A Method for Analog Circuits Visualization

Bogdan G. Arsintescu
Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
Bogdan@CaS.ET.TUDelft.nl

Abstract

A method for visualization of analog circuit schematic starting from a netlist is described in this paper. The images obtained have a comprehensible form, symmetrical structures and other analog topological patterns being discovered in the circuit netlist. The method is based on two algorithms. The first one, the symmetry algorithm, exploits the possibility to extract topology information from a graph representation of the circuit. The second one performs a grid expansion on a directed graph structure, using the results of the symmetry algorithm. The method has been implemented in the "A-warD" program. Test-cases demonstrate the effectiveness of the method.

1. Introduction

In recent years, automated methods for the design of analog integrated circuits have been proposed [1, 2, 3]. These methods helped the IC designers in handling the increasing complexity of the IC designs. Late stages of the IC design process, i.e. the layout design, are usually repeated several times until an acceptable layout is obtained, due to the specific constraints of analog circuits. Analog circuits are more sensitive to layout parasitics than digital ones. One method to detect these unwanted influences easier is by examining a netlist of the circuit extracted from the layout. For analog circuits, a visual inspection approach can be more helpful than a simulation in many cases. Several approaches have been made to develop tools for extracting a netlist with parasitics from a integrated circuit layout [4, 5].

Analog circuit topology is very important in the interpretation of the netlist. Stray capacitances of critical nets must be found in the netlist and matching between the devices is to be checked. Therefore a tool for analog circuit visualization starting from a netlist would represent a major help for the IC designer. The designer uses circuit schematics in the early stages of the design, so that a tool capable of re-drawing an image close to that schematic (i.e. in a human comprehensible form) would be very helpful. To our knowledge, only one approach was reported so far [6] for this topic. This method, however, can be used only in digital circuitry visualization. The image consists of library cell modules displayed as boxes. No methodology has been proposed so far for general analog circuits topology search and visualization.

In this paper we introduce a methodology for re-drawing analog circuits. Our aim is to create a "nice" circuit image using only the circuit netlist. This circuit's image will give the IC designer the opportunity to check "at a glance" the circuit structure and the capacitances of the critical nets. This method can drastically reduce the amount of time usually spent on debugging of analog circuit designs. Our method works for CMOS netlists as well as for Bipolar ones. For reason of convenience we will refer only to MOS transistors (MOSTs), but one can also consider Bipolar circuits by replacing the name of the corresponding transistor terminals.

Our method is based on a two step approach. First we identify analog circuits specific topological information. For this purpose, we introduce an algorithm capable of finding symmetrical pairs in a general circuit structure without user interaction, firstly described in [7]. Furthermore, this algorithm identifies current mirror and biasing groups patterns. Subsequently, all this topology information is transferred to the second algorithm, which produces a correct-by-construction circuit symbols placement by means of grid expansion. The results of this placement process are routed by a simple router. The placement and routing methods emphasize on graphical appearance rather than on compaction.

The method will work on general circuit graph structure, i.e. the input circuit does not have to be fully symmetrical, nor should the circuit graph be a planar graph. All the parasitics extracted from the netlist can be also placed in the final drawing. On this circuit image the designer can easily detect the unwanted parasitics or other layout errors. The time spent on debugging is reduced. Small error margins for circuit parameters can be detected using the proposed

This paper was publised in *Proceedings of the ICCD 1996*, October 7–9, Austin, Texas, section 3.3.2.

method, also.

This paper is organized as follows: Section 2 provides an overview of the method. In Section 3 and 4 the proposed algorithms for pattern search and constructive placement respectively are presented. Experimental results are reported in Section 5, and finally some conclusions are drawn.

2. General Description of the Method

Analog circuits are usually smaller in size than the digital ones. Complexity increases usually (e.g. for amplifiers) by cascading several stages preserving widely known topologies (e.g. differential structures, cascode connection). New designs rarely consists of new architectures, usually they are better combinations of old ones. Given an analog circuit structure, a differential design uses three basic topological patterns:

- **simple current mirror pattern:** a set of MOSTs with the gates connected to the same net and the sources connected to the same net, accordingly.
- **biasing groups:** a set of MOSTs with the gates connected to the same net and one of them connected as a diode.
- **symmetrical pair:** a pair of identical MOSTs connected with the same terminal to the same net, or to corresponding terminals of another topological symmetric pair.

Since the first two patterns can be described by the definition of the biasing group, both of them will be hereafter referred to as *mirror pattern*.

Our method for creating a circuit schematic image starting from a netlist is derived from the steps a human operator performs for the same procedure. First, we identify symmetrical pairs, and we impose that their symbols are placed face mirrored. Then, for simple mirrors patterns, we attempt to place them on the same line, usually back mirrored. Finally, we place these groups on a grid. The placement is optimized by minimum wire length and minimum number of wire bends. In this respect we propose two novel algorithms.

The Symmetry Algorithm. This algorithm finds all above mentioned patterns in the circuit structure. Given a CMOS circuit, a bipartite adjacency graph is constructed as follows. The "left" set of vertices correspond to the set of circuit nets and the "right" set of vertices is the set of MOSTs terminals. An edge between two vertices exists if the corresponding net of the start vertex is connected to the corresponding MOST terminal vertex.

This structure helps in identifying the symmetrical groups in the circuit. The algorithm starts by finding a symmetrical pair of MOSTs connected to the same net. Subsequently we try to augment the group with another symmetrical pair connected to the same cycle node of the previous pair. The procedure stops when no other symmetrical pair can be appended to any of the cycle nodes of the symmetrical chain of pairs. Figure 1 (b) show the symmetrical pairs chains obtained for the Cascode OTA in Figure 1 (a). The symmetry axes are drawn with dotted lines, and the pair of each element is its mirror image with respect to the axis.

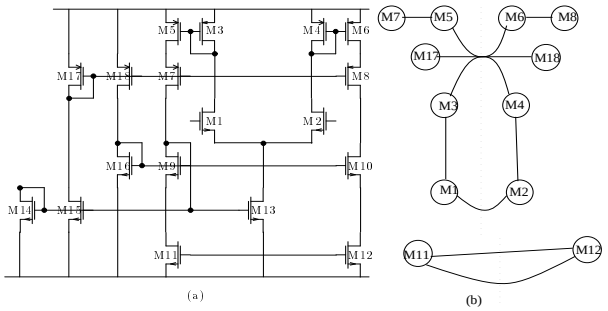

Figure 1. Circuit symmetries graph (a) COTA schematics (b) symmetrical groups

The algorithm has the following characteristics:

- It handles general graph structures, i.e. not only completely symmetrical circuits,
- It chains the symmetrical pairs, so that it discovers a complete symmetrical structure, not only pairs of elements
- It finds all the symmetrical groups even if they are not part of the same chain.

This algorithm finds all the set of MOSTs corresponding to mirror patterns also, searching firstly for a MOST connected as a diode and then for all the MOSTs with the gate connected to the same net as the gate of the former. All this topological information about the input circuit is given to the constructive placement algorithm.

The Constructive Placement Algorithm. The second proposed algorithm needed in order to achieve our goal is the constructive placement algorithm. This algorithm constructs a directed circuit graph $G(V, E)$, its vertices V being the electrical nets of the circuit, and its edges linking two vertices if the associated vertices are the source and drain of the same MOST, respectively. The direction of the edge is the direction of the current flow, that is from source to drain or vice-versa for p-MOSTs and n-MOSTs, respectively. The graph structure of the Cascode OTA in Figure 1 (a) is presented in Figure 2 (a).

The groups identified in the previous algorithm, i.e. mirror patterns and symmetrical pairs, are referred to as $rows$. This name was chosen because they will appear at the same horizontal level in the final image. We merge these rows until every circuit element is contained in one row only. We add dummy nodes and edges to this graph (with dotted lines in Figure 2 (b)) until every row represents a cut in the graph

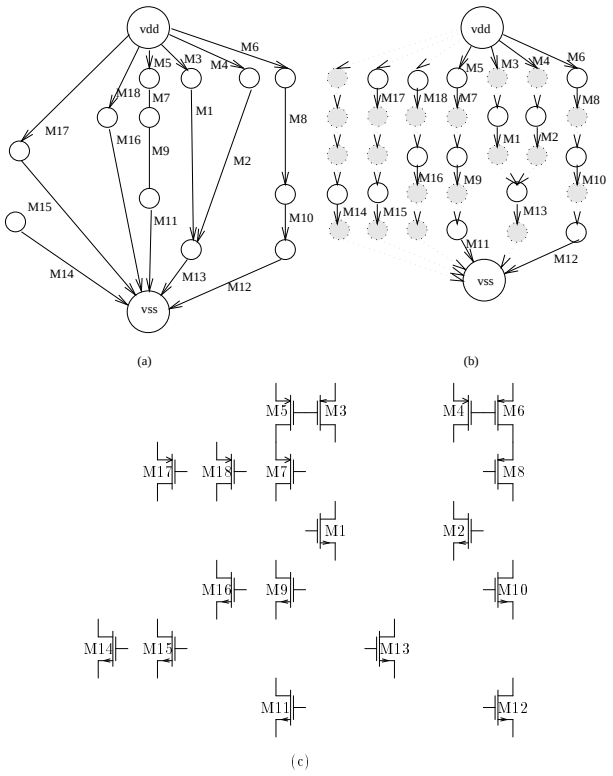

Figure 2. Circuit graph (a) before grid expansion (b) after grid expansion (c) circuit symbols matrix

$G(V, E)$. The new $rows$ are mapped as rows of circuit elements in the circuit schematic drawing with empty spaces corresponding to dummy edges.

We use some heuristics (described in Section 4) to insure the desired order and the relative orientation of the elements in a row. Subsequently, the graph is mapped on a matrix of circuit symbols. Each row in the mapped matrix corresponds to a row of edges, and every column corresponds to a path between the power supply corresponding nets.The corresponding matrix for the graph in Figure 2 (b) is presented in Figure 2 (c).

This algorithm requires only pointers to power nets, i.e. V_{dd} and V_{ss}. The output net pointer is optional, and in case it is specified, the possibility that the schematic drawing or mirror groups may be mirrored around the symmetry axes is eliminated.

To complete the task, we perform a detailed routing. Since we have only mono-dimensional blocks that are placed in fixed positions, the circuit nets can be drawn between the modules. Every multi-terminal net is transformed into a Steiner tree [8], and the Steiner points are marked by a small dot. After routing, the results are provided to the user in a structured graphical language as a set of objects

```
Algorithm: SYM
  INPUT: circuit bipartite graph
  FOR A = every net in the circuit
    FOR B = all MOSTs connected to A
      Mirror-Group = WBM(A, B, Cost-Mirror)
      Symm-Group = WBM(A, B, Cost-Symmetry)
      IF (Symm-Group ≠ ∅)
        Symm-Group = Augment(Symm-Group)
      END-IF
      SAVE Mirror-Group, Symm-Group
    END-FOR
  END-FOR
  OUTPUT Mirror-Group(s) and Symm-Group(s)
END_main

augment(S)
  FOR S =     the pair of nets connected
              to a symmetrical pair of MOSTs
    FOR B = all MOST terminals connected to S
      New-Symm-Group =
                  WBM(S, B, Cost-Symmetry)
      IF (New-Symm-Group ≠ ∅)
        Concatenate S with New-Symm-Group
        New-Symm-Group =
                  Augment(New-Symm-Group)
      END-IF
      RETURN New-Symm-Group
    END-FOR
  END-FOR
END_augment
```

Figure 3. Symmetry Algorithm

characterized by position and orientation and a set of wires. These results can be visualized with a compatible graphic package. The details concerning the graphical language will not be emphasised in what follows.

3. Symmetry Algorithm

An algorithm capable of recognizing symmetry in a fully symmetrical electronic network graph was proposed in [9]. Analog circuit design seldom leads to fully symmetrical circuits. We introduced an algorithm that finds all symmetrical groups according to topology patterns specific for analog circuitry in [7]. Our algorithm can handle general circuit structures, using the information in the adjacency bipartite graph as defined in Section 2. We use the Weighted Bipartite Matching (WBM) algorithm [10] with special cost functions defined for each pattern definition.

The patterns that we are searching for are symmetrical pairs and mirror patterns, as defined in Section 2. The cost

functions are Boolean, i.e. return a true value only if the searched pattern structure is found. These two patterns will match with almost all transistors in a CMOS analog circuit. Our tests on 30 circuits of up to 60 MOSTs showed that at most two transistors could not be included in any pattern.

The algorithm is outlined in Figure 3. WBM input list of nodes are a net A or a list of nets S on one side and the list of MOSTs terminals B on the other side. The WBM algorithm will return a set of MOSTs corresponding to a pattern found with respect to the cost functions. These cost functions, i.e. $CostSymmetry$ and $CostMirror$, heuristically evaluate the weights of the edges based on the pattern definitions.

The $CostMirror$ function will return a non-zero value only if the mirror pattern is found among the set B of MOSTs connected to the net A. Since mirror patterns correspond to bias-group definition, we do not make any difference between these two patterns. The $CostSymmetry$ function will return a positive value if one or more identical circuit element pairs are found among the set B of MOSTs connected to the other set by a circuit net, only. The cost function compares MOSTs dimensions also. For reason of clarity of the image, this detail was not included in Figure1 (a). One modification with respect to WBM is that the algorithm will return a nonempty result only if the result of the cost function is positive. WBM results with cost zero are not interesting, since they are not a result of a pattern matching with respect to the cost function.

The $AugmentPath$ function will recursively search for new symmetrical pairs connected at the same terminal of an already found symmetrical pair. Once a symmetrical pair connected to the same net is found, our algorithm grows a symmetrical graph. The symmetry axes of the group crosses the initial net and any other net common to both elements in a symmetrical pair. In Figure 1 (b) are shown two symmetrical graphs found for the Cascode OTA (Fig 1 (a)). One can see that multiple symmetrical pairs can be found for the same net, i.e. for V_{dd}, the pairs are [M3-M4], [M5-M6], [M17-M18]. The symmetries discovered by this algorithm are pair of elements symmetrical with respect to netlist structure rather than circuit behavior. However, the set of topological symmetric elements includes the set of behavioral symmetric pairs, i.e. our algorithm may enforce more symmetries than necessary.

The depth of the recursive procedure is at most $n/2$, where n is the number of MOSTs in the circuit in case the circuit is completely symmetrical. Our algorithm avoids duplicating symmetrical structures. The maximum number of $AugmentPath$ calls is, therefore, $n/2$. The maximum number of augmenting attempts in every call of $AugmentPath$ is $\frac{m(m-1)}{6}$ (where m is the number of MOSTs connected to a pair), since the net has to be connected to a certain terminal of the circuit element. Because the circuit graph is sparse and the symmetrical graph obtained is hardly bifurcated, the SYM algorithm described above is expected to perform in linear time.

```
Algorithm CPA
  INPUT: Circuit graph G(V, E)
  L_v = V_dd
  DO
    G(V, E) = GRID-EXPAND(L_v)
  WHILE L_v ≠ V_ss
  FOR each row between V_dd and V_ss
    Arrange-Row(row)
  END-FOR
  OUTPUT Circuit graph G(V, E)
END CPA
Algorithm: GRID-EXPAND
  INPUT: L_v list of vertices
  FOR e = every out-coming edge from L_v
    FOR S = any row with an out-coming edge from L_v
    IF (row(e) ≠ S)
      Create New Node N;
      Create Edge e* between
            the source vertex of e and N;
      Move source vertex of edge e to vertex N;
      Add e* to S
      Add e* to G(V, E)
    END-IF
  END-FOR
  OUTPUT new circuit graph G(V, E)
END GRID-EXPAND
```

Figure 4. Constructive Placement Algorithm

4. Constructive Placement Algorithm

The final position of the MOSTs in the circuit schematics is decided at the end of the Constructive Placement Algorithm (CPA). The algorithm is outlined in Figure 4. This algorithm can work only if the user gives specific pointers to V_{dd} and V_{ss} nets, since the vertices associated to these nets are the start and end points of the algorithm. For reasons of clarity, we will refer to the directed circuit graph $G(V, E)$ in terms of its circuit correspondence.

Before starting this algorithm, we have to process the information we obtained from the Symmetry algorithm. All elements grouped in the same pattern will be in the same row. The row definition is introduced in Section 2. If any circuit element appears in two different rows, we merge these rows in a single one, containing all elements of the input lists only once. When no other two rows can be merged, we can start the constructive placement procedure.

We chose any row among those having an edge outcoming from V_{dd}. *GRID-EXPAND* adds a new node be-

tween all the edges not common to the selected row. We move the "undesired" edge source from its initial point to the new node and we create a new edge (in order to preserve the graph connectivity), which is added to the current row. The *GRID-EXPAND* function will return a list of nodes corresponding to the end points of actual row of edges. This list is the input of a new call to *GRID-EXPAND*. In every new step we select a new row with an edge out-coming from one node of this list. The operation ends when all rows have been expanded, and their end points are V_{ss}.

A special case are edges that are not contained in any path (chain of edges) between V_{dd} and V_{ss} corresponding vertices. We create dummy nodes and edges for all rows already processed until a path exists.

The graph changes are illustrated in Figure 2: while the initial graph (see Figure 2 (a)) has a general structure, the final one (see Figure 2 (b)) has a regular structure. Only the vertices corresponding to nets on the symmetry axis of a centroidal group can connect two paths between V_{dd} and V_{ss} corresponding vertices.

The function *Arrange-Row* changes the order of the paths in the graph according to a set of heuristic rules derived from circuit topology. The first criterion is that the mirror groups should have the MOST connected as a diode to the right in the figure. The second one is that between the two elements of a symmetrical pair there is either another symmetrical pair, or no MOST at all. In our approach we also avoid dummy element insertion between symmetrical pairs if possible, because this results in extra wire crossings during routing. Another heuristic is used in case one row contains elements from two different mirror or biasing groups. Note that, in this case only, the placement of one of the mirror is breaking the first rule. The last heuristic completes the number of rules necessary to reorder the paths. This affirmation is based on the following property.

Property 1 *The maximum number of mirror groups per row is two.*

Proof: This observation is based on the following remark: No row can contain MOSTs from more than four different patterns, i.e. two mirror groups and two symmetrical pairs. This can be proved since the following statements are true simultaneously:

1. Current mirrors and bias-groups do not have common MOSTs.
2. Symmetrical pairs do not have common elements
3. Different stages of the circuit are not biased from the same group

The first two criteria are always true by definition, the third one is true only for correctly designed circuits. Their mixed constraints result in the bound of four. Since the connection between two mirror or biasing group can be a symmetrical pair of MOST only, then we should have at least one symmetrical pair. The fourth element cannot be a mirror, because another symmetrical pair is needed to connect them. In this case we will exceed the bound of four elements per group. The maximum number is four, i.e. two mirror groups and two symmetrical pairs. □

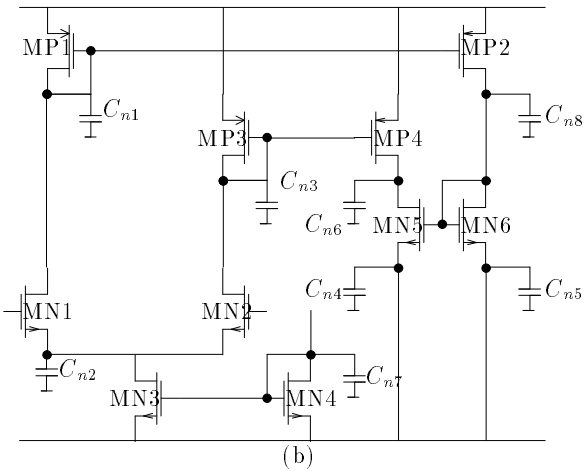

Figure 5. OTA image with stray capacitances

This arrangement procedure insures correct placement and orientation for each MOST. The dummy edges will be considered as void circuit elements.

This algorithm will perform on general graph structures, because edge intersection will occur between dummy edges only. The complexity of *GRID-EXPAND* algorithm is $O(n)$, where n is the number of distinct rows. *Arrange-row* complexity is $O(m)$, where m is the number of different groups that have corresponding edges in the row.

5. Results

The work described in this paper was implemented in the program called "A-warD" (Analog Reverse Draw) written in the "C" language. As input for the program we use SPICE netlists, and the PicTEX graphical language package [11] together with "eepic" extension [12] for the output circuit's image description. This program is included in our analog layout framework as a tool for debugging the layout.

The functionality was tested on several medium sized analog circuits. For our study cases of about 30 medium size circuits, "A-ward" always generated a comprehensible image, identical or very close to the one used at the behavioral design level.

The image in Figure 1 (a) was generated by the "A-Ward" program. In what follows, we will present two other practical examples. The first example presents an OTA circuit structure (see Figure 5), together with the stray capacitances. One extra feature that will be added to "A-ward" is that the

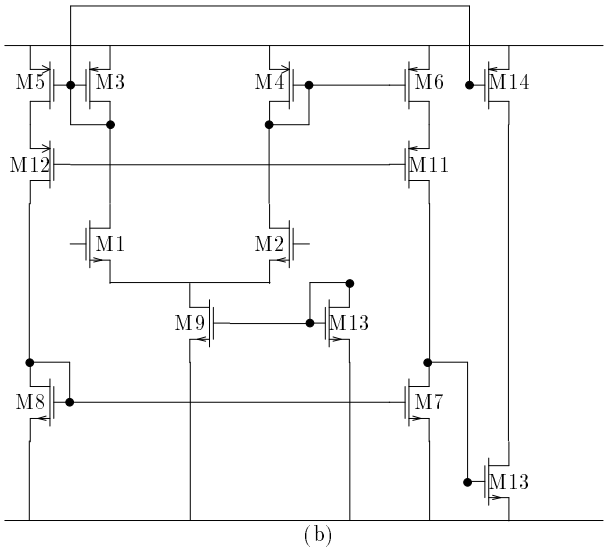

Figure 6. High PSRR OTA image

user can choose the set of parasitics she want to visualize together with the circuit schematics (e.g. stray capacitances, crosstalk parasitics).

In case parasitics are to be displayed, the grid is slightly enlarged and parasitic symbols are placed in the image as part of the MOST symbol, that is, they are not handled separately by the algorithm and are considered a part of the transistor. As of now, our visualization method implemented in "A-warD" can handle netlists containing only transistors. However, the extension to netlists containing other circuit elements is straight forward.

The second example (Figure 6) presents a High PSRR OTA. Note that first upper row of elements contains two symmetrical pairs and two mirror patterns, so that the mirror group [M3-M5] is placed according to the special rule discussed in the previous Section. One can notice that room for compression exists, but our aim was to emphasize clarity of the image rather than size, since "A-ward" is a visualization tool.

6. Conclusions and future work

An original approach for analog circuits visualization was presented. Using the new algorithms and heuristics we described, it is possible to generate circuit schematics similar to hand-drawn ones. The designer can visualize node capacitances and other parasitics from a layout extracted netlist.

The method is based on a symmetry finding algorithm, and a correct-by-construction placement and routing procedure. Three circuit examples were reported showing the results that can be obtained with this method.

Combined with a Layout-vs-Schematic algorithm, our tool can be very useful in verification of the design. Future development will target this idea, together with expansion of the pattern matching to digital networks and switching capacitors circuit structures.

This method was implemented in the "A-warD" program and is included in the analog design framework in development at Delft University of Technology.

References

[1] E. Malavasi and D. Pandini. Optimum CMOS stack generation with analog constraints. *IEEE Trans. on CAD*, 14(1):107–120, Jan. 1995.

[2] S. S.W. Mehranfar. A technology-independent approach to custom analog cell generation. *IEEE J. Solid-State Circ.*, 26(3):386–393, Mar. 1991.

[3] J. Cohn, D. Garrod, R. Rutenbar, and L. Carley. *Analog Device-level Layout Automation*. Kluwer, 1994.

[4] F. Beeftink, A. J. v. Genderen, and N. P. v. d. Meijs. Accurate and eficient layout to circuit extraction for high-speed mos and bipolar/biCMOS integrated circuits. In *Proceedings of the ICCD*, pages 360–365, 1995. URL http://cas.et.tudelft.nl/research/space.html.

[5] K. Belkhale and P. Banerjee. Parallel algorithms for vlsi circuit extraction. *IEEE Trans. on CAD*, 10(5):604–618, May 1991.

[6] J. Russack. Voyeur - a circuit visualization tool. Home-page of the "Voyeur" visualization tool at http://www.cse.ucsc.edu/voyeur/html/voyeur.html.

[7] B. G. Arsintescu and S. A. Spânoche. Global stacking for analog circuits. In *Proceedings at Euro – DAC*, 1996.

[8] T. Lengauer. *Combinatorial Algorithms for Integrated Circuit Layout*. Wiley, 1990.

[9] M. Kole and O. Herman. Modeling symmetry in analog electronic circuits. In *Proceedings ISCAS*, pages 315–318, 1994.

[10] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to algorithms*. McGraw Hill, MIT Press, 1992.

[11] M. Goossens, F. Mittelbach, and A. Samarin. *The LaTeX Companion*. Addison-Wesley, 1994.

[12] C. Kwok. EEPIC: extension to epic and LaTeX picture environment. User guide to "eepic" package.