with a layout that makes up the interior of the cell. However, for floor planning not all templates have to be filled for the chip to be processed.

5. Conclusion

The HILL specification language is a powerful tool for describing planar circuit layouts. It has a cell concept that is flexible, yet keeps track of all the relevant topological information that has to be transferred across cell boundaries. This enables the designer to go through with a truly hierarchical design. HILL allows to choose between three modes of layout description (graphical, by composition, by explicit layout) that cater to different design styles and tasks. It provides the full mechanism of a high level language to allow the algorithmic description of layouts that a regular, yet non-trivial. Furthermore it separates electrical, topological, and geomterical information, which results in excellent interfacing of the different design phases (circuit, layout).

Supplemented with an efficient switch level simulator and a powerful compacter we believe that the HILL system serves well for the hierarchical specification of symbolic layouts.

6. References

[CABBAGE] M.Y.Hsueh, "Symbolic Layout and Compaction of Integrated Circuits", Ph.D.thesis, EECS-Dep. University of California, Berkeley, CA (1979).

[FLOSS] R.A.Auerbach,B.W.Lin,E.A.Elsayed, "layouts for the Design of VLSI Circuits," Computer Aided Design 13,5 (1981) 271-276.

[GV82] L.Guibas,J.Vuillemin, "On Fast Binary Addition in MOS Technologies," Int. Conf. on Circuits and Computers (1982) 147-150.

[HILL82] T.Lengauer,K.Mehlhorn, "HILL- Hierarchical Layout Language, A CAD System for VLSI Design," TRA82/10, FB10, Univ. d. Saarl., Saarbrücken (1982).

[HILL83] T.Lengauer, K.Mehlhorn, "Report on the HILL Specification Language," TR A83/05, FB10 Univ. d. Saarl., Saarbrücken (1983).

[L82a] T.Lengauer, "On the Solution of Inequality Systems Relevant to IC Layout," 8th Workshop on Graphtheoretic Concepts in Computer Science (WG82)(ed. W.Schneider) (1982).

[L82b] T.Lengauer, "The Complexity of Compacting Hierarchically Specified Layouts of Integrated Circuits," 23rd FOCS (1982) 358-368.

[L83] T.Lengauer, "Efficient Algorithms for the Constraint Generation for Integrated Circuit Layout Compaction," 9th Workshop on Graphtheoretic Concepts in Computer Science (WG83) (ed. M.Nagl) (1983).

[LM83] T.Lengauer,K.Mehlhorn,"VLSI Complexity, Efficient VLSI Algorithms, and the HILL Design System, TR A83/03, FB10, Univ. d. Saarl., Saarbrücken (1983).

[MC80] C.Mead,L.Conway, Introduction to VLSI Systems, Addison-Wesley (1980).

[MNN82] K.Mehlhorn,S,Näher,M.Nowak, "HILLSIM- Ein Simulator für MOS Schaltkreise," TR A82/08, FB10, Univ. d. Saarl., Saarbrücken (1982).

[MULGA] N.H.E.Weste, "MULGA- An Interactive Symbolic Layout System for the Design of Integrated Circuits," The Bell System Technical Journal 60,6 (1981) 823-857.

[PASCAL] K.Jensen,N.Wirth, PASCAL User Manual and Report, Springer Verlag (1975).

[STICKS] J.D.Williams, "STICKS- A Graphical Compiler for High Level LSI Design," Nat.Comp.Conf. (1978) 289-295.