GORDIAN

User's guide

Georg Sigl, Konrad Doll, Henning Spruth

July 11, 1995

Institute of Electronic Design Automation
Technical University of Munich
Professor K. Antreich
Arcisstr. 21, 80290 Munich, Germany
Tel.: (+49 89) 2105 – 3666

Contents

1 Introduction

This manual describes the commands and parameters used to control the placement program GORDIAN. The following conventions are used: words written in typewriter style (i.e. `in this style`) must be typed exactly as they appear in this document, whereas words written in italics (i.e. *in this style*) must be replaced by the actual name used.

2 First time use

If you are using GORDIAN for the first time you may proceed as follows:

1. Create a directory *circuit_name/*.
2. Copy the description of the netlist and of the cell library in this directory. If you are using GORDIAN outside the SPACE system, it is most convenient to genereate these data in the Proud-format as described in Appendix A. A converter from VPNR [oas90] to Proud is described in Appendix B.
3. Create directories `input/, output/` and `daba/` (`daba/` is necessary only for the use inside SPACE).
4. Copy a control file as described in Section 8 in the directory `input/`. For a first run, it is appropriate to take the minimal control file as shown in Table 5. In a second run, if you want improved quality add the control word `DYNNET` (see Table 7).
5. Start GORDIAN as described in Section 3 in the directory *circuit_name/*. For this first time run, the options `-pscreen -pnonstop -pmst` are recommended to give some impression how GORDIAN works. For other options see Section 4.
6. Look at the protocol file of the run, which is in the directory `output/`.
7. The placement is written in a file `place.`*controlfile* as explained in Section 7.

3 Program invocation

On Unix systems, GORDIAN can be invoked like this:

`gordian` *[-options] controlfile*

where *-options* consists of a list of desired options preceded by minus signs (see Section 4), and *controlfile* consists of the name of the file containing the control parameters *without* the suffix `.mc` (see Section 8).

At the moment of the call, the directory from which the program is invoked must contain the subdirectories `input` and `output` and the control file named *controlfile*`.mc` must be in the directory `input`. Within the SPACE-System a directory `daba` is required additionally.

If the options for graphics output are to be used, the program XOUTPIPE must be in the current search path.

4 Options

The following options must be separated by blanks or tabs. Options beginning with `-p` control the graphic display on the screen.

`-b`	Batch mode. No output is generated on the screen. If the program terminates abnormally, an error message will be printed on the screen as well as written to the protocol file.
`-v`	Verbose mode. Detailed output on the screen.
`-vv`	Very verbose mode. Extensively detailed output on the screen.

These options do not affect in any way the output written to the protocol file, which remains the same with the three different options.

`-i`	Interactive mode. At the end of the placement of a non-row-bases sea-of-gates circuit, it is possible to interactively select the best placement result amongst those proposed by GORDIAN.
`-h`	Show help. Display a list of valid command line options and control words (the list in section 8 was generated that way) and stop.
`-pscreen`	After each level of the partitioning, a graphic representation of the current placement is displayed in a window of the screen. The distinction between each subset of the modules is made with the help of colors.

If graphic display is desired, then the option `-pscreen` should be specified. Graphic display is currently possible on terminals and workstations running under X-Window system. The actual X output is done by a separate program XOUTPIPE, which is started automaticcally by GORDIAN. Therefore, if graphics output is used, the XOUTPIPE program must be in the current search path when GORDIAN is started. This separation of the display task from the actual GORDIAN program has the advantage of asynchronous refreshes and a smaller program executable.

`-pps`	Generation of PostScript plot-files as input for the laser printer.
`-pnonstop`	No stops are made between each graphic display level.
`-pnolevels`	A graphic representation is produced only after the final placement, not at every level.
`-pall`	Displays the modules, the nets and their names.
`-pnets`	Displays the star model of the nets.
`-pmst`	Displays the minimal spanning trees for each net. (instead of `-pnets`).
`-pmodnames`	Displays the module names.
`-pnetnames`	Displays the net names.
`-pnoframes`	No module frames are displayed. It is faster and the colors are also easier to differentiate.
`-pmcsets`	The Min-cut initial partition of each region is displayed.

`-pninetydegrees` The display is rotated 90 degrees.

`-pnotfilled` The modules frames are not filled in.

For example, the program invoked with the option for graphic display of the placement and the minimal spanning trees after each partitioning step would look like:

`gordian -pscreen -pmst` *controlfile*

5 Input and Output

This section describes the input necessary for GORDIAN and the output generated by the program. GORDIAN needs information about the cells to be placed (the dimensions and pin locations of the cells), the net list, some geometrical data of the available placement area (e.g. the number of standard-cell rows or the basic cell grid of the sea-of-gates circuit) and some information about the location of the I/O connectors or pad cells. The circuit directory must contain some files (one or several) specifying this information.

GORDIAN may be used in different environments, which are specified in the control file by the necessary control words `DESIGN_FILE`, `DESIGN_FORMAT`, `TARGET_SYSTEM` (see Section 8).

The result of the placement is written by GORDIAN in the file `place.`*controlfile* or to a SPACE database.

Furthermore, GORDIAN generates a protocol file called *controlfile*`.mco` covering the entire run of the program (e.g. wiring length, new dimensions, processing time) in the subdirectory `output` of the circuit directory.

6 GORDIAN in the SPACE-environment

When GORDIAN is used in the SPACE-environment, the keyword `SPACE` must be specified for all of the control words `DESIGN_FILE`, `DESIGN_FORMAT`, `TARGET_SYSTEM`.

GORDIAN reads the circuit data from the following files of the `daba` directory.

glbea.asc Number of modules, nets and connected pins

glgeo.asc grid unit

glktg.asc part of the cell library used in the circuit

glbpl.asc integer module coordinates, module rotation and placement mark
`MODULE module[].xy[] .rotation .plmark`

glbto.asc number of pins of each module `MODULE module[].npins`

glkai.asc width, height, and type of a module `MODULE module[].wh[] .cattype`

glsin.asc information about the nets `NET net[]`

glgam.asc dimensions of the placement area, channel capacities, and pad locations

pnlst.asc list of pins

The following files will be created or changed by GORDIAN:

mckoo.asc center coordinates of the modules after each level of partitioning

sltree.asc slicing tree `STNODE stnode[]` und `STINF stinf[]`

glkai.asc *glgam.asc glbpl.asc*

7 GORDIAN with special design description in proud format

When GORDIAN is used with proud design description, the keyword `Proud` must be specified as `DESIGN_FORMAT`. The netlist will then be read from the file specified under `DESIGN_FILE`, e.g. `design.`*xy*. A file (e.g. `celllib.`*xy*, the name is taken from the file `design.`*xy*) describing the cell library must also be present in the current directory. If a file `netweight.`*xy* exists, net weights will be read from this file. The `TARGET_SYSTEM` may be `VENUS`, `SPACE` or `DOMINO` for standardcells, `MARS` or `SPACE` for Sea-of-Gates, respectively.

The resulting placement is written in a file `place.`*controlfile*. The format of the place-File is:

cell_name row position

Where *row* is the number of the row starting at the bottom of the circuit and *position* is the position of the cell within the row beginning at the left side.

If the control word `WRITE_SPACE_DABA` is specified, a full SPACE database will be created by GORDIAN.

8 The Control file

The control file must be present in the subdirectory `input` of the circuit directory. Moreover, to be recognized by GORDIAN, its name must end with the suffix `.mc`. This subsection contains a list of all the control words and a table indicating the validity of each control word with respect to the types of design style and target systems.

Most invalid set of control parameters will cause the program to stop. However, it is not fully guaranteed that some bizarre set of control parameters may not cause the program to produce a runtime error. Therefore, the user should carefully verify the validity of his/her control file before running GORDIAN. All parameters that are not specified in the control file are set to the default value.

The following list was generated automatically by invoking Gordian with the `-h` option. The entries marked with `(internal use)` are experimental and/or obsolete features that should not be used for production runs.

DESIGN_FILE [input] Specifies the file that contains the circuit description or SPACE for the SPACE database in the daba/subdirectory. Type: string of length 2000. Default value:

DESIGN_FORMAT [input] Specifies the format of the design database. This is a required parameter. Type: choice, one of PROUD, PRIZE,

ICLIB, SPACE, BOE, KLEIN, UNKNOWN_DES_FORMAT. Default value: UNKNOWN_DES_FORMAT

DESIGN_STYLE [input] Set the designstyle; this is a required argument. Type: choice, one of STANDARDCELL, SEA_OF_GATES, MACROCELL, UNKNOWN_DES_STYLE. Default value: UNKNOWN_DES_STYLE

DOMAINFILE [input] Specifies the name of the file containing the domains/clusters on the highest level of hierarchy. Type: string of length 2000. Default value:
(internal use)

READ_ROTATE [input] Rotate design by 90 degrees after reading netlist and before writing placement. Not implemented for all design formats. Type: implicit boolean. Default value: FALSE

TARGET_SYSTEM [input] Specifies target design system. This is a required parameter. Type: choice, one of SPACE, TOSHIBA, SOGSCX, MARS, DOMINO, VENUS, KLEIN, UNKNOWN_TARGET_SYSTEM. Default value: UNKNOWN_TARGET_SYSTEM

ASCII_OUTPUT [output] Write an additional ASCII placement file place.foo. Type: implicit boolean. Default value: FALSE

BIGPROT [output] The program produces a very detailed protocol file of the different actions of the program. WARNING: This option is not recommend for large circuits, since the amount of data written into the protocol file may slow down the program considerably, and may result in memory problems. Type: implicit boolean. Default value: FALSE

WRITE_SPACE_DABA [output] After reading the design, create a database in the internal SPACE format. Type: implicit boolean. Default value: FALSE

PLACEMENT_FILE [I/O] ? Type: string of length 2000. Default value:
(internal use)

COLPART [global] Partition the set of cells only horizontally until each region corresponds to a standard cell column. If an integer value is specified, the regions are then partitioned in the y-direction for as many times as the values specifies. When invoked, STANDARDCELL and CUTDIR=VERTICAL are automatically set. Type: integer. Default value: 0
(internal use)

CUTDIR [global] Sets initial cut direction; the default depends on the design style. Type: choice, one of HORIZONTAL, VERTICAL, UNDEF. Default value: UNDEF

DYNNET [global] Use a linearized objective function, which is solvedby iterative weight updates. This results in a significantly better placement and longer run time. Type: implicit boolean. Default value: FALSE

DNMINPINNUMBER [global] Minimum number of pins per net to leave the net node in the graph model. This is useful to avoid a high number of fill-ins in the system matrix, which would be generated by elimination of a net node with many connected pins. Type: integer. Default value: 15
(internal use)

DNMAXLEVEL [global] Highest level where the linear objective function will be used. If DNMAXLEVEL>1, this is the absolute level number. For 0.0 < DNMAXLEVEL < 1.0, the level is calculated as DMAXLEVEL times an estimated total level count. Type: floating point. Default value: 0.8
(internal use)

FINALLEVEL [global] Stop global placement after the given number of levels. Type: integer. Default value: 0
(internal use)

ICMAXLEVEL [global] Maximum level to perform iterative partitioning, if < 1.0, iterative cuts are performed until level=ICmaxlevel * estimated number of levels. Type: floating point. Default value: 0.0
(internal use)

ICITERATE [global] Global placement and iterative partitioning will be alternated ICITERATE times for each subregion. If DYNNET is specified, the following override defaults are used: ICITERATE = 3 for circuits with up to 10000 cells, ICITERATE = 5 for circuits with more than 10000 cells. Type: integer. Default value: 1

MAXPINSPERNET [global] Nets with more pins than specified in MAXPINSPERNET are eliminated, i.e. not considered during placement. However, a net whose removal would cause the circuit to be split into two or more disconnected components is not removed. Type: integer. Default value: 60

MINCUTPART [global] Specifies the proportion of the total area on each side of the cut where cells can be swapped to minimize the number of cut nets (range between 0.0 and 1.0). Type: floating point. Default value: 0.0

NETSNOTWEIGHTED [global] The weights of the nets are ignored. Type: implicit boolean. Default value: FALSE
(internal use)

NOFINALPLACEMENT [global] Only perform global placement. Type: implicit boolean. Default value: FALSE
(internal use)

REITERATION [global] Starting from the calculated module coordinates of the current level, the program will repartition the module set and recompute a solution to the equality system with the current centers of gravity. Type: implicit boolean. Default value: FALSE

RELCUT [global] The y coordinate of the cut is to be determined according to the relative placement of the cell (valid only together with COLPART). Type: implicit boolean. Default value: FALSE
(internal use)

RESIDBOUNDFACTOR [global] Stopping criterion for the iteration of the global placement solution. It is a factor of the starting norm of the residue (norm of the gradient) and indicates when the global placement can still be improved. Stop if: RES_last / RES_start <= RESIDBOUNDFACTOR. Type: floating point. Default value: 1.0e-4

SEMIMOVABLEPADS [global] Read semimovable pads, not currently implemented. Type: implicit boolean. Default value: FALSE
(internal use)

SHOW_MST_LENGTHS [global] After each global placement solution (before the following partitioning), the sum of the lengths of the MSTs of all nets are calculated and printed. Type: implicit boolean. Default value: FALSE

STEINERMODUL [global] Insert semimovable pseudo cell at the cut lines. obsolete? Type: implicit boolean. Default value: FALSE
(internal use)

WITHOUT_PRECOND [global] Turn off preconditioning during solution with conjugate gradients. Type: implicit boolean. Default value: FALSE
(internal use)

WRITE_EQSYS [global] The integer specifies the level of hierarchy for which the equation system and the constraints of the quadratic programming problem modeling the global placement will be written to disk. Type: integer. Default value: -1
(internal use)

ADJUSTPADS [final] Update the positions of the I/O pads at the end of cell placement. Type: implicit boolean. Default value: FALSE

REL_DOMAIN_AREA obsolete. Type: floating point. Default value: 0.5
(internal use)

DONT_EQUATE_COLUMNS [final] The heights of the columns are not evened out. Type: implicit boolean. Default value: FALSE
(internal use)

ESO_MODIFY_EPSILON [final] Maximum width of a strip where cells are rearranged using Enumerative Slicing Optimization, expressed as a fraction of width or height of the ESO region. Type: floating point. Default value: 0.0
(internal use)

ESO_MODIFY_MAX [final] Maximum number of cells to be taken together into a single ESO strip for rearrangement of cells according to coordinates of centers of gravity of the associated nets. Type: floating point. Default value: 1

EQUALDISTPADS [final] Together with ADJUSTPADS, distribute the pads equally over the sides of the chip. Type: implicit boolean. Default value: FALSE

PLACE_SOGBLOCKS [final] Don't use ESO to rearrange cell but place them unchanged. Type: implicit boolean. Default value: FALSE
(internal use)

PROTSHAPES [final] Produce a file containing the shapes of all regions during ESO. Type: implicit boolean. Default value: FALSE
(internal use)

RESETROTATION [final] The rotation angles of the cell are initially set to 0. WARNING: This control word is only valid when the design data is obtained from a database using the SPACE format. Type: implicit boolean. Default value: FALSE

RM_ESO_TREES [final] Free memory allocated during ESO shape function calculation. This reduces memory requirements but inflicts higher CPU times during the fixing phase. Should definetely be used for large circuits. Type: implicit boolean. Default value: FALSE

ROTATION [final] For macro cell circuits, try to optimize net length after ESO by rotating them 180 degrees. Type: implicit boolean. Default value: FALSE

ROW_INDENT [final] Controls the reduction in the heights of the rows relative to the average row length for standard cell designs (between 0.0 and 1.0) Type: floating point. Default value: 0.0

SHAPEFUNCTIONS [final] The maximum number of cells in a region to be rearranged during ESO. Higher values will lead to a more compact placement and vastly higher CPU and memory requirements. Typical values are from 10 to 40. Type: integer. Default value: 1

1_D_PLACEMENT [final] Perform one-dimensional placement for final placement in combination with COLPART. Type: implicit boolean. Default value: FALSE

ANALYSE_MODELL [timing] Misspelled synomyn for ANALYSIS_MODEL retained for backwards compatibility. Don't use this variant. Type: choice, one of STARM, HALF_PERI, HALF_PERI_WITH_W_RES, STAREL. Default value: STAREL
(internal use)

ANALYSIS_MODEL [timing] Model for timing analysis. STARM = Starmodel using the mean value of the upper and lower bounds as proposed by Penfield and Rubinstein. This results in a different pin to pin delay for all pins of one net. HALF_PERI = Half perimeter bouning box of all pins of a net neglecting the wire resistances. Leads to the same output delay for all pins of a net. HALF_PERI_WITH_W_RES = Half perimeter bouning box of all pins of a net considering the wire resistances. Leads to the same output delay for all pins of a net. STAREL = Using the Elmore delay formuls based on the center of mass star model. Type: choice, one of STARM, HALF_PERI, HALF_PERI_WITH_W_RES, STAREL. Default value: STAREL

TIMING [timing] Perform timing driven placement. Start to optimize a path only if the actual path delay exceeds the maximum path delay that is specified with MAX_ARRIVAL_TIME or in the design file. Type: implicit boolean. Default value: FALSE

MAX_ARRIVAL_TIME [timing] The maximum path delay to be tolerated if TIMING is set. If this parameter is set to -1.0, use the value given in the design file. Type: floating point. Default value: -1.0

OPTIMIZE_TIMING [timing] Perform timing optimization. In contrast to TIMING, the control word OPTIMIZE_TIMING tries to minimize the maximum path delay as far as possible by weighting critical nets right from the start if the placement phase. All nets which are contained in paths for which path_delay > CON * maximum_path_delay holds are considered critical. Type: implicit boolean. Default value: FALSE

CON [timing] Factor determining the number of paths which are concerned as critical paths. All paths where the path delay is bigger than CON * maximum_path_delay are considered as critical. Type: floating point. Default value: 0.99
(internal use)

TIMING_WEIGHT_W [timing] Net weighting factor in timing driven placement if WEIGHTING = PROPORTIONAL is invoked. Type: floating point. Default value: 1.0

MAX_WEIGHT [timing] Maximum net weight in timing driven GORDIAN. It is used as an upper bound to avoid extreme high netweights. Type: floating point. Default value: 100.0
(internal use)

WEIGHTING [timing] Weighting model for the nets in the timing driven GORDIAN Type: choice, one of PROPORTIONAL, LAGRANGE, PIN_W. Default value: PROPORTIONAL

FAST_BACK_PROP [timing] Use fast back propagation in timing analysis. Decreases coputational effort considerably at no loss of placement quality. Cannot be used together with WEIGHTING=LAGRANGE. Type: implicit boolean. Default value: FALSE

FIX_NUM_CP [timing] Specifies number of critical paths to be considered. number of critical paths = number of input pins of the primary outputs and the sequential cells - number of cell /FIX_NUM_CP; can only be used if FAST_BACK_PROP is set. Can also be used together with CON. In this case the smaller of the two values is used. Type: floating point. Default value: 0.0
(internal use)

WIRE_CAPACITY_X [timing] Specifies wire capacity in x-direction in pF/m. Type: floating point. Default value: 242.0

WIRE_CAPACITY_Y [timing] Specifies wire capacity in y-direction in pF/m. Type: floating point. Default value: 242.0

WIRE_RESISTANCE_X [timing] Specifies wire resistance in x-direction in Ohm/m. Type: floating point. Default value: 25.5e03

WIRE_RESISTANCE_Y [timing] Specifies wire resistance in y-direction in Ohm/m. Type: floating point. Default value: 25.5e03

MEASURE_FINAL_ROUTING [timing] Don't perform a placement. Just do a timing analysis of a final routed circuit. Type: implicit boolean. Default value: FALSE
(internal use)

MEASURE_RITUAL_RESULT [timing] Don't perform a placement. Just do a timing analysis of a final routed circuit which was placed by RITUAL. Type: implicit boolean. Default value: FALSE
(internal use)

END Last statement in controlfile. Type: implicit boolean. Default value: FALSE

The control words given in Table 2 must always appear in a control file to specify the circuit to be processed, while others are necessary only for some types or are optional at all.

For the placement of a circuit based on a specific cell design, the control file may only contain control parameters valid for that design. For example, a combination of `COLPART` and `SHAPEFUNCTIONS` is not permitted. Table 1 contains the valid control words for standard-cell and sea-of-gates designs.

Table 1: Valid control words

Standard-cell	*Sea-of-gates*
ADJUSTPADS	---
COLPART	---
CUTDIR	CUTDIR
DYNNET	DYNNET
DNMAXLEVEL	DNMAXLEVEL
DNMINPINNUMBER	DNMINPINNUMBER
EQUALDISTPADS	---
---	ESO_MODIFY_EPSILON
---	ESO_MODIFY_MAX
FINALLEVEL	FINALLEVEL
ICITERATE	ICITERATE
ICMAXLEVEL	ICMAXLEVEL
MAXPINSPERNET	MAXPINSPERNET
MINCUTPART	MINCUTPART
MINMCMOD	MINMCMOD
NETSIZE_WEIGHTED	NETSIZE_WEIGHTED
NETSNOTWEIGHTED	NETSNOTWEIGHTED
---	PLACE_SOGBLOCKS
PLACEMENT_FILE	PLACEMENT_FILE
---	PROT_SHAPES
REITERATION	REITERATION
RELCUT	RELCUT
RESIDBOUNDFACTOR	RESIDBOUNDFACTOR
---	ROTATION
---	RM_ESO_TREES
ROW_INDENT	---
---	SHAPEFUNCTIONS
1_D_PLACEMENT	1_D_PLACEMENT

Table 3 illustrates an example of a control file for the generation of a placement of a standard-cell circuit. Table 4 illustrates an example of a control file for the generation of a placement of a sea-of-gates.

Table 2: Necessary control words

DESIGN_FILE
DESIGN_FORMAT
DESIGN_STYLE
TARGET_SYSTEM
END

Table 3: Example of a standard-cell control file:

`DESIGN_FILE = stddes.block1;`	The circuit description file is `stddes.block1`.
`DESIGN_FORMAT = PROUD;`	
`DESIGN_STYLE = STANDARDCELL;`	
`TARGET_SYSTEM = VENUS;`	Only for standard-cell designs.
`MINCUTPART = 1.0;`	All modules can be exchanged by Min-Cut to find the optimal cut (a value between 0.5 and 1.0 is recommended).
`DYNNET;`	A linear objective function is minimized by dynamic net weight updates
`END;`	End of the control file.

Table 4: Example of a sea-of-gates control file:

`DESIGN_STYLE = SEA_OF_GATES;`	A sea-of-gates circuit.
`DESIGN_FILE = design.block;`	The circuit description file is `design.block`.
`DESIGN_FORMAT = PROUD;`	Design description is read from a PROUD file.
`TARGET_SYSTEM = MARS;`	The circuit shall be processed for a MARS master.
`SHAPEFUNCTIONS = 7;`	Parameter for the surface utilization. Slicing structures up to 7 modules will be generated and rated with the help of form functions.
`RM_ESO_TREES;`	Conserve memory.
`MINCUTPART = 1.0;`	All modules should be allowed to be exchanged by Min-Cut to find the optimal cut (a value between 0.5 and 1.0 is recommended).
`CUTDIR = VERTICAL;`	It will begin with a vertical cut.
`RESIDBOUNDFACTOR = 1e-4;`	The stopping criterion for the norm of the residue.
`MAXPINSPERNET = 30;`	Nets with over 30 pins will be eliminated.
`END;`	End of the control file.

Table 5: Example of a "minimal" standard-cell control file:

```
DESIGN_STYLE = STANDARDCELL;
DESIGN_FILE = stddes.block1;
DESIGN_FORMAT = PROUD;
TARGET_SYSTEM = SPACE;
END;
```

Table 6: Example of a “minimal” macrocell like sea-of-gates control file:

```
DESIGN_STYLE = SEA_OF_GATES;
DESIGN_FILE= d.circ;
DSIGN_FORMAT = PROUD;
TARGET_SYSTEM = SPACE;
SHAPEFUNCTIONS = 4;
END;
```

Table 7: Example of a “minimal” GordianL standard-cell control file:

```
DESIGN_STYLE = STANDARDCELL;
DESIGN_FILE= stddes.block1;
DESIGN_FORMAT = PROUD;
TARGET_SYSTEM = VENUS;
DYNNET;
END;
```

Table 8: Example of a GordianL standard-cell control file if GordianL is to be followed by Domino:

```
DESIGN_STYLE = STANDARDCELL;
DESIGN_FILE= stddes.block1;
DESIGN_FORMAT = PROUD;
TARGET_SYSTEM = DOMINO;
DYNNET;
END;
```

A The Proud input format

This is a short description of the Proud format used in GORDIAN. It is similar to the input format used in the Proud program of Ren-Song Tsay developed at UC Berkeley [TK88]. No comments or remarks are allowed in the cell library file and in the design file.

A.1 Cell Library File (celllib.*circuit_name*)

The cell library file describes the geometrical structure of each cell in detail.

In the file, the first line gives the number of logical cells, `nrLogCells`, i.e. the number of cells described in the file.

```
Example:
    ** nrLogCells **
    4
```

Following this line, logical cells are listed sequentially as the order of logical cell id. Three subsections are used to describe each logical cell: head line, pin list, and physical cell description.

A.1.1 Head line

It contains four fields on the same line.

(1) `logCellId`: logical cell id numbered sequentially from 0 to `nrLogCells`-1.

(2) `logCellName`: logical cell name, a string with less than eight characters excluding spaces and tabs.

(3) `nrPhyCells`: number of physical cells. Currently it can be only 1.

(4) `nrPins`: the number of pins that are going to be listed.

```
Example:
    ** logCellId logCellName nrPhyCells nrPins **
    0 NAND 1 3
```

A.1.2 Pin List

Repeated all on the same line alternately with `pinId` and `pinName`. The `pinId` has to be sequentially from 0 to `nrPins` -1.

```
Example:
    ** pinId pinName pinId pinName ... **
    0 in0 1 in1 2 out
```

A.1.3 Physical Cell Description

If there are more than one physical implementations, the physical cells are described sequentially from 0 to `nrPhyCells`-1. Currently only one physical cell is allowed.

(1) head line:

  (a) `phyCellId`: It can only be 0.
  (b) `phyCellName`: physical cell name, a string with less than eight characters excluding spaces and tabs.
  (c) `nrCorners`: Must be 4 corners since Gordian can handle only rectangular cells.
  (d) `startMos`: NOT USED, set to 0.
  (e) `nrStartTrks`: NOT USED, set to 1.
  (f) `nrM1Obs`: NOT USED, set to 0.
  (g) `nrM2Obs`: NOT USED, set to 0.
  (h) `tMask`: NOT USED.

```
Example:
    ** phyCellId phyCellName nrCorners startMos nrStartTrks nrM1Obs
                                                     nrM2Obs tMask **
    0 NAND0 4 0 1 0 0 15
```

(2) Corner Coordinates: integer x and y coordinates of each corner in counterclockwise order. All in one line.

```
Example:
    ** cor1.x cor1.y cor2.x cor2.y ... **
    0 0 630 0 630 156 0 156
```

(3) `startTracks`: NOT USED, set it to 0.

```
Example:
    ** startTrack1 startTrack2 ... **
    0
```

(4) Pin lists:

  (a) `pinId`: pin id number from 0 to `nrPins`-1.
  (b) `pinName`: a string that must be consistent with the corresponding logical pin name.
  (c) `llx, lly, urx, ury`: we assume each pin is a rectangle. Hence these four integers give the lower left x coordinate, lower left y coordinate, upper right x coordinate, upper right y coordinate of the rectangle relative to the lower left corner of the physical cell.
  (d) `layer`: the layer that the pin can be accessed, usually NOT USED. If layer is 1024 it is assumed that this pin is a feedthrough pin.

```
Example:
    ** pinId pinName llx lly urx ury layer **
    0 in  56  156 56  156 4
    1 out 196 156 196 156 4
    2 feed 300 0 300 0 1024
    3 feed 300 156 300 156 1024
```

A.2 Design File design.*circuit_name*

Basically, this file describes the net list. Additionally for MARS sea-of-gates the underlying base cell image and chip size are defined. For standardcells this file contains the number of rows.

The main objects to be described are modules (also called instances), which are used for net list specification. There physical description can be taken from the cell library.

I/O pads are modules already placed around chip boundary.

The design file contains five parts: physical description of chip, cell library pointer, design statistics, I/O pad description, and module description.

A.2.1 Physical Description of the Chip

Sea-of-Gates

(1) `nrCols`: number of base cell columns (integer).

```
Example:
    ** nrCols **
    nrCols=    42
```

(2) `nrRows`: number of base cell rows (integer).

```
Example:
    ** nrRows **
    nrRows=    20
```

(3) `bcX, bxY`: integer base cell dimension in x and y direction, in terms of your internal unit. Most commonly used units are one micron or half micron.

```
Example:
    bcX=    5
    bcY=    13
```

(4) `xGridWidth, yGridWidth`: Assumed to be 1.

```
Example:
    xGridWidth=    1
    yGridWidth=    1
```

As the numbers given in the previous examples, you can figure out the chip configuration. The width of the chip is `nrCols * bcX`, i.e. 42 * 5 = 210. The height of the chip is `nrRows * bcY`, i.e. 20 * 13 = 260.

Standardcells

(1) nrofrows: the number of standardcell rows. If zero is specified Gordian tries to calculate an appropriate number.

```
Example:
    nrofrows= 0
```

A.2.2 Cell Library Pointer

This field, `cellLibFile`, provides the name of the cell library file.

```
Example:
    cellLibFile:    celllib.example
```

A.2.3 Design Statistics

Design statistics about number of I/O pads, modules, nets and pins are given to help the program read this file fast.

(1) `nrIos`: number of I/O pads. It should include dummy I/O pads.

```
Example:
    nrIOs=    16
```

(2) `nrMods`: number of modules.

```
Example:
    nrMods=    16
```

(3) `nrNets`: number of signal nets.

```
Example:
    nrNets=    20
```

(4) `nrPins`: number of pins of both I/O pads and modules listed in this design file.

```
Example:
    nrPins=    51
```

A.2.4 I/O Pad Description

Each I/O pad is described in two sub-sections: head line, pin list.

(1) Head line: eight fields on the same line, separated by spaces.

  (a) `ioId`: starts from 0 to `nrIOs`−1. The id must be in the same order as the I/O pad listed in the design file.

  (b) `ioName`: any string.

  (c) `logCellId`: the corresponding cell id in the cell library that this I/O pad is implemented.

  (d) `phyCellId`: NOT USED. This field is always 0.

  (e) `nrPins`: the number of pins that will be listed in next sub-section. These pins are used in connecting nets.

(f,g) `x, y`: the coordinates of lower left corner of this I/O pad relative to the lower left corner of the chip core area.
When using the sea of gates layout style the position of the I/O pads must be given in x,y coordinates.
`side, pos`: for standardcells without predefined chip dimensions side and position relative to the other pads on the same side can be specified. Position is counted from left to right and from bottom to top.
side means:

1	bottom
2	left
3	top
4	right

(h) `pattern`: Rotation pattern.
pattern means:

0	no transformation
1	no transformation
2	flip along x-axis (map ”⊑” shape to ”⊒”)
4	flip along y-axis
8	rotation of 180 degrees
10	mirror along y-axis and then rotation of 90 degrees
20	rotation of 270 degrees
40	rotation of 90 degrees
80	mirror along x-axis and then rotation of 90 degrees

```
Example:
    ** ioId ioName logCellId phyCellId nrPins x y pattern **
    5 io5 0 0 1 80 27 0
```

(2) Pin list: every pin occupies one line with four fields.

(a) `pinId`: a sequential number of this pin from 0 to `nrPins`−1. The ordering of pins is not important.

(b) `logPinId`: the id number of the corresponding pin on the designated logical cell.

(c) `netId`: the id number of the signal net that this pin belongs to. The netId runs from 1 to nrNets.

(d) `pinType`: NOT USED.

```
Example:
    ** pinId logPinId netId pinType **
    5 0 19 0
```

A.2.5 Module Description

The format in this section is exactly the same as that for I/O pad except `ioId` and `ioName` are now `modId` and `modName`, and the `modId` should be numbered sequentially from `nrIOs` to `nrMods+nrIOs`−1.

```
Example:
    ** modId modName logCellId phyCellId nrPins x y pattern **
    17 M0 3 0 2 20 35 0
```

```
** pinId logPinId netId pinType **
5 0 1 0
6 1 2 0
```

A.3 An example

The Figure 2 shows a very small example. The corresponding `design` file and `celllib` file are given in Figure 1.

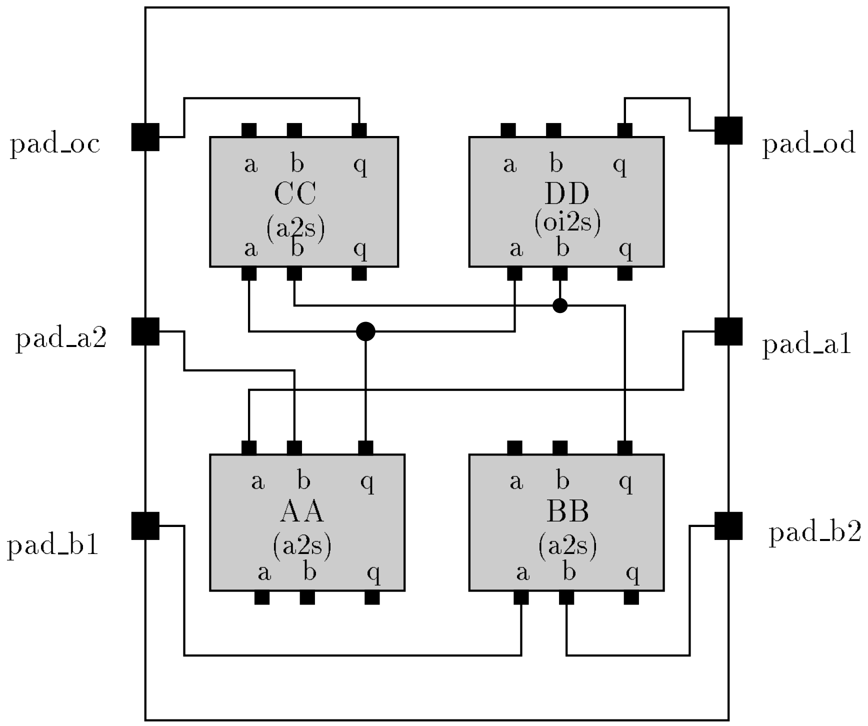
```
nrofrows= 2
cellLibFile: celllib.ex
nrIOs= 6
nrMods= 4
nrNets= 8
nrPins= 18
0 pad_b1 2 0 1 2 1 0
0 0 1 0
1 pad_b2 2 0 1 4 1 0
1 0 2 0
2 pad_oc 2 0 1 2 3 0
2 0 3 0
3 pad_a1 2 0 1 4 3 0
3 0 4 0
4 pad_a2 2 0 1 2 2 0
4 0 5 0
5 pad_od 2 0 1 4 2 0
5 0 6 0
6 AA 0 0 3 0 0 0
6 0 4 0
7 2 5 0
8 4 7 0
7 BB 0 0 3 0 0 0
9 0 1 0
10 2 2 0
11 4 8 0
8 CC 0 0 3 0 0 0
12 0 7 0
13 2 8 0
14 4 3 0
9 DD 1 0 3 0 0 0
15 0 7 0
16 2 8 0
17 4 6 0
```

Design file: `design.ex`

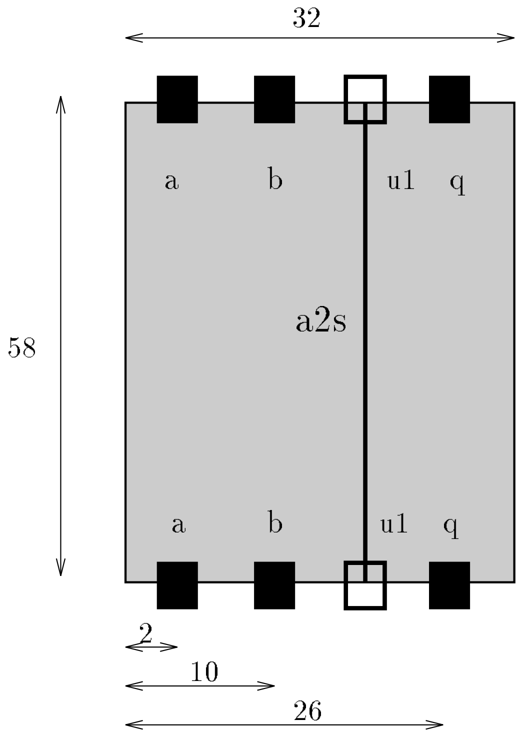
```
3
0 a2s 1 8
0 a 1 a 2 b 3 b 4 q 5 q 6 u1 7 u1
0 P0 4 0 1 0 0 15
0 0 32 0 32 58 0 58
0
0 a 2 58 5 58 4
1 a 2 0 5 0 4
2 b 10 0 13 0 4
3 b 10 58 13 58 4
4 q 26 0 29 0 4
5 q 26 58 29 58 4
6 u1 18 0 21 0 1024
7 u1 18 58 21 58 1024
1 oi2s 1 6
0 a 1 a 2 b 3 b 4 q 5 q
0 P0 4 0 1 0 0 15
0 0 24 0 24 58 0 58
0
0 a 18 58 21 58 4
1 a 18 0 21 0 4
2 b 2 58 5 58 4
3 b 2 0 5 0 4
4 q 10 58 13 58 4
5 q 10 0 13 0 4
2 pad 1 1
0 p0
0 P0 4 0 1 0 0 15
0 0 4 0 4 4 0 4
0
0 p0 2 0 2 0 4
```

Celllib file: `celllib.ex`

Figure 1: Example design and celllib file



Design



Library cell a2s

Figure 2: Example design with 4 modules, 6 IOs and 8 nets and library cell a2s

B Conversion from VPNR into Proud-Format

To facilitate the use of GORDIAN a converter from VPNR-format into Proud-format is provided with the package. This converter allows all MCNC-benchmark circuits to be converted, such that placements can be computed with GORDIAN, however it can be installed only if either the MCNC-benchmark distribution or the OASIS system are available.

The program call of the converter is as follows:

`vp2pr` *[-options] vpnr_file*

vpnr_file is the name of the VPNR file to be converted. The following options are supported:

- `-v` — Run in verbose mode printing some characteristics of the converted circuit.
- `-r` *#rows* — *#rows* specifies the number of standard cell rows.
- `-d` *cell_library* — Here the name of a library file that describes the cells in the VPNR-format can be specified. Alternatively the cell description may be included in *vpnr_file* in front of the netlist description.

The converter generates two files `design.`*vpnr_file-prefix* `celllib.`*vpnr_file-prefix* with the netlist and the cell library in Proud-format.

WARNING: To obtain the filenames correctly no point should be in the *vpnr_file-prefix*, i.e. file.vpnr and not file.xy.vpnr.

C Conversion from GORDIAN into VPNR-Format

If the GORDIAN results should be converted back in VPNR-Format call:

`go2vp` *[-v] -p* `place`.controlfile *vpnr_file*

This program writes a new VPNR-file similar to the old *vpnr_file* to the standard output, containing a placement taken from the `place.`*controlfile*-file (see Section 7) generated by GORDIAN. The option `-v` is ignored, which is useful to integrate GORDIAN in the OASIS-system.

D GORDIAN used within OASIS

The script `gordian_vpnr` can be used to execute all steps necessary to create a *vpnr_file* containing the placement result obtained with GORDIAN. The call is:

`gordian_vpnr` *[-r -L -P -c -o]*

The following options can be specified:

- `-r` *#rows* — *#rows* specifies the number of standard cell rows. If this is not specified, GORDIAN calculates an appropropriate number.
- `-L` *cell_library* — Here the name of a library file that describes the cells in the VPNR-format can be specified. The default is `db.new`.

-P *placeprogram* — The program *placeprogram* is called for placement.

-c *controlfile* — The name of a special controlfile can be specified. This controlfile must be copied into the directory `gordian/input/` by the user.

-o *output_vpnr* — The name of the vpnr file containing the placement result.
The default is `tmp`.

To use this script, the programs `vp2pr` and tt go2vp must be installed. The call requires a vpnr file describing the netlist which is usually the file *circuit_name*.`placer` and a cell library file (e.g. *db.new*) in the current directory. The script then generates a directory `gordian` and all subdirectories required to run gordian. Furthermore it copies a minimal controlfile from a template directory in the subdirectory input. The GORDIAN-output on the screen is written into a file `place.log`.

References

[KSJ88] Jürgen M. Kleinhans, Georg Sigl, and Frank M. Johannes. GORDIAN: A new global optimization / rectangle dissection method for cell placement. In *IEEE International Conference on Computer-Aided Design*, pages 506–509, November 1988.

[KSJA91] Jürgen M. Kleinhans, Georg Sigl, Frank M. Johannes, and Kurt J. Antreich. GORDIAN: VLSI placement by quadratic programming and slicing optimization. *IEEE Transactions on Computer-Aided Design*, CAD-10(3):356–365, March 1991.

[oas90] OASIS manual. Technical report, Microelectronics Center of North Carolina, Research Triangle Park, North Carolina, 1990.

[SDJ91] Georg Sigl, Konrad Doll, and Frank M. Johannes. Analytical placement: A linear or a quadratic objective function? In *ACM/IEEE Design Automation Conference*, volume 28, pages 427–432, San Fransisco, 1991.

[TK88] Ren-Song Tsay and Ernest Kuh. PROUD1.0: A package for high complexity vlsi circuit placement. Technical report, Department of EECS and the Electronics Research Laboratory, University of California, Berkeley, CA 94720, 1988.