KRPAN FPGA P&R Software

Author: Marko Mlinar
marko.mlinar@campus.fri.uni-lj.si

Rev. 0.3
April 20, 2001

Revision History

Rev.	Date	Author	Description
0.1	02 feb 01	Marko Mlinar	Updated template for SW requirements
0.2	10 feb 01	Marko Mlinar	Added initial project info
0.3	17 apr 01	MM	

1 Introduction

Placement and routing software is a tool, which automatically (or with some user help) distributes given elements, so that they match certain criteria. For FPGA (Field Programmable Gate Array) this usually is limited number of FPGA resources (connections, number of programmable elements, speed of (or part of) circuit, etc). More about resources and their functionality can be found in FPGA Architecture. All these functions serve simple goal – to program user defined net-list into desired FPGA.
Since P&R is NP-complete problem, no optimal practical solution for large placement can be found, so we are forced to search for sub-optimal solution.
Java programming language was chosen, to allow full portability on several platforms. Also we conjecture that Java will become more used and supported and will have more computing potential. It is estimated Java would run 100% slower than matching C program, for this application.

P&R Model

More general model is assumed, than described in above document: given rectangular array ($X \times Y$) of K input LUTs (Look Up Tables), where each of them is capable of calculating any binary function of N inputs and one output. They are connected using routing matrix.
Due to simplicity the optimization process is divided into three parts[1]:

- **Mapping:** net-list is covered with LUTs to satisfy: number of inputs <= K
- **Placing:** such positions are assigned to LUTs that total wire-length is minimized (wire length is approximated using sum of minimal spanning trees for every net)
- **Routing:** in this step we try to find optimal Steiner trees using given free routing resources

Thus P&R goal is to find placement of LUTs and nets (wire segments), that satisfies:

1. Every element in the user-specified net-list must be mapped into specific suitable architecture resource
2. Every LUT is placed in the rectangular array
3. If LUT is placed at position (x, y), no other LUT can share same position
4. For each connection a - b in given net-list there exist wire segment placement that connects a and b in desired direction
5. User specified constraints are met (e.g. path delay, clock speed…)

Note: Due to practical reasons (and algorithm goal) some assumptions were made that could prohibit P&R software to reach optimal solution.

[1] Authors are aware that (for this architecture) solution can be found in single step.

2 Description

This section specifies program, its functionality and usage information in detail. Also program arguments and input/output file format are explained here.

2.1 Program Usage

Program KRPAN is a command line tool. Valid parameters must be supplied as specified below:

2.1.1 Help Option ***-h***

Displays following help screen.

```
Program KRPAN v0.2, FPGA Mapping&Placement&Routing utility.
(C) 2001 Marko Mlinar.
KRPAN comes with ABSOLUTELY NO WARRANTY; see license.txt for details.

        Usage: KRPAN input_file [output_file] [-option [-option ...]]

 Valid options:
 -h -? --help     this screen
 -v --verbose     verbose output to log file
 -d --debug               verbose debug output to log file
 -g --graphics    display progress graphically
 -sd --sdir {dir} sets source dir (default ".")
 -dd --ddir {dir} sets destination dir (default ".")
 -l  --log {file} sets different log file (default "KRPAN.log")
 -b
 -c --clear       include unused cells in configuration

Usage example:
KRPAN -sd examples -dd . -g test.v
(generates bit stream specification file named 'test.v.bin', if successful)
```

2.1.2 Verbose Option ***-v***

Forces program to display current progress details.

2.1.3 Debug Option ***-d***

Similar to verbose option, but stronger. Program internal data and structures are displayed. Most important of internal structures is Graph.toString() print-out:

```
Graph: ffs.mapped.preplaced before annealing
Number of nodes: 10

IOC Node: rst(x-1 y21 temp2) nets(2): GND, *rst
IOC Node: clk(x24 y12 temp2) nets(2): GND, *clk
IOC Node: a(x11 y-1 temp2) nets(2): GND, *a
IOC Node: b(x17 y24 temp2) nets(2): GND, *b
IOC Node: q(x-1 y15 temp2) nets(2): q, *null
GPC Node: (y_reg_LUT,y_reg)(x15 y16 temp9) nets(9): b, GND, GND, GND, GND, GCLK0, GND, GRST, *y
GPC Node: (x_reg_LUT,x_reg)(x17 y21 temp9) nets(9): a, GND, GND, GND, GND, GCLK0, GSET, GND, *x
GPC Node: (a14_1,q_reg)(x13 y18 temp9) nets(9): a, GND, GND, GND, GND, GCLK1, n130, GND, *q
GPC Node: (a13_0,null)(x19 y23 temp9) nets(9): y, GND, GND, GND, GND, GND, GND, GND, *n93
GPC Node: (U33,null)(x19 y21 temp9) nets(9): x, GND, GND, GND, GND, GND, GND, GND, *n130

GND Net GND, nodes(34) *null, rst, clk, a, b, (y_reg_LUT,y_reg), (y_reg_LUT,y_reg), (y_reg_LUT,y_reg), (y_reg_LUT,y_reg),
(y_reg_LUT,y_reg), (x_reg_LUT,x_reg), (x_reg_LUT,x_reg), (x_reg_LUT,x_reg), (x_reg_LUT,x_reg), (x_reg_LUT,x_reg),
(a14_1,q_reg), (a14_1,q_reg), (a14_1,q_reg), (a14_1,q_reg), (a14_1,q_reg), (a13_0,null), (a13_0,null), (a13_0,null),
(a13_0,null), (a13_0,null), (a13_0,null), (a13_0,null), (U33,null), (U33,null), (U33,null), (U33,null), (U33,null),
(U33,null), (U33,null)
VCC Net VCC, nodes(1) *null
GCLK0 Net GCLK0, nodes(3) *SR1, (y_reg_LUT,y_reg), (x_reg_LUT,x_reg)
GCLK1 Net GCLK1, nodes(2) *SR3, (a14_1,q_reg)
GRST Net GRST, nodes(2) *SR5, (y_reg_LUT,y_reg)
GSET Net GSET, nodes(2) *SR4, (x_reg_LUT,x_reg)
PIO_CLK Net PIO_CLK, nodes(1) *null
Nets(9):
Net rst, nodes(3) *rst, SR5, SR4
Net clk, nodes(2) *clk, SR1
Net a, nodes(3) *a, (x_reg_LUT,x_reg), (a14_1,q_reg)
Net b, nodes(2) *b, (y_reg_LUT,y_reg)
Net x, nodes(2) *(x_reg_LUT,x_reg), (U33,null)
Net q, nodes(2) *(a14_1,q_reg), q
Net y, nodes(2) *(y_reg_LUT,y_reg), (a13_0,null)
Net n93, nodes(2) *(a13_0,null), SR3
Net n130, nodes(2) *(U33,null), (a14_1,q_reg)
```

First nodes are listed, each having type, name, some parameters including nets node is connected to. Net, that node is writing to (output net) is marked with star *.
After list of nodes list of all global and all local nets is supplied. Each net description starts with net name, total number of nodes it is connected to and list of them. Net source node is marked with * and is listed first.
There are various node types, as shown in Table 1 that can be printed or displayed, depending of state P&R tool is in. If node is not to be displayed, it is displayed in gray color - meaning some internal error has occurred.

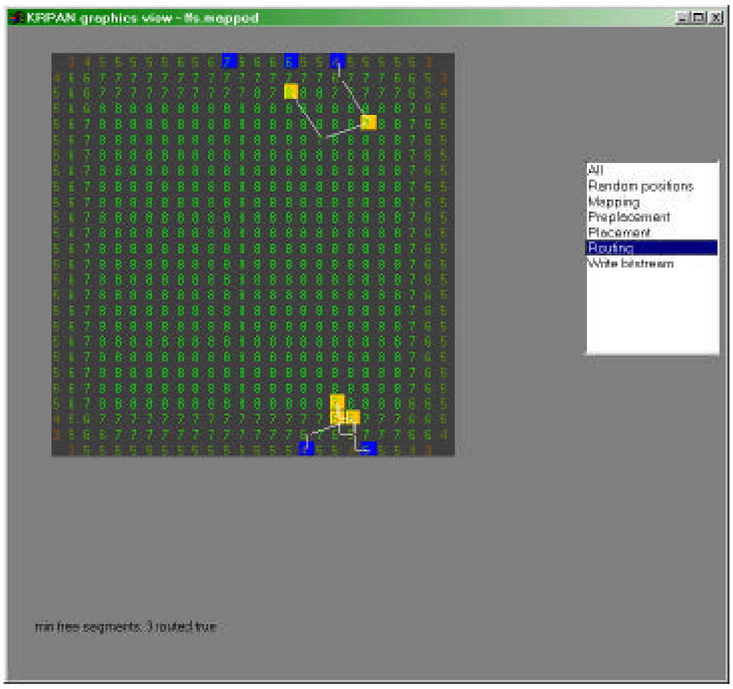
Table 1: Node types

Log name and color displayed	Matching Class	Description
🟦 PORT*x*	NodePort	Input(0) or output(1) module port
🟪 PRIM	NodePrim	Primitive function (e.g. AND2)
🟨 FF	NodeFF	D type flip-flop
🟥 LUT*x func*	NodeLUT	*x* input Look-Up-Table with function *func*
🟦 IOC	NodeIOC	IOC as defined in architecture specification
🟧 GPC	NodeGPC	GPC as defined in architecture specification
⬜ SR	NodeSR	Special resource node - not displayed

2.1.4 Graphics Option ***-g***

Displays progress graphically, using Java AWT. Graphical mode allows user to manually place (using first mouse button) nodes and get overview on circuit and possible problems. Figure 1 shows sample graphical display after routing. Numbers represent number of free routing channels per node. "Matches" represents connections. Bottom label displays current status and object info currently under mouse. Double click on right list selects action SW should perform.

Figure 1: Graphical display of routed net-list.



min free segments: 3 routed true

2.1.5 Log Option ***-l***

Specifies different log file name; default KRPAN.log.

*2.1.6 Source and Destination Directory Option (**-sd** and **-dd**)*

Specifies source or destination directory. Source directory specifies relative to where main source file is, and where libraries and dependent files are located.

2.1.7 Include Empty Cells Option ***-c***

If specified, cells that are not assigned are included in configuration bit-stream as empty.

2.2 Input File

Various formats are supported by KRPAN. File format is specified with its extension:

- EDIF [*.edf]
- Verilog - HDL [*.v]

WARNING: KRPAN does not support complete language grammars, for languages specified above. Only technology independent data should be supplied.

2.2.1 Limitations of EDIF parser

Only technology independent data is extracted from EDIF file, all other data is `ignored`.

2.2.2 Limitations of VERILOG parser

This version only supports technology independent net lists for GTECH library. Exactly one module should be in each file. Other limitations are:

- no memory support (e.g.: `reg [31:0] mem [1023:0]`)
- no special wire support - only `wire`, `input` and `output` are valid
- no meta data info such as #

2.2.3 Specifying Special Signal

Some port names are reserved and are used for accessing special resources:

- `PIO_CLK` accesses the GPIO clock
- `WB_CLK` accesses the WISHBONE clock, which is driven to FPGA WB interface

They should be defined as external ports, e.g.:

```
module neg(PIO_CLK, out_clk);
input PIO_CLK;
output out_clk;

  assign out_clk = ~PIO_CLK;
endmodule;
```

2.3 Log File

Specifies stream where should be debug and verbose output written. Default value is `KRPAN.log`.

2.4 Output File

Default output file name is input file name, with added `.bin` extension. It holds bit stream data of specified design and it is functionally equal to input net-list. See bit stream appendix in FPGA architecture document.

2.4 Example of Use

Command line:

java –jar KRPAN.jar tst.edf

tst.edf file

```
(edif TST
  (edifversion 2 0 0 )
  (ediflevel 0 )
  (keywordmap (keywordlevel 0 ))
  (library MCNC
    (ediflevel 0)
    (technology (numberdefinition))
      (cell INV (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port IN
              (direction INPUT))
            (port OUT
              (direction OUTPUT)))))
      (cell AND2 (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port I0
              (direction INPUT))
            (port I1
              (direction INPUT))
            (port OUT
              (direction OUTPUT)))))
      (cell OR2 (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port I0
              (direction INPUT))
            (port I1
              (direction INPUT))
            (port OUT
              (direction OUTPUT)))))
      (cell NAND2 (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port I0
              (direction INPUT))
            (port I1
              (direction INPUT))
            (port OUT
              (direction OUTPUT)))))
      (cell NOR2 (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port I0
              (direction INPUT))
            (port I1
              (direction INPUT))
            (port OUT
              (direction OUTPUT))))))
  (library EDIF_TEST
    (ediflevel 0)
    (technology (numberdefinition))
      (cell TST (celltype generic)
        (view nl (viewtype net-list)
          (interface
            (port a
              (direction INPUT))
            (port b
              (direction INPUT))
            (port c
              (direction INPUT))
            (port d
              (direction INPUT))
            (port e
              (direction INPUT))
            (port f
              (direction INPUT))
            (port p
              (direction OUTPUT))
            (port q
              (direction OUTPUT)))
          (contents
              (instance I1 (viewref nl (cellref NAND2
(libraryref MCNC))))
            (instance  I2  (viewref  nl  (cellref  NOR2
(libraryref MCNC))))
              (instance I3 (viewref nl (cellref NAND2
(libraryref MCNC))))
            (instance  I4  (viewref  nl  (cellref  OR2
(libraryref MCNC))))
            (instance  I5  (viewref  nl  (cellref  AND2
(libraryref MCNC))))
              (instance I6 (viewref nl (cellref NAND2
(libraryref MCNC))))
            (instance  I7  (viewref  nl  (cellref NOR2
(libraryref MCNC))))
            (instance I8 (viewref nl (cellref AND2 (libraryref MCNC))))
            (instance I9 (viewref nl (cellref OR2 (libraryref MCNC))))
            (instance I10 (viewref nl (cellref AND2 (libraryref MCNC))))
            (instance I11 (viewref nl (cellref NAND2 (libraryref MCNC))))
            (instance I12 (viewref nl (cellref OR2 (libraryref MCNC))))
            (net na
              (joined
                (portref a)
                (portref I0 (instanceref I8))
                (portref I0 (instanceref I5))))
            (net nb
              (joined
                (portref b)
                (portref I0 (instanceref I1))))
            (net nc
              (joined
                (portref c)
                (portref I1 (instanceref I1))
                (portref I0 (instanceref I2))))
            (net nd
              (joined
                (portref d)
                (portref I1 (instanceref I2))))
            (net ne
              (joined
                (portref e)
                (portref I0 (instanceref I3))))
            (net nf
              (joined
                (portref f)
                (portref I1 (instanceref I3))))
            (net n1
              (joined
                (portref OUT (instanceref I1))
                (portref I0 (instanceref I12))
                (portref I0 (instanceref I4))))
            (net n2
              (joined
                (portref OUT (instanceref I2))
                (portref I1 (instanceref I7))
                (portref I1 (instanceref I4))))
            (net n3
              (joined
                (portref OUT (instanceref I3))
                (portref I1 (instanceref I6))
                (portref I1 (instanceref I10))))
            (net n4
              (joined
                (portref OUT (instanceref I4))
                (portref I1 (instanceref I5))
                (portref I0 (instanceref I6))))
            (net n5
              (joined
                (portref OUT (instanceref I5))
                (portref I0 (instanceref I9))))
            (net n6
              (joined
                (portref OUT (instanceref I6))
                (portref I1 (instanceref I8))
                (portref I0 (instanceref I7))))
            (net n7
              (joined
                (portref OUT (instanceref I7))
                (portref I1 (instanceref I9))
                (portref I0 (instanceref I10))))
            (net n8
              (joined
                (portref OUT (instanceref I8))
                (portref I0 (instanceref I11))))
            (net n9
              (joined
                (portref OUT (instanceref I9))
                (portref I1 (instanceref I11))))
            (net n10
              (joined
                (portref OUT (instanceref I10))
                (portref I1 (instanceref I12))))
            (net n11
              (joined
                (portref OUT (instanceref I11))
                (portref p)))
            (net n12
              (joined
                (portref OUT (instanceref I12))
                (portref q)))))))
  (design TST
    (cellref TST
      (libraryref EDIF_TEST))))
```

2.5 Constraints

(not implemented yet)

3 Modules

This section should cover program structure, internal functionality, and dependencies and describes how program should be built. See KRPAN API specification for more internal details.

3.1 Program Structure

Program is basically divided into following parts, for which we can assume that are executed sequentially:

1. parser (org.opencores.edifp and org.opencores.verilog package)
2. constraint handling (org.opencores.constraints package, *UNIMPLEMENTED*)
3. mapping (org.opencores.mapping package)
4. preplacement (org.opencores.placement package)
5. placement (org.opencores.placement package)
6. routing (org.opencores.routing package)
7. constraint testing (org.opencores.constraints package, *UNIMPLEMENTED*)
8. bitstream generation (org.opencores.structure package)

In following sections above parts are explained in detail.

3.1.1 Parser

Two manually built LL(1) parsers for net-list loading. Both use large hash-type dictionary for identifier names, which shortens loading time.

3.1.2 Constraint Handling

(Not yet implemented)

3.1.3 Mapping

Elements (e.g. gates) are mapped to special purpose functions and other elements, which exist in the architecture (see FPGA Architecture Specification document for more details).
Optimal mapping (for unit-delay net-list) algorithm, called FlowMap, was chosen because its speed and good results.

3.1.4 Pre-placement

The goal of this part is to find valid position for every resource, so there are no conflicts. Only valid placement is found, not optimal.

3.1.5 Placement

After we have valid placement, we search for better placement solution, which allows us to minimize delay and to improve routing resources utilization. Simulated annealing is used for searching for near-optimal solution.

3.1.6 Routing

This part assigns valid positions to metal segments and connects them in a way that design functionality stays the same with minimal possible delays.

3.1.7 Constraint Handling

(Not yet implemented)

3.1.8 Bitstream Generation

Actually generated using `writeBitstream` method and it is not in special package.

3.2 Building and Running KRPAN

KRPAN is Sun Java compatible. It requires at least Sun Java 2 SDK v1.2. It is available for various platforms. There are many other Java compilers and interpreters, which can be used, but only if they are fully compatible with Sun's. Some of them also allow binary file generation, but this is platform specific. In this section building with `javac` is supplied.
First decompress jar files, which contains sources, documents and examples:

```
jar xf sources.jar
jar xf examples.jar
jar xf docs.jar
mkdir class
```

When decompressing jar packages you should get following directory structure:

```
examples
    edif
      v
source
      org
          opencores
                  edifp
                  graphics
                  JLex
                  mapping
                placement
                routing
                structure
                util
                verilogp
doc
    index-files
      org
         …
class
```

Then compile sources using:
```
javac -O -d class -g:none source/org/opencores/*.java
source/org/opencores/edifp/*.java source/org/opencores/graphics/*.java
source/org/opencores/JLex/*.java source/org/opencores/mapping/*.java
source/org/opencores/placement/*.java source/org/opencores/routing/*.java
source/org/opencores/structure/*.java source/org/opencores/util/*.java
source/org/opencores/verilogp/*.java
```

You should get a whole directory tree of classes into `class` directory. Now run:

```
java -cp class org.opencores.Main [KRPAN parameters]
```

Your KRPAN should be working now. Please refer to your `jar` utility documentation if you want to build `.jar` file. OpenCores signed jar package (named `KRPAN.jar`) is also available for download at `www.opencores.org`.

5 Performance

This section should discuss the performance (e.g. speed, quality) and do some comparisons.