**Toro – a vpr front-end**

**User REference Guide**

Jeff Rudolph

[jrudolph@ti.com](mailto:jrudolph@ti.com)

##### Table of Contents

[Introduction 1](#_Toc341697975)

[Running Toro 2](#_Toc341697976)

[Input Files 2](#_Toc341697977)

[Options File Syntax 3](#_Toc341697978)

[Architecture File Syntax 6](#_Toc341697979)

[Fabric File Syntax 7](#_Toc341697980)

[Circuit File Syntax 8](#_Toc341697981)

[Output Files 10](#_Toc341697982)

[Use Models 10](#_Toc341697983)

[Appendix: Building Toro 11](#_Toc341697984)

[Appendix: Preprocessor Support 12](#_Toc341697985)

[Appendix: Regular Expression Support 13](#_Toc341697986)

[Appendix: References 13](#_Toc341697987)

[Revision History 14](#_Toc341697988)

# Introduction

Toro is front-end to the FPGA VPR (Versatile Place and Route) tool. Toro is an open source EDA tool developed by TI. VPR is a research tool from the University of Toronto.

For introductory background on FPGA architectures and design tools, see **Appendix: References** [0] and [1].

For more information on VPR, see **Appendix: References** [2], [3], and [4].

VPR is currently part of the VTR (Verilog-to-Routing) project - a collaboration between several university research groups to provide a set of open-source tools for FPGA research.

For further information on VTR, see: <http://code.google.com/p/vtr-verilog-to-routing/>

The Toro front-end was developed by TI to enable deploying VPR’s functionality in a production environment, given that VPR’s primarily focus is on academic research. Towards this end, Toro was designed to address key deployment areas such as:

* usability
* supportability
* reliability
* extendibility

The following context diagram illustrates the overall Toro front-end relationship with VPR.

The following diagram illustrates a graphical view of a floorplan file that is suitable for placement.

**IOs**

**CLBs**

hard block

RAM

hard block

LUTs

DSP

LUTs

LUTs

LUTs

LUTs

RAM

LUTs

DSP

LUTs

LUTs

LUTs

LUTs

RAM

LUTs

DSP

LUTs

LUTs

LUTs

LUTs

RAM

LUTs

DSP

LUTs

LUTs

LUTs

LUTs

Notes:

* The IO tiles are intentionally drawn to demonstrate support for a variable number of IOs per side.
* The IO tiles are intentionally drawn to highlight non-alignment with rows or columns.

For global routing, the architecture description must include additional information needed to generate a floorplan file that represents global routing channels.

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

**global**

**tracks**

For detailed routing, the architecture description must include all information needed to generate a floorplan file that represents detailed routing tracks.

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

LUTs

**connection**

**block**

**switch**

**block**

**route**

**tracks**

# Running Toro

Toro supports the following command line arguments:

Usage: toro

-h[elp] Shows program help

-v[ersion] Shows program build version

-o[ptions] <file(s)> Input options file name(s)

-x[ml] *file* Input architecture file (VPR's XML)

-b[lif] *file* Input circuit file (VPR's BLIF)

-a[rchitecture] *file* Input architecture file

-f[abric] *file* Input fabric file

-c[ircuit] *file* Input circuit file

+a[rchitecture] *file* Output architecture file

+f[abric] *file* Output fabric file

+c[ircuit] *file* Output circuit file

+l[og] *file* Output log file name

-e[xecute] ( pack + place + route ) | all | none

In addition to the Toro-specific command line arguments, Toro also supports the full range of VPR (version 6.0) arguments in a pass-thru mode.

VPR’s native command line arguments are documented in the "**VPR User's Manual**". See **Appendix: References** [5]

# Input Files

Toro accepts the following input files:

| **Input File** | **Description** |
| --- | --- |
| options | Defines various program input, output, message, and run-time options.  The Toro options file provides a superset of the program arguments available from the Toro and VPR command lines.  Multiple input options files are supported. In the event of conflicting options, the most recent definition is used. |
| architecture | Defines a FPGA architecture description.  The Toro architecture file provides a superset of VPR’s XML file. For example, the architecture file supports optional block size and origin specifications.  In addition, the Toro architecture file format offers isolation and enhanced stability with respect to VPR’s XML file format changes. |
| fabric | Defines a FPGA fabric model.  The Toro fabric file specification includes blocks, switchboxes, channels, and segments. |
| circuit | Defines a FPGA circuit design.  The Toro circuit file provides a superset of VPR’s BLIF file. For example, the circuit file supports pre-packed, pre-placed, and/or pre-routed specifications. |
| XML | Defines a FPGA architecture description in XML format  This VPR-native format is a functional subset of Toro’s architecture file.  For more information on VPR's XML file, see **Appendix: References** [5]. |
| BLIF | Defines a FPGA circuit design in BLIF format.  This VPR-native format is a functional subset of Toro’s circuit file.  A BLIF file is typically generated by ABC. ABC is a Verilog optimization and technology mapping tool developed by the Berkeley Logic Synthesis and Verification Group.  For more information on VPR's BLIF file, see **Appendix: References** [5]. |

## Options File Syntax

The following syntax rules apply:

* All keywords are case-insensitive.
* Standard whitespace characters are space, tab, linefeed, and carriage-return.
* All **boldface** options indicate default values.
* All *italicized* options indicate user-defined values.
* All string values must be quoted.
* All numeric values are in microns, unless otherwise noted.
* Parentheses ‘**()**’ notation indicates a list of options.
* Plus ‘**+**’ notation indicates multiple selection from a list of options.
* Or ‘**|**’ notation indicates single selection from a list of options.
* Square bracket ‘**[]**’ notation indicates optional data.
* Empty ‘**nil**’ notation indicates an undefined (i.e. no default) value.
* Infinity ‘**inf**’ notation indicates an unlimited value.
* Supported comment formats include Perl, C, C++, and XML styles.

The options file includes support for standard macro preprocessor directives. For more information, see **Appendix: Preprocessor Support**.

// ----------------------------------------------------------------- //

// Input options //

// ----------------------------------------------------------------- //

INPUT\_FILE\_XML [=] *file* | **"\*.toro.xml"**

INPUT\_FILE\_BLIF [=] *file* | **"\*.toro.blif"**

INPUT\_FILE\_ARCH[ITECTURE] [=] *file* | **"\*.toro.arch"**

INPUT\_FILE\_FABRIC [=] *file* | **"\*.toro.fabric"**

INPUT\_FILE\_CIRCUIT [=] *file* | **"\*.toro.circuit"**

INPUT\_ENABLE\_XML [=] ON | **OFF** ( TRUE | FALSE )

INPUT\_ENABLE\_BLIF [=] ON | **OFF** ( TRUE | FALSE )

INPUT\_ENABLE\_ARCH[ITECTURE] [=] ON | **OFF** ( TRUE | FALSE )

INPUT\_ENABLE\_FABRIC [=] ON | **OFF** ( TRUE | FALSE )

INPUT\_ENABLE\_CIRCUIT [=] ON | **OFF** ( TRUE | FALSE )

// ----------------------------------------------------------------- //

// Output options //

// ----------------------------------------------------------------- //

OUTPUT\_FILE\_LOG [=] *file* | **"\*.toro.log"**

OUTPUT\_FILE\_OPTIONS [=] *file* | **"\*.toro.options"**

OUTPUT\_FILE\_XML [=] *file* | **"\*.toro.xml"**

OUTPUT\_FILE\_BLIF [=] *file* | **"\*.toro.blif"**

OUTPUT\_FILE\_ARCH[ITECTURE] [=] *file* | **"\*.toro.arch"**

OUTPUT\_FILE\_FABRIC [=] *file* | **"\*.toro.fabric"**

OUTPUT\_FILE\_CIRCUIT [=] *file* | **"\*.toro.circuit"**

OUTPUT\_FILE\_LAFF [=] *file* | **"\*.toro.laff"**

OUTPUT\_FILE\_[RC\_]DELAYS [=] *file* | **"\*.toro.rc"**

OUTPUT\_EMAIL\_METRICS [=] *address* | **nil**

OUTPUT\_ENABLE\_LOG [=] **ON** | OFF ( TRUE | FALSE )

OUTPUT\_ENABLE\_OPTIONS [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_XML [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_BLIF [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_ARCH[ITECTURE] [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_FABRIC [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_CIRCUIT [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_LAFF [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_ENABLE\_[RC\_]DELAYS [=] ON | **OFF** ( TRUE | FALSE )

OUTPUT\_LAFF\_MODE [=] { NONE FABRIC\_VIEW BOUNDING\_BOX INTERNAL\_GRID **ANY** }

OUTPUT\_[RC\_]DELAY\_MODE [=] **ELMORE** | D2M

OUTPUT\_[RC\_]DELAY\_SORT [=] **[BY\_]NETS** | [BY\_]DELAYS

OUTPUT\_[RC\_]DELAY\_NETS [=] *name* | { *name* ... } | **nil**

OUTPUT\_[RC\_]DELAY\_(BLOCK|CLB)[S] [=] *name* | { *name* ... } | **nil**

OUTPUT\_[RC\_]DELAY\_MAX\_[WIRE\_]LENGTH [=] *microns* | **nil**

// ----------------------------------------------------------------- //

// Message options //

// ----------------------------------------------------------------- //

FORMAT\_MIN\_GRID [=] *microns* | **0.01**

FORMAT\_TIME\_STAMPS [=] ON | **OFF** ( TRUE | FALSE )

[DISPLAY\_]INFO\_ACCEPT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]INFO\_REJECT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]WARNING\_ACCEPT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]WARNING\_REJECT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]ERROR\_ACCEPT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]ERROR\_REJECT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]TRACE\_ACCEPT [=] *string* | { *string* ... } | **nil**

[DISPLAY\_]TRACE\_REJECT [=] *string* | { *string* ... } | **nil**

TRACE\_READ\_OPTIONS [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_READ\_XML [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_READ\_BLIF [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_READ\_ARCH [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_READ\_FABRIC [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_READ\_CIRCUIT [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_VPR\_SHOW\_SETUP [=] ON | **OFF** ( TRUE | FALSE )

TRACE\_VPR\_ECHO\_FILE[S] [=] ON | **OFF** ( TRUE | FALSE ) |

*echo\_file\_type [echo\_file\_name]*

// ----------------------------------------------------------------- //

// Execution options //

// ----------------------------------------------------------------- //

HALT\_MAX\_WARNING[S] [=] *count* | **inf**

HALT\_MAX\_ERROR[S] [=] count | **1**

EXECUTE\_MODE [=] ( PACK + PLACE + ROUTE ) | NONE | **ALL**

// ----------------------------------------------------------------- //

// Fabric options //

// ----------------------------------------------------------------- //

// Note: The following options apply to Toro only...

FABRIC\_BLOCK[S]\_[OVERRIDE\_]ENABLE [=] ON | **OFF** ( TRUE | FALSE )

FABRIC\_CHANNEL[S]\_[OVERRIDE\_]ENABLE [=] ON | **OFF** ( TRUE | FALSE )

FABRIC\_SWITCHBOX[ES]\_[OVERRIDE\_]ENABLE [=] ON | **OFF** ( TRUE | FALSE )

FABRIC\_C[ONNECTION]B[LOCK][S]\_[OVERRIDE\_]ENABLE

[=] ON | **OFF** ( TRUE | FALSE )

// ----------------------------------------------------------------- //

// Packer options //

// ----------------------------------------------------------------- //

// Note: See VPR User's Manual for more details...

PACK\_ALGORITHM [=] AAPACK

PACK\_AAPACK\_CLUSTER\_SEED = TIMING[\_DRIVEN] | MAX\_INPUTS

PACK\_AAPACK\_CLUSTER\_NETS = MIN\_CONNECTIONS | MAX\_CONNECTIONS

PACK\_AAPACK\_(AREA|ALPHA)\_WEIGHT = <float> | 0.75

PACK\_AAPACK\_(NETS|BETA)\_WEIGHT = <float> | 0.9

PACK\_AAPACK\_AFFINITY\_MODE = ANY | NONE

PACK\_[BLOCK|CLB]\_SIZE [=] <int>

PACK\_LUT\_SIZE [=] <int> | 4

PACK\_COST\_MODE [=] ROUTABILITY[\_DRIVEN] | TIMING[\_DRIVEN]

// ----------------------------------------------------------------- //

// Placer options //

// ----------------------------------------------------------------- //

// Note: See VPR User's Manual for more details...

PLACE\_ALGORITHM [=] SIMULATED\_ANNEALING

PLACE\_CHANNEL\_WIDTH [=] <int> | nil

PLACE\_RANDOM\_SEED [=] <int> | 1

PLACE\_TEMP[ERATURE]\_INIT [=] <float> | 100.0

PLACE\_TEMP[ERATURE]\_INIT\_FACTOR [=] <float> | 20.0

PLACE\_TEMP[ERATURE]\_INIT\_EPSILON [=] <float> | 20.0

PLACE\_TEMP[ERATURE]\_EXIT [=] <float> | 0.01

PLACE\_TEMP[ERATURE]\_EXIT\_FACTOR [=] <float> | 0.005

PLACE\_TEMP[ERATURE]\_EXIT\_EPSILON [=] <float> | 0.005

PLACE\_TEMP[ERATURE]\_REDUCE [=] <float> | 0.8

PLACE\_TEMP[ERATURE]\_REDUCE\_FACTOR [=] <float> | 0.44

PLACE\_TEMP[ERATURE]\_INNER\_NUM [=] <float> | 10.0

PLACE\_SEARCH\_LIMIT[\_ALPHA] [=] <float> | 0.44

PLACE\_COST\_MODE [=] ROUTABILITY[\_DRIVEN] |

[PATH\_]TIMING[\_DRIVEN] |

NET\_TIMING[\_DRIVEN]

PLACE\_TIMING\_COST\_FACTOR [=] <float> | 0.5

PLACE\_TIMING\_UPDATE\_INTERVAL <int> | 1

PLACE\_TIMING\_UPDATE\_COUNT <int> | 0

PLACE\_[TIMING\_]SLACK\_INIT\_WEIGHT [=] <float> | 1.0

PLACE\_[TIMING\_]SLACK\_FINAL\_WEIGHT [=] <float> | 8.0

// Note: The following options apply to Toro only...

PLACE\_RELATIVE\_ENABLE [=] ON | **OFF** ( TRUE | FALSE )

PLACE\_RELATIVE\_ROTATE[\_ENABLE] [=] ON | **OFF** ( TRUE | FALSE )

PLACE\_RELATIVE\_INIT[IAL\_]PLACE[\_RETRY] [=] *count* | **3**

PLACE\_RELATIVE\_INIT[IAL\_]MACRO[\_RETRY] [=] *count* | **10**

PLACE\_PRE[\_]PLACED\_ENABLE [=] ON | **OFF** ( TRUE | FALSE )

// ----------------------------------------------------------------- //

// Router options //

// ----------------------------------------------------------------- //

// Note: See VPR User's Manual for more details...

ROUTE\_ALGORITHM [=] PATHFINDER

ROUTE\_TYPE [=] GLOBAL | DETAILED

ROUTE\_WINDOW\_SIZE [=] <int> | 3

ROUTE\_CHANNEL\_WIDTH [=] <int> | nil

ROUTE\_MAX\_ITERATIONS [=] <int> | 50

ROUTE\_CONGESTION\_HISTORICAL\_FACTOR [=] <float> | 1.0

ROUTE\_CONGESTION\_INIT\_FACTOR[S] [=] <float> … | 0.5

ROUTE\_CONGESTION\_PRESENT\_[GROWTH\_]FACTOR [=] <float> | 1.3

ROUTE\_BEND\_COST\_FACTOR [=] <float> |

ROUTE\_RESOURCE\_COST\_MODE [=] DEMAND\_ONLY (routability-driven mode default) |

DELAY\_NORMALIZED (timing-driven mode default)

ROUTE\_COST\_MODE [=]BREADTH\_FIRST | DIRECTED\_SEARCH | TIMING[\_DRIVEN]

ROUTE\_TIMING\_MAX\_CRIT[ICALITY] [=] <float> | 0.99

ROUTE\_CRITMAX [=] <float> | 0.99

ROUTE\_TIMING\_ASTAR\_FACTOR [=] <float> | 1.2

ROUTE\_TIMING\_SLACK\_CRITICALITY [=] <float> | 1.0

// Note: The following options apply to Toro only...

ROUTE\_TRIM\_EMPTY\_CHANNEL[S] [=] **ON** | OFF ( TRUE | FALSE )

ROUTE\_TRIM\_OBS\_CHANNEL[S] [=] **ON** | OFF ( TRUE | FALSE )

ROUTE\_PRE[\_]ROUTED\_ENABLE [=] ON | **OFF** ( TRUE | FALSE )

ROUTE\_PRE[\_]ROUTED\_ORDER [=] **FIRST** | AUTO

// ----------------------------------------------------------------- //

## Architecture File Syntax

TBD…

## Fabric File Syntax

<!-- ======================================================================== -->

<FABRIC NAME [=] "fabric\_name">

<!-- -------------------------------------------------------------------- -->

<CONFIG>

<REGION> x1 y1 x2 y2 </REGION>

<POLYGON> x1 y1 x2 y2 </POLYGON>

</CONFIG>

<!-- -------------------------------------------------------------------- -->

<I[NPUT]O[UTPUT]

NAME [=] "fabric\_io\_name"

MASTER [=] "arch\_io\_name"

>

<ORIGIN> x y </ORIGIN>

<REGION> x1 y1 x2 y2 </REGION>

<PIN

NAME [=] "arch\_pin\_name"

TYPE [=] "in[put]" | "out[put]" | "cl[oc]k"

SIDE [=] "left" | "right" | "bottom" | "top"

OFFSET [=] offset

WIDTH [=] width

>

<C[ONNECTION]B[OX]> pattern </C[ONNECTION]B[OX]>

</PIN>

<SLICE

COUNT [=] count

CAPACITY [=] capacity

/>

<TIMING[\_ANALYSIS]

C[AP[\_IN]] [=] cap

T|DELAY [=] delay

/>

</I[NPUT]O[UTPUT]>

<!-- -------------------------------------------------------------------- -->

<P[HYSICAL]B[LOCK]

NAME [=] "fabric\_pb\_name"

MASTER [=] "arch\_pb\_name"

>

<ORIGIN> x y </ORIGIN>

<REGION> x1 y1 x2 y2 </REGION>

<PIN

NAME [=] "arch\_pin\_name"

TYPE [=] "in[put]" | "out[put]" | "cl[oc]k"

SIDE [=] "left" | "right" | "bottom" | "top"

OFFSET [=] offset

WIDTH [=] width

>

<C[ONNECTION]B[OX]> pattern </C[ONNECTION]B[OX]>

</PIN>

<SLICE

COUNT [=] count

CAPACITY [=] capacity

/>

<TIMING[\_ANALYSIS]

C[AP[\_IN]] [=] cap

T|DELAY [=] delay

/>

</P[HYSICAL]B[LOCK]>

<!-- -------------------------------------------------------------------- -->

<S[WITCH]B[OX]

NAME [=] "fabric\_sb\_name"

MASTER [=] "arch\_sb\_name"

>

<ORIGIN> x y </ORIGIN>

<REGION> x1 y1 x2 y2 </REGION>

<MAPPING>

"left" | "right" | "bottom" | "top" index ...

</MAPPING>

<TIMING[\_ANALYSIS]

R[ES] [=] res

C[AP]\_IN [=] cap

C[AP]\_OUT [=] cap

T|DELAY [=] delay

/>

</S[WITCH]B[OX]>

<!-- -------------------------------------------------------------------- -->

<CHANNEL

NAME [=] "fabric\_channel\_name"

ORIENT [=] ( HOR[I]Z[ONTAL] | VERT[ICAL] )

COUNT [=] count

>

<REGION> x1 y1 x2 y2 </REGION>

</CHANNEL>

<!-- -------------------------------------------------------------------- -->

<SEGMENT

NAME [=] "fabric\_segment\_name"

INDEX [=] index

WIDTH [=] width

>

<LINE> x1 y1 x2 y2 </LINE>

<TIMING[\_ANALYSIS]

R[ES] [=] res

C[AP] [=] cap

/>

</SEGMENT>

</FABRIC>

## Circuit File Syntax

<!-- ========================================================================//

<CIRCUIT NAME [=] "circuit\_name">

<!-- -------------------------------------------------------------------- -->

<PORT|I[NPUT]O[UTPUT]

NAME [=] "circuit\_name"

MASTER [=] "arch\_name"

STATUS [=] "float" | "placed" | "fixed"

SOURCE [=] ".names"|".latch"|".input"|".output"

>

<PIN

NAME [=] "name"

TYPE [=] "input" | "output" | "clock" | "power"

/>

<PLACE[MENT]>

NAME [=] "fabric\_name"

X [=] “x”

Y [=] “y”

Z|SLICE [=] “z”

</PLACE[MENT]>

</PORT|I[NPUT]O[UTPUT]>

<!-- -------------------------------------------------------------------- -->

<BLOCK|P[HYSICAL]B[LOCK]

NAME [=] "circuit\_name"

MASTER [=] "arch\_name"

STATUS [=] "float" | "placed" | "fixed"

SOURCE [=] ".names"|".latch"|".input"|".output"

>

<PACK[ING]

NAME [=] "blif\_name"

<HIER> "hier\_name" ... </HIER>

</PACK[ING]>

<PLACE[MENT]>

NAME [=] "fabric\_name"

X [=] “x”

Y [=] “y”

</PLACE[MENT]>

<REGION> x1 y1 x2 y2 </REGION>

<RELATIVE>

"left" | "right" | "bottom" | "top" "circuit\_name" ...

</RELATIVE>

</BLOCK|P[HYSICAL]B[LOCK]>

<!-- -------------------------------------------------------------------- -->

<INST

NAME [=] "circuit\_name"

MASTER [=] "arch\_name"

SOURCE [=] ".names"|".latch"|".input"|".output"

>

<CLOCK

TYPE [=] "fe" | "re" | "ah" | "al" | "as"

STATE [=] "0" | "1" | "2" | "3"

/>

<PIN

NAME [=] "name"

TYPE [=] "input" | "output" | "clock" | "power"

/>

</INST>

<!-- -------------------------------------------------------------------- -->

<NET

NAME [=] "circuit\_net\_name"

TYPE [=] "signal" | "clock" | "power" | "global"

STATUS [=] "open" | "g[lobal\_]routed" | "routed" | "fixed"

ROUTABLE [=] ON | OFF ( TRUE | FALSE )

>

<PIN

INST [=] "inst\_name"

PORT [=] "port\_name"

PIN [=] "pin\_name"

TYPE [=] "input" | "output" | "clock" | "power"

/>

<G]LOBAL\_]ROUTE

NAME [=] "fabric\_channel\_name"

LENGTH [=] length

/>

<ROUTE

<PIN

INST [=] "inst\_name"

PORT [=] "port\_name"

PIN [=] "pin\_name"

TYPE [=] "input" | "output" | "clock" | "power"

/>

<SEGMENT

NAME [=] "segment"

TRACK [=] track

>

<CHANNEL> x1 y1 x2 y2 </CHANNEL>

</SEGMENT>

<S[WITCH]B[OX]

NAME [=] "fabric\_sb\_name"

>

<SIDES>

"left" | "right" | "bottom" | "top" index

"left" | "right" | "bottom" | "top" index

</SIDES>

</S[WITCH]B[OX]>

</ROUTE>

</NET>

</CIRCUIT>

# Output Files

Toro optionally generates the following output files:

| **Output File** | **Description** |
| --- | --- |
| log | Reflects all program output messages (generate by either Toro or VPR). |
| options | Reflects all program execution options (both default and user-defined). |
| architecture | Reflects the current FPGA architecture description.  The output architecture file format is identical to the input file format. |
| fabric | Reflects the current FPGA fabric model.  The output fabric file format is identical to the input fabric file format. |
| circuit | Reflects the current FPGA circuit design.  The output circuit file format is identical to the input circuit file format. |
| XML | Reflects the current architecture description *in VPR’s XML format*. |
| BLIF | Reflects the current circuit design *in VPR’s BLIF format*. |
| LAFF | Represents a fabric and circuit view that you can use to quickly display and visualize pack, place, and route results.  LAFF stands for LISP Archival File Format and was originally created by TI for EDA data storage. |
| RC delays | Represents circuit RC delays.  Note: The functionality is not currently supported. |

# Use Models

Toro support two target FPGA users - the **architect** and the **designer**.

The **architect** is expected to be 'expert' in FPGA physical design. The architects' goal is to use Toro to explore various FPGA architectures and select an appropriate fabric model.

The **designer** is expected to be 'experienced' with FPGA physical design. The designer's goal is to use Toro to implement a circuit design based on a given fabric model.

The following diagrams illustrate several use models from the architect and designer perspectives.

# Appendix: Building Toro

Toro is an open source project developed at TI's System Architecture Lab. Source code is available under the GNU General Purpose License (GPL).

The following steps summarize how to build Toro.

1. Download source from: <https://vtr-verilog-to-routing.googlecode.com/svn/trunk>

This should include the following source directories:

toro

vpr

libarchfpga

libpccts

libcommon\_c

libcommon\_c++

1. Make VPR’s API library.

cd vpr

make

1. Make Toro executable.

cd toro

make

Note: These build instructions need more validation...

To date, Toro has been built and tested on the following platforms:

* Linux (Red Hat 64-bit), using gcc version 3.4.6 or newer
* Linux (Ubuntu 32-bit), using gcc version 4.6.3 or newer
* SunOS 5.8 (Solaris 2.7) using gcc version 2.95.2 or newer
* Windows 7 using Visual C++ 2010 Express

# Appendix: Preprocessor Support

Toro automatically applies the standard C macro preprocessor when reading an input options file.

The macro preprocessor, “cpp”, includes support for the following set of special directives:

|  |  |
| --- | --- |
| #define *symbol*  #define *symbol* *value* | Defines a *symbol*. Optionally defines a value to be substituted for each occurrence of the *symbol*. |
| #undef *symbol* | Undefines a *symbol*. |
| #ifdef *symbol*  #endif | Conditional compilation if *symbol* has been previously defined. |
| #ifndef *symbol*  #endif | Conditional compilation if *symbol* has not been previously defined. |
| #if defined *symbol*  #elif defined *symbol*  #else  #endif | Conditional compilation if *symbol* has been previously defined. |
| #include *file* | Specifies a *file* to be included. |

# Appendix: Regular Expression Support

Many of the Toro options include support for regular expression (RE) pattern matching. A valid regular expression is constructed as follows:

* The special characters are: + \* ? . [ ] ^ $ \ |
* A single character RE followed by a plus ‘+' matches one or more occurrences of the RE. For example, [A-Z]+ matches one or more upper-case characters.
* A single character RE followed by an asterisk ‘\*' matches zero or more occurrences of the character. For example, [A-Z]\* matches zero or more upper-case characters.
* A single character RE followed by a question mark ‘?' matches zero or one occurrences of the RE. For example, ab?c matches either abc or ac.
* A period or dot ‘.' matches any single character except the newline character. For example, ab.c matches either abxc or abyc.
* A set of characters enclosed in ‘[]'s matches any single character. For example, [abc] matches a, b, or c. A range is defined with a dash. For example, [A-Z] matches any upper-case letter. If the first character of the set is ‘^', then the RE matches any character except those in the set. For example, [^abc] matches any character except a, b, or c.
* If caret or circumflex ‘^' is at the beginning of a RE, then the matched string must be at the beginning of the line.
* If a dollar sign ‘$' is at the end of a RE, then the matched string must be at the end of the line.
* The concatenation of two or more RE's matches the corresponding concatenation of strings. For example, [A-Z][a-z]\* matches any capitalized word.
* A backslash ‘\’ followed by any special character matches the literal character (i.e. a ‘\' escapes a special character).
* A pipe or ‘|’ matches one of multiple RE. For example, a|b|c matches either character a, b, or c.

# Appendix: References

1. Kuon, R. Tessier, and J. Rose, "FPGA Architecture: Survey and Challenges", Foundations and Trends in EDA, Nov 2006, pp. 135-253.
2. D. Chen, J. Cong, and P. Pan, "FPGA Design Automation: A Survey", Foundations and Trends in EDA, Nov 2006, pp. 195-330.
3. V. Betz, "Architecture and CAD for Speed and Area Optimization of FPGAs", University of Toronto Ph.D Dissertation, 1998.
4. V.Betz, J.Rose, and A.Marquardt, "Architecture and CAD for Deep-Submicron FPGAs", Kluwer Academic Publishers, 1999.
5. J. Luu, J. Anderson, and J. Rose, "Architecture Description and Packing for Logic Blocks with Hierarchy, Modes and Complex Interconnect", Proc. FPGA, 2011, pp.227-236.
6. V. Betz, T. Campbell, et al, "VPR User's Manual", 6.0 Full Release, University of Toronto, January 19, 2012.
7. ***rc delay reference ?*** W.C. Elmore, "The Transient Response of Damped Linear Network with Particular Regard to Wideband Amplifiers", J. Applied Physics, 1984, pp. 55-63.
8. ***rc delay reference ?*** C.J. Alpert, A. Devgan, C. Kashyap, "A Two Moment RC Delay Metric for Performance Optimization", ISPD 2000, pp. 73-78.
9. ***arch reference ?*** J. Rose and S. Brown, "Flexibility of Interconnection Structures for Field Programmable Gate Arrays", JSSC, March 1991, pp. 277-282.
10. ***clb size reference ?*** E. Ahmed and J. Rose, "The Effect of LUT and Cluster Size on Deep-Submicron FPGA Performance and Density", IEEE Trans. on VLSI, March 2004, pp. 288-298.
11. ***prior pack reference ?*** J. Cong, J. Peck, and Y. Ding, "RASP: A General Logic Synthesis System for SRAM-based FPGAs", ACM Symp. on FPGAs, 1996, pp. 137-143.
12. ***prior pack reference ?*** H. Shin and C. Kim, "A Simple Yet Effective Technique for Partitioning", IEEE Trans. on VLSI, Sept 1993, pp. 380-386.
13. ***prior pack reference ?*** J. Lam and J. Delosme, "Performance of a New Annealing Schedule", DAC, 1988, pp. 306-311.
14. ***prior route reference ?*** L. McMurchie and C. Ebeling, "PathFinder: A Negotiation-Based Performance-Driven Router for FPGAs", ACM Symp. on FPGAs, 1995, pp. 111-117.

# Revision History

WIP – Work in Progress – No revision history will be available until Version 1.0 is REady…

| **Revision Number** | **Revision**  **Date** | **Revision**  **Description** |
| --- | --- | --- |
| 1.0.0 | 21 December 2012 | Generated by Jeff Rudolph, [jrudolph@ti.com](mailto:jrudolph@ti.com), 214.480.1703. |