ftlutil

# FARADAY DESIGN KIT FTL UTILITY

**User Guide**
**Rev.: 202001.0.0v1.0**
**Issue Date: July 2020**



# REVISION HISTORY

## Faraday Design Kit ftlutil User Guide

| Date | Rev. | From | To |
|---|---|---|---|
| Sept. 2003 | 1.0 | - | Original |
| Apr. 2004 | 2.0 | - | 2004 major release |
| Nov. 2006 | 200601.4.1v1.0 | - | 2006 major release |
| Jan. 2007 | 200701.1.1v1.0 | - | 2007 major release |
| Jan. 2008 | 200801.1.1v1.0 | - | 2008 major release |
| Nov. 2008 | 200802.1.2v1.0 | - | 200802.1.2 hot fix |
| Mar. 2009 | 200901.1.1v1.0 | - | 2009 major release |
| Mar. 2010 | 201001.1.1v1.0 | - | 2010 major release |
| Apr. 2011 | 201101.0.0v1.0 | - | 2011 major release |
| May 2012 | 201201.0.0v1.0 | - | 2012 major release |
| Mar. 2013 | 201301.0.0v1.0 | - | • 201301 major release<br>• Pages 6 and 7: Added Tester and NamingCheck descriptions<br>• Updated Section 3.1 |
| Nov. 2013 | 201302.0.0v1.0 | - | 201302 major release |
| May 2014 | 201401.0.0v1.0 | - | 201401 major release |
| Dec. 2018 | 201806.0.0v1.0 | - | 201806 major release |
| Jul. 2020 | 202001.0.0v1.0 | - | 202001 major release |

Printed in Taiwan 2020



Faraday Technology Corporation
No. 5, Li-Hsin Road III, Hsinchu Science Park, Hsinchu City, Taiwan 300, R.O.C.

Faraday's home page can be found at:
http://www.faraday-tech.com

# TABLE OF CONTENTS


Chapter 1 Introduction ..... 1
Chapter 2 Syntaxes of Core Program ..... 3
Chapter 3 Control File ..... 5
3.1 Header ..... 8
3.2 ControlValueGroup ..... 9
3.3 MapSignalGroup ..... 10
3.4 MaskGroup ..... 11
3.5 PeriodMaskGroup ..... 13
3.6 CheckGroup ..... 18
3.7 CycleGroup ..... 18
Chapter 4 ftlutil Usage ..... 21
4.1 RESEQ Mode ..... 22
4.2 MASK Mode ..... 22
4.3 CHANGEFREQ Mode ..... 24
4.4 CHECK Mode ..... 25
4.5 CYCLE Mode ..... 25
4.6 A More Convenient UI Script: ftlutil_ui ..... 25
Chapter 5 Message List ..... 29
5.1 Messages Related to Command Argument ..... 30
5.2 Messages Related to Control File ..... 30
5.3 Messages Related to FTL ..... 33
5.4 Messages Related to Simulation Mismatch File ..... 33

# LIST OF TABLES

Table 3-1. The Definition of Trigger Actions ..... 13
Table 3-2. Vector Change Table ..... 14

# LIST OF FIGURES

Figure 1-1. Operation Flow of ftlutil ..... 2
Figure 3-1. Different Force Behaviors between Current Version and Previous Version ..... 9
Figure 4-1. Re-sequence FTL Pattern of IP to Design Top Module ..... 22
Figure 4-2. FTL Verification Flow with Faraday Design Kits of ftl2ver, fsim, and ftlutil ..... 23
Figure 4-3. Difference between CHANGEFREQ and RESEQ Patterns ..... 24

# Chapter 1

# Introduction



**ftlutil** is a utility used to control the ftl patterns. It provides the following functions:

- Re-sequence: Record the pattern sequence according to a golden *ftl* header file or pattern.
- Mask file: Mask or replace a pattern according to the simulation mismatch output file.
- Change frequency: Users need to manually write a new *ftl* header file with new timgen utility (Which has a cycle time longer than the original one) in it. Based on the new header file, **ftlutil** can merge the patterns in the continuous cycles into one pattern.
- Conditional mask: Mask the value of a signal according to the value of another conditional signal
- Periodical mask: Periodically mask the signal value according to a trigger signal, mainly for USB 2.0
- Pattern compression/expansion
- Check sequence: Check the pattern sequence between the golden *ftl* pattern and other *ftl* patterns
- Count total cycle: Count the total cycles of a series of the *ftl* patterns

Figure 1-1 depicts the operation flow of **ftlutil**.

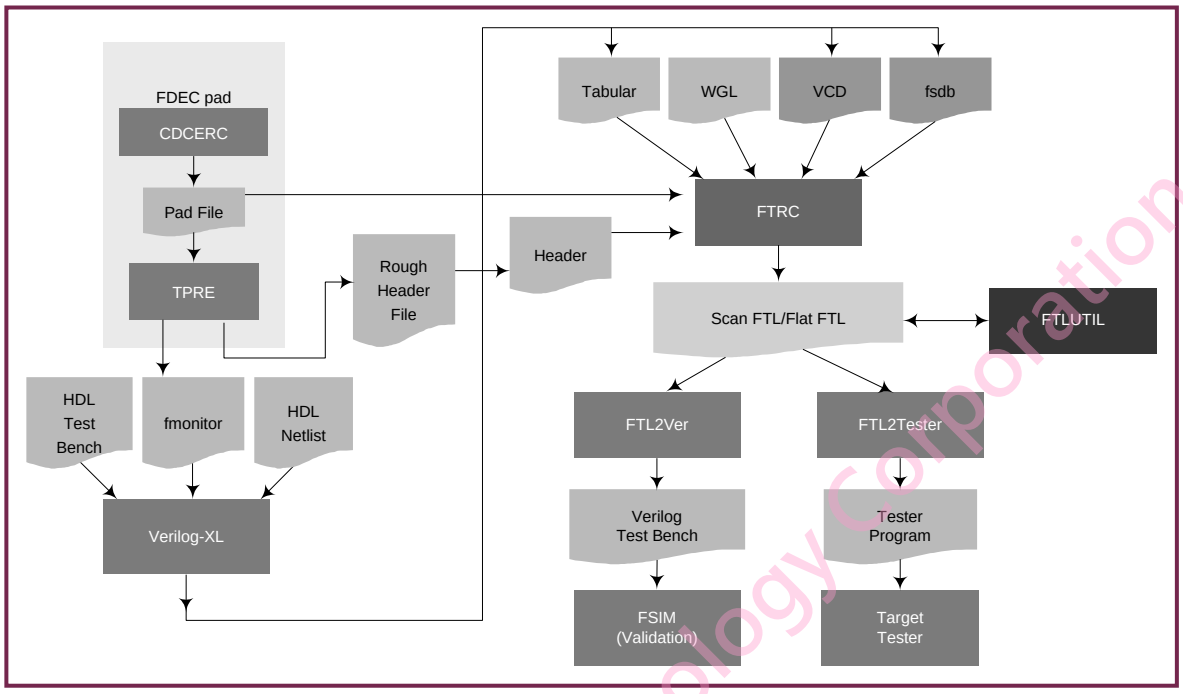


**Figure 1-1. Operation Flow of ftlutil**

# Chapter 2

# Syntaxes of Core Program



Below is the syntax of **ftlutil**:

| | |
|---|---|
| -i | Control file |
| -t | Print template control file |
| -log | Output log file |
| -v | Show version |
| -h | Show message of the usage |

# Chapter 3

# Control File



This chapter contains the following sections:

3.1 Header
3.2 ControlValueGroup
3.3 MapSignalGroup
3.4 MaskGroup
3.5 PeriodMaskGroup
3.6 CheckGroup
3.7 CycleGroup

The template of the control file is shown below, which mainly includes seven sections:

- Header
- ControlValueGroup
- MapSignalGroup
- MaskGroup
- PeriodMaskGroup
- CheckGroup
- CycleGroup

```
// Faraday Tester Language Utility Version 200209.1.1
// Copyright (c) 2001-2002 by Faraday Technology Corporation
// template control file

Header {
     Tester = E320 ;  // [E320|NONE]
     Mode = RESEQ ; // [RESEQ|CHECK|MASK|CYCLE|CHANGEFREQ]
     NamingCheck = ON ; // [ON|OFF]
     InputFile    = "source.ftl";
     OutputFile   = "output.ftl";
     TargetHeader = "TargetHeaderFile";
     TimeGen = Source; //[Source|Target]
     Conversion = None; //[Compress|Compress:MinRepeatNum|Expand|None]
     DefaultInputValue = '0' ;
     DefaultInoutValue = 'X' ;
     DefaultOutputValue = 'X' ;
}

/*
ControlValueGroup {
       TARGET_SIG =  '1' ;
}
*/

/*
MapSignalGroup {
       SOURCE_SIG = TARGET_SIG;
       SOURCE_BI_SIG = {TARGET_IN_SIG,TARGET_OUT_SIG,TARGET_EN_SIG:OUTIF1} ; //[OUTIF1|OUTIF0]
}
*/

/*
MaskGroup {
       MismatchFile = "mismatch.out" -mode MASK; //[MASK|REPLACE]
       MaskRegion TARGET_SIGNAL -begin 0 -end 10 -value '0';
```

```
        MaskCondition TARGET_SIGNAL -value '0' -when COND_SIGNAL = value;
}
*/

/*
PeriodMaskGroup {
        RefSignal TARGET_SIGNAL -Trigger Posedge;//[Level0|Level1|Posedge|Negedge|ALL]
        Period  8 -begin 0 -end 1000;
        Period  8 -backward -begin 0 -end 1000;
        MaskSignal TARGET_SIGNAL1 -Mask "00011001";
        MaskSignal TARGET_SIGNAL2 -Mask "00011011";
}
*/

/*
CheckGroup {
        CheckFile = "check_file_name";
}
*/

/*
CycleGroup {
        CheckFile = "check_file_name";
}
*/
```

## 3.1 Header

The header section defines the operation mode and the basic information. This section is required for all operation modes. The commands defined in this section are explained below:

- Tester: Tester E320 has lots of special rules that a pattern needs to satisfy to ensure its testability. This option allows users to assign whether to check the pattern with the E320 tester rules or not. If the output pattern will be tested on E320, please make sure to turn this option as "E320". Otherwise, users can set it as "NONE".
- Mode: Define the available operation modes, including [RESEQ|CHECK|MASK|CYCLE|CHANGEFREQ]
- NamingCheck: For the character length of a signal name, there is a limitation. For IP design, users can turn OFF this option to skip this check.
- InputFile: Define the original ftl file in the RESEQ, MASK, and CHANGEFREQ modes and the golden ftl file in the CHECK mode
- OutputFile: Define the result ftl file in the RESEQ, MASK, and CHANGEFREQ modes
- TargetHeader: Define the file that contains the target pattern sequence. This file can be an ftl file or an ftl header file. This command is used only in the RESEQ mode.
- TimeGen: Define timegen in the source file or in the target header file used in the result ftl file. This command is used only in the RESEQ mode. In the CHANGEFREQ mode, ftlutil automatically uses timegen in the target header file.
- Conversion: Define the conversion type of the generated the ftl pattern to be either compressed or expanded. "Compression" means converting the successive identical pattern into one REPEAT pattern, while "Expansion" means converting the SCAN pattern or the REPEAT pattern to the flat ftl pattern. If "Compress:MinRepeatNum" is provided, the repeat number that is less than MinRepeatNum will be expanded.
- DefaultInputValue: Define the value of the input signal that is defined in the target header file but is absent in the source ftl pattern
- DefaultOutputValue: Define the value of the output signal that is defined in the target header file but is absent in the source ftl pattern
- DefaultInoutValue: Define the value of the bidirectional signal that is defined in the target header file but is absent in the source ftl pattern

## 3.2 ControlValueGroup

ControlValueGroup is used to force a signal to a specific value in the merged *ftl* pattern in the RESEQ mode. The command syntax is very simple, and the valid values are 0, 1, L, and H.

```
        TARGET_SIG =  '1' ;
```

Please note that the value has to be enclosed in a pair of single quotation marks as show above.

The time to set ControlValueGroup is when users want to re-sequence *ftl* of an IP to *ftl* of a chip, and some control pins must be set to specific values to enable this IP. Under such a condition, users can use ControlValueGroup to specify the values of the control signals.

Please note that when a signal is defined in ControlValueGroup, the timegen of this signal will be changed to DNRZ 0. The behavior of a pattern will be out of expectation if the change is not made. For example, to keep the test_enable signal to stay high in all cycles, users may assign '1' to the test_enable signal. However, if the timegen of the test_enable signal is RZ, the actual behavior of the test_enable signal will act as a clock signal instead of staying high. To prevent unexpected situations, users should force this signal to a specific value. The timegen of the forced signal should also be changed to DNRZ 0.

Figure 3-1 shows the differences between the previous version and the current version when a signal with RZ timegen is assigned to a forced value of '1'.

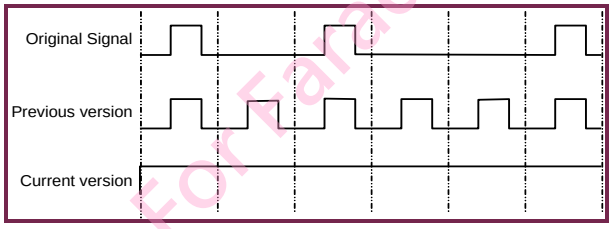


**Figure 3-1. Different Force Behaviors between Current Version and Previous Version**

## 3.3 MapSignalGroup

MapSignalGroup is used to create the mapping relation between the source *ftl* file and the target *ftl* header file. This section is only valid in the RESEQ mode. By default, this tool assumes that the source *ftl* file and the target *ftl* header file contain the same signals. The difference should only be the timegen and the signal sequence. **ftlutil** will assume the signal in the source *ftl* file should be re-sequenced to the position of the signal with the same name in the target *ftl* header file.

Sometimes, the signal name may not be the same between the source *ftl* file and the target *ftl* header file. For example, if users want to map the *ftl* pattern of an IP to the one at the chip level, the signal name may not be the same. At this situation, users can define the mapping relation in MapSignalGroup. The syntax is very simple:

```
SOURCE_SIG_NAME = TARGET_SIG_NAME;
```

Where SOURCE_SIG_NAME is the signal name in the source *ftl* file, and TARGET_SIG_NAME is the signal name in the target *ftl* header file. In addition to defining the signal mapping relation, MapSignalGroup can also define the mapping relationship of a bidirectional signal in the source *ftl* file. Users can map the bidirectional signal in the source *ftl* file to three signals in the target *ftl* header file. The syntax is shown below:

```
SOURCE_BI_SIG = {TARGET_IN_SIG,TARGET_OUT_SIG,TARGET_EN_SIG:OUTIF1}
```

When SOURCE_BI_SIG is the source bidirectional signal, TARGET_IN_SIG is the target input signal, TARGET_OUT_SIG is the target output signal, and TARGET_EN_SIG is the target bidirectional control signal, users should define the active levels of these control signals. The valid options are OUTIF1 and OUTIF0. OUTIF1 means that the value of the control signal is '1' and the bidirectional signal should be in the output mode. On the contrary, OUTIF0 means that the value of the control signal is '0' and the bidirectional signal should be in the input mode.

## 3.4 MaskGroup

MaskGroup is used to define some mask criteria which can help users mask some unwanted output comparisons. It supports three different masking functions, mask mismatch from a simulation output, mask values of a certain signal in a certain region, and mask values of a signal when the conditional signal is triggered.

The syntax of the first function is shown below:

```
MismatchFile = "mismatch.out" -mode MASK; //[MASK|REPLACE]
```

When mismatch.out is the file name of the mismatch output file generated by **fsim** in the Faraday design kit, and the mask mode can be set to either MASK or REPLACE mode, **ftlutil** will mask the values of the mismatch signals to 'X' in the MASK mode, and **ftlutil** will change the values of the mismatch signals to the values in the simulation output, in the REPLACE mode.

The syntax of the section function is shown below:

```
MaskRegion TARGET_SIGNAL -begin 0 -end 10 -value '0';
```

When TARGET_SIGNAL is the signal name to be masked, the first cycle and the last cycle of this signal can be masked by "–begin" and "–end", respectively. When "–value" is used to specify the value of the mask, some users may be confused about this command and the "force" command defined in the ControlValueGroup section. When using the force command, it will change the timegen to DNRZ0, and the value of the entire pattern will be forced to a specified value. On the contrary, the MaskRegion command will only perform the literal change and users can mask a certain region of the time.

The syntax of the last function in the MaskGroup is shown below:

```
MaskCondition TARGET_SIGNAL -value '0' -when COND_SIGNAL = 'value';
```

When TARGET_SINGAL is the signal to be masked, the value defined in the argument, "-value", represents the mask value, and it should be in a pair of single quotation marks as shown above. COND_SINGAL is the conditional signal and the value defined after CON_SIGNAL is the conditional value. When the value of the conditional signal becomes the conditional value, the mask condition will be treated as true, and the corresponding signal will be masked with the mask value.

A small example, as shown below, is used to demonstrate this feature. It is assumed that the input file is the same input file used in the previous example.

```
// Faraday Tester Language Utility Version 2001.09.02
// Copyright (c) 2001 by Faraday Technology Corporation
// template control file

Header {
     Mode = MASK ; // [RESEQ|CHECK|MASK]
     InputFile    = "hs01t.ftl";
     OutputFile   = "new.ftl";
     TargetHeader = "TargetHeaderFile";
     TimeGen = Source; //[Source|Target]
     Conversion = None; //[Compress|Expand|None]
     DefaultInputValue = '0' ;
     DefaultInoutValue = 'X' ;
     DefaultOutputValue = 'X' ;
}
MaskGroup {
       MaskCondition TXREADY -value 'X' -when CLK = 'H';
}

The following is the result:
// FTLUTIL FTL 2.0
// Date: Tue Aug 13 20:10:01 2002
// Source File: hs01t.ftl
// Input: 0  Output: 3   Bidirection: 0

TESTTYPE    FUNC;
INPUT(1)    DATA_IN_15;
OUTPUT(2)    TXREADY;
OUTPUT(2)    LINESTATE_1;
OUTPUT(2)    LINESTATE_0;
OUTPUT(2)  CLK;
TIMEUNIT        1PS;
CYCLE           2000;
STABLEREGION    0;
TIMEGEN(1)   DNRZ,    1000;
TIMEGEN(2)   STROBE,  1500;
SEQUENCE   DATA_IN_15,    TXREADY, LINESTATE_1,  LINESTATE_0, CLK;

BEGIN
XLLL_L;  // 1
XLLL_L;  // 2
XXLL_H;  // 3  << Mask cycle
XXLL_H;  // 4  << Mask cycle
XXLL_H;  // 5  << Mask cycle
XXLL_H;  // 6  << Mask cycle
XLLL_L;  // 7
XLLL_L;  // 8
XLLL_L;  // 9
XLLL_L;  // 10
END
```

## 3.5 PeriodMaskGroup

PeriodMaskGroup is used to help users periodically mask/replace the pre-defined cycles of a pre-specified signal after the trigger signal has been triggered. The original motivation of the period mask feature is to support the testing of some high-speed signals in a chip, whose speed is multiple times of the one for the normal signal of that chip. Due to the limitation of the *ftl* format, users can not specify two different testing speeds in one ftl. To fully test the high-speed signal, users must set the operation frequency of *ftl* to the highest speed of the signal frequency. In such a condition, it will be troublesome for us to compare all cycles to the one of other normal output signals. It seems like to compare multiple times within a cycle of the normal output signal.

To prevent such a trouble, **ftlutil** provides a feature to periodically mask the unnecessary compare cycles.

To enable this feature, PeriodMaskGroup should be defined in the control file listed below.

```
PeriodMaskGroup {
       RefSignal CLK -Trigger Posedge;//[Level0|Level1|Posedge|Negedge]
       Period  8 ;
       MaskSignal TXREADY -Mask "11111100";
       MaskSignal LINESTATE_0 -Mask "11-11100";
       MaskSignal LINESTATE_1 -Mask "11-11100";
       MaskSignal CLK -Mask "11111100";
}
```

This control file shows that the trigger signal is CLK and the trigger action is “Posedge”. When the trigger signal is triggered, **ftlutil** will start to mask four signals, TXREADY, LINESTATE_0, LINESTATE_1, and CLK, for eight cycles.

The definitions of the trigger action are different for different situations. The relations are listed in Table 3-1.

**Table 3-1. The Definition of Trigger Actions**

| | DNRZ | RZ | RO | SBC | STROBE |
|---|---|---|---|---|---|
| Level0 | 0 | 0\|1 | 0\|1 | 0\|1 | L |
| Level1 | 1 | 0\|1 | 0\|1 | 0\|1 | H |
| Posedge | 0 → 1 | 1 | 0 | 0\|1 | L → H |
| Negedge | 1 → 0 | 1 | 0 | 0\|1 | H → L |

For example, when the timegen of the trigger signal is "DNRZ" and the trigger action is "Level0", the trigger signal will be triggered when it is set to '0'. The value defined after the "Period" command shows how many cycles should be masked. The signal name defined after the "MaskSignal" command is the signal to be masked. The binary string defined after the "-Mask" argument indicates which cycle in a particular period should be masked; for example, the length of the binary string should be the same as the value of the period. MSB of the binary string indicates whether the first cycle of the to-be-masked signal should be masked or not; the value '1' indicates masking to 1 and the value '0' indicates masking to 0, while the second bit indicates the status of the mask/replace of the second cycle, the third bit indicates the status of the mask/replace of the third cycle, and so on. It should be noted that the first cycle is the cycle at which the trigger signal is triggered and if the trigger signal is an output signal, it could also be masked/replaced, too. Table 3-2 shows how the vector changed after the user applies different values.

**Table 3-2. Vector Change Table**

| User apply | 0 | 1 | L | H | X | - |
|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| L | L | H | L | H | L | L |
| H | L | H | L | H | H | H |
| Z | L | H | L | H | X | Z |
| X | L | H | L | H | X | X |

Part of the control file used to control **ftlutil** is listed below.

```
// Faraday Tester Language Utility Version 2001.09.02
// Copyright (c) 2001 by Faraday Technology Corporation
// template control file

Header {
     Mode = MASK ; // [RESEQ|CHECK|MASK]
     InputFile    = "hs01t.ftl";
     OutputFile   = "new.ftl";
     TargetHeader = "TargetHeaderFile";
     TimeGen = Source; //[Source|Target]
     Conversion = None; //[Compress|Expand|None]
     DefaultInputValue = '0' ;
     DefaultInoutValue = 'X' ;
     DefaultOutputValue = 'X' ;
}
PeriodMaskGroup {
       RefSignal CLK -Trigger Posedge;//[Level0|Level1|Posedge|Negedge]
       Period  8 ;
```

```
        MaskSignal TXREADY -Mask "XX-XXX--";
        MaskSignal LINESTATE_1 -Mask "LHLH---Z";
        MaskSignal LINESTATE_0 -Mask "XX-11100";
        MaskSignal CLK -Mask "-----100";
}
```

The original *ftl* file and the triggered signal at cycle 3 are listed below.

```
// FTLUTIL FTL 2.0
// Date: Tue Aug 13 20:10:01 2002
// Source File: hs01t.ftl
// Input: 0  Output: 3   Bidirection: 0

TESTTYPE    FUNC;
INPUT(1)    DATA_IN_15;
INOUT(1,2)   TXREADY;
OUTPUT(2)    LINESTATE_1;
OUTPUT(2)    LINESTATE_0;
OUTPUT(2)  CLK;
TIMEUNIT        1PS;
CYCLE           2000;
STABLEREGION    0;
TIMEGEN(1)   DNRZ,    1000;
TIMEGEN(2)   STROBE,  1500;
SEQUENCE   DATA_IN_15,    TXREADY, LINESTATE_1,  LINESTATE_0, CLK;

BEGIN
1LLL_L;  // 1
1LLL_L;  // 2
0LLL_H;  // 3  << Trigger at this time
01LL_H;  // 4
11LL_H;  // 5
0LLL_H;  // 6
1LLL_L;  // 7
1LLL_L;  // 8
0LLL_L;  // 9
0LLL_L;  // 10
END
```

The sample below shows the results of all to-be-mask signals that had been masked to include the trigger signal, but the input states of TXREADY were skipped.

```
// FTLUTIL FTL 2.0
// Date: Tue Aug 13 20:10:01 2002
// Source File: hs01t.ftl
// Input: 0  Output: 3   Bidirection: 0

TESTTYPE    FUNC;
INPUT(1)    DATA_IN_15;
OUTPUT(2)    TXREADY;
OUTPUT(2)    LINESTATE_1;
OUTPUT(2)    LINESTATE_0;
OUTPUT(2)  CLK;
TIMEUNIT        1PS;
CYCLE           2000;
```

```
STABLEREGION    0;
TIMEGEN(1)   DNRZ,    1000;
TIMEGEN(2)   STROBE,  1500;
SEQUENCE   DATA_IN_15,     TXREADY, LINESTATE_1,  LINESTATE_0, CLK;

BEGIN
1LLL_L;  // 1
1LLL_L;  // 2
0XLX_H;  // 3   << Trigger at this time
0XHX_H;  // 4
11LL_H;  // 5
0XHH_H;  // 6
1XLH_L;  // 7
1XLH_H;  // 8
0LLL_L;  // 9
0LZL_L;  // 10
END
```

PeriodMaskGroup can be used to backward mask the pattern of certain event. The following example of the mask command file (mask.cmd) is going to demonstrate this event. When the X_SIN1 signal has a posedge transition, users should mask the nearby eight cycles with the mask pattern, "-XXXXXXX". This mask command enables only one cycle to 166 cycles.

```
Header {
     Mode = MASK ;
     InputFile    = "test.ftl";
     OutputFile   = "test_new.ftl";
     Conversion = None; //[Compress|Expand|None]
}
PeriodMaskGroup {
       RefSignal X_SIN1 -Trigger Posedge;//[Level0|Level1|Posedge|Negedge]
       Period 8 -backward -begin 1 -end 116 ;
       MaskSignal X_SIN1 -Mask "-XXXXXXX";

}
PeriodMaskGroup {
       RefSignal X_SIN1 -Trigger Posedge;//[Level0|Level1|Posedge|Negedge]
       Period 8  -begin 1 -end 116 ;
       MaskSignal X_SIN1 -Mask "-XXXXXXX";
}
```

The original ftl pattern is:

```
// FTLUTIL FTL 2.0
// Date: Thu Feb 26 11:19:33 2009
// Source File: header.ftl
// Input: 2  Output: 2   Bidirection: 0

TESTTYPE        FUNC;
OUTPUT(5)     X_SIN2;
INPUT(1) X_TST_PGM;
INPUT(1) X_FDOT;
OUTPUT(5)     X_SIN1;
```

```
TIMEUNIT        100PS;
CYCLE           16;
STABLEREGION    0;
TIMEGEN(1)    DNRZ,    0;
TIMEGEN(5)    STROBE,  10;

SEQUENCE   X_SIN2,     X_TST_PGM,    X_FDOT,  X_SIN1;


BEGIN
X00X;    // 1 : 0
REPEAT <3>   // 2 - 4
X00X;    // 4 : 48
X00X;    // 5 : 64
X00X;    // 6 : 80
Z00L;    // 7 : 96
REPEAT <102> // 8 - 109
Z00L;    // 109 : 1728
SCAN <2> // 110 - 111
INITIAL:
Z00-;    // 110 : 1744
X_SIN1 LL;
ENDSCAN  // 110 - 111
REPEAT <2>   // 112 - 113
Z00L;    // 113 : 1792
Z00H;    // 114 : 1808
REPEAT <6> //115-118
Z00L;    // 118 : 1872
Z00H;    // 119 : 1888
Z00H;    // 120 : 1904
END
```

The ftl pattern after applying the mask command with **ftlutil** will be:

```
// FTLUTIL FTL 2.0
// Date: Thu Feb 26 13:10:12 2009
// Source File: test.ftl
// Input: 2  Output: 2  Bidirection: 0

TESTTYPE        FUNC;
OUTPUT(5)    X_SIN2;
INPUT(1) X_TST_PGM;
INPUT(1) X_FDOT;
OUTPUT(5)    X_SIN1;

TIMEUNIT        100PS;
CYCLE           16;
STABLEREGION    0;
TIMEGEN(1)    DNRZ,    0;
TIMEGEN(5)    STROBE,  10;

SEQUENCE   X_SIN2,     X_TST_PGM,    X_FDOT,  X_SIN1;


BEGIN
X00X;    // 1 : 0
REPEAT <3>   // 2 - 4
```

```
X00X;    // 4 : 48
X00X;    // 5 : 64
X00X;    // 6 : 80
Z00L;    // 7 : 96
REPEAT <99>  // 8 - 106
Z00L;    // 106 : 1680
Z00X;    // 107 : 1696
Z00X;    // 108 : 1712
Z00X;    // 109 : 1728
SCAN <2> // 110 - 111
INITIAL:
Z00-;    // 110 : 1744
X_SIN1 XX;
ENDSCAN  // 110 - 111
Z00X;    // 112 : 1776
Z00X;    // 113 : 1792
Z00H;    // 114 : 1808
Z00X;    // 115 : 1824
REPEAT <5>   // 116 - 120
Z00L;    // 120 : 1904
Z00H;    // 121 : 1920
Z00H;    // 122 : 1936
END
```

## 3.6 CheckGroup

CheckGroup is used to help users sequence consistently among several *ftl* files. Users can specify a golden *ftl* file in the header section to add the checking of the *ftl* files in the CheckGroup section.

## 3.7 CycleGroup

CycleGroup is used to help users count the total cycles of several pattern files. Both the STIL and FTL pattern files can use this command to count the cycles. This information can help users identify the pattern files that can be loaded into the tester at a time. To enable this function, users should change the operation mode in the header section to CYCLE, and list both the STIL and FTL pattern files in the CycleGroup section.

The report output from Cycle Group has the following three items:

1. CYCLE: The total cycle numbers of the pattern including the repeat commands
2. TESTER_CYCLE: The total cycle numbers of the pattern excluding the repeat commands
3. TOTAL_TIME: CYCLE * (Cycle time in the pattern file), for example: 1274913 * 100 ns = 127,491,300 ns = 127.4913 ms
4. FILE_SIZE: The size of the pattern file.

5. FILE_DATE: The date of the pattern file.

Below is an example of the cycle report:

```
************************************************************************
Faraday Tester Language Utility Version 200802.1.2

Copyright (c) 2001-2007 by Faraday Technology Corp.

Date: Wed Feb 25 11:01:48 2009
************************************************************************
** Information(): Checking license feature FTLUTIL... ok.

************************  SUMMARY  *********************************
OPERATION MODE       : CYCLE
************************************************************************
Parsing [AReCo_Scan_0.ftl] ...
Parsing [AReCo_Scan.ftl] ...
Parsing [ATPG_SEQ_5.stil] ...
Parsing [FA_SEQ_5.stil] ...
************************************************************************************
                      CYCLE REPORT
************************************************************************************
 FILE            TESTER_CYCLE   CYCLE        TOTAL_TIME   FILE_SIZE     FILE_DATE
************************************************************************************
AReCo_Scan_0.ftl    380368    380372         19.0186ms    55134943      2008.10.24
AReCo_Scan.ftl      380368    380372         19.0186ms    55134943      2008.10.24
ATPG_SEQ_5.stil      13385     13385          1.3385ms    1014422       2009.02.24
FA_SEQ_5.stil           20        20           0.002ms    1877          2009.02.24
************************************************************************************
TOTAL:              774141    774149         39.3777ms
************************************************************************************
************************************************************************
Processing Time : 20 s
************************************************************************
** Information: Total: 0 error(s), 0 warning(s) encountered.
************************************************************************
Program ends normally
```

The difference between the tester cycle and the cycle is demonstrated in the following example. If users have an *ftl* pattern as shown below, TESTER CYCLE will ignore the REPEAT command and CYCLE will count the REPEAT command:

```
......
TIMEUNIT        1NS;
   CYCLE           100;
......
BEGIN                          // TESTER_CYCLE            CYCLE
0XXXXXXX......                 //    1                      1
0XXXXXXX......                 //    2                      2
0XXXXXXX......                 //    3                      3
0XXXXXXX......                 //    4                      4
0XXXXXXX......                 //    5                      5
0XXXXXXX......                 //    6                      6
0XXXXXXX......                 //    7                      7
0XXXXXXX......                 //    8                      8
0XXXXXXX......                 //    9                      9
0XXXXXXX......                 //    10                     10
0XXXXXXX......                 //    11                     11
REPEAT<1274899>
0XXXXXXX......                 //    12                     1274910
0XXXXXXX......                 //    13                     1274911
0XXXXXXX......                 //    14                     1274912
0XXXXXXX......                 //    15                     1274913
END
```

# Chapter 4

# ftlutil Usage

This chapter contains the following sections:


4.1 RESEQ Mode
4.2 MASK Mode
4.3 CHANGEFREQ Mode
4.4 CHECK Mode
4.5 CYCLE Mode
4.6 A More Convenient UI Script: ftlutil_ui

## 4.1 RESEQ Mode

When users invoke an IP to a design, a FTL pattern of this IP is used to test this IP at the chip level. For example, as shown in Figure 4-1, when users have an IP, ADDA, in the design, FSC0HD999A and the IP ports, A, B, C, and D, are connected to the top module ports, FA, FB, FC, and FD; **ftlutil** is used to perform re-sequence to the FTL in order to test this IP by using the FA, FB, FC, FD ports.

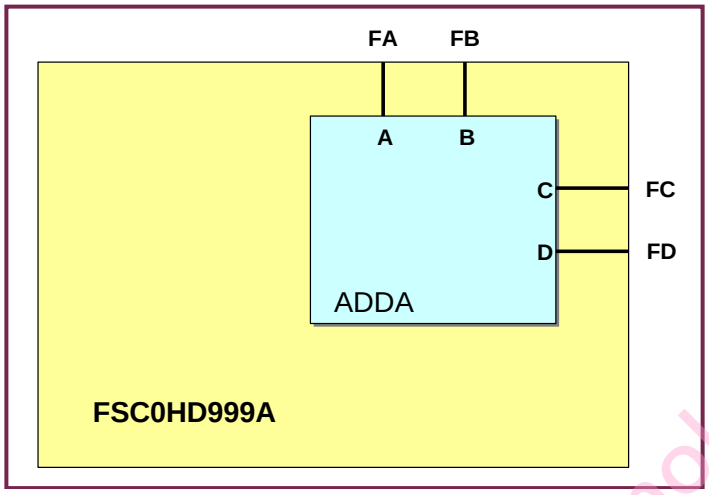


**Figure 4-1. Re-sequence FTL Pattern of IP to Design Top Module**

## 4.2 MASK Mode

Another function of **ftlutil** is masked as shown in Figure 4-2. Users can use the Faraday Design Kit, **ftl2ver,** to translate the FTL pattern to the Verilog test bench, and use this test bench to verify the FTL pattern through a Verilog simulation. The test bench has some built-in API to check the simulation result of the output signals. If the simulation result of an output signal is different from the predicted result in the original FTL pattern, it is called a simulation mismatch. With the built-in API in the test bench, all the simulation mismatches will be written to the mismatch file (.out file) after the simulation. In order to make sure that the pattern works normally at the tester, based on the mismatch file, Faraday Design Kit, **ftlutil**, can automatically mask the simulation mismatches as "X" (don't care) or replace the simulation mismatches with the simulation results. Users should mask these values very carefully because once the value has been masked as "X", the signal at the corresponding cycle will not be checked at a tester. It means the more users masked, the less a tester can check. The tester has some limitations to avoid the testing errors and overkill the good die. It is necessary for users to mask some dangerous mismatches.

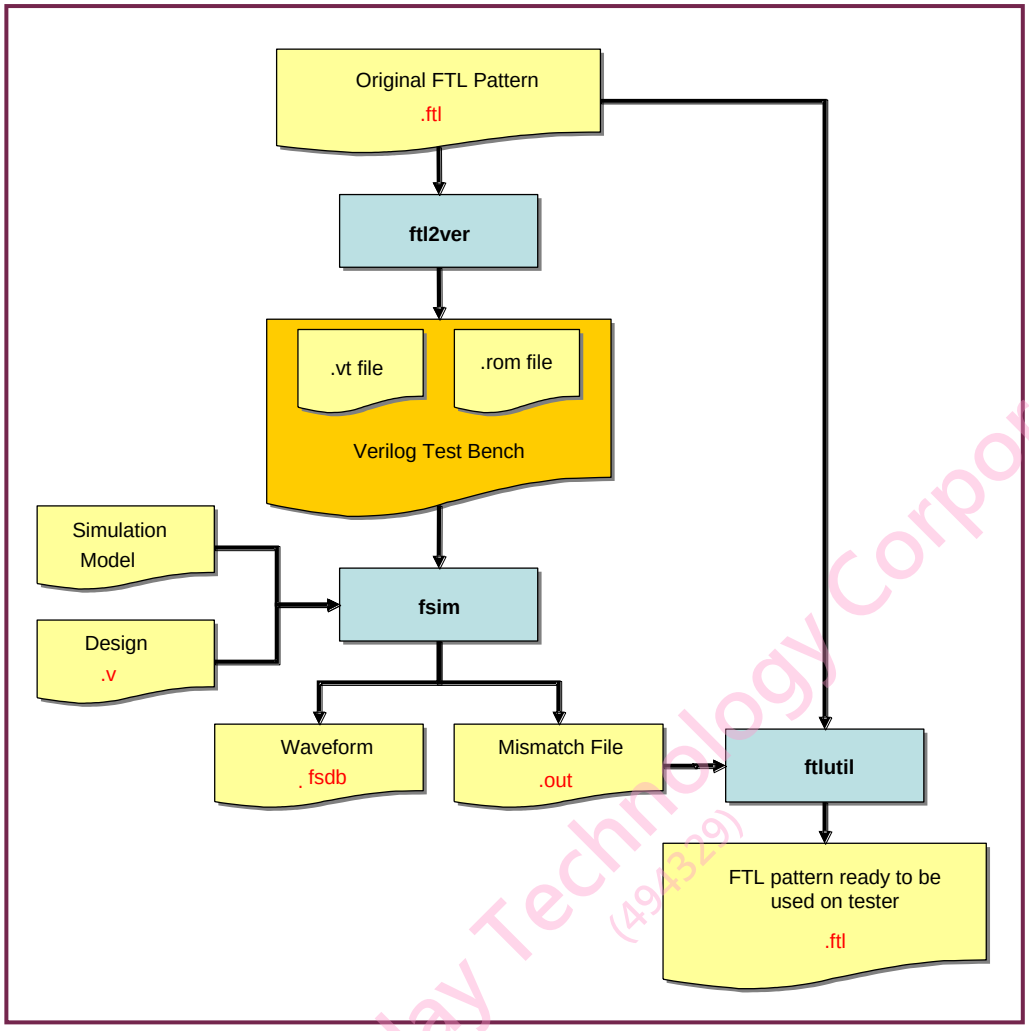


**Figure 4-2. FTL Verification Flow with Faraday Design Kits of ftl2ver, fsim, and ftlutil**

## 4.3 CHANGEFREQ Mode

While adopting an IP into the user chip, the cycle time of the IP pattern might not exactly the same as the chip. The CHANGEFREQ function allows users to change the cycle time of the pattern the same as the chip. While using CHANGEFREQ, the waveform will still be the same. Only the cycle time of the pattern will be changed. While using RESEQ function to make a pattern with new frequency, the cycle time will be changed and also the waveform will be magnified or shrunk as the ratio between the original and new frequencies. Figure 4-3 demonstrates this. Beginning from the original waveform shown at the middle of Figure 4-3, the red dotted line indicates the cycle time of the pattern after CHANGEFREQ. The cycle time is amplified five times. The five cycles (Bits) of the original pattern is mixed to be only one cycle (Bit). By changing TIMEGEN of the pattern, the simulation waveform can keep the same. The upper side of the figure shows the pattern after RESEQ. The cycle is also amplified five times. The original pattern of one cycle is the same pattern after RESEQ. TIMEGEN is controlled by users based on the settings in the *ftl* header file.

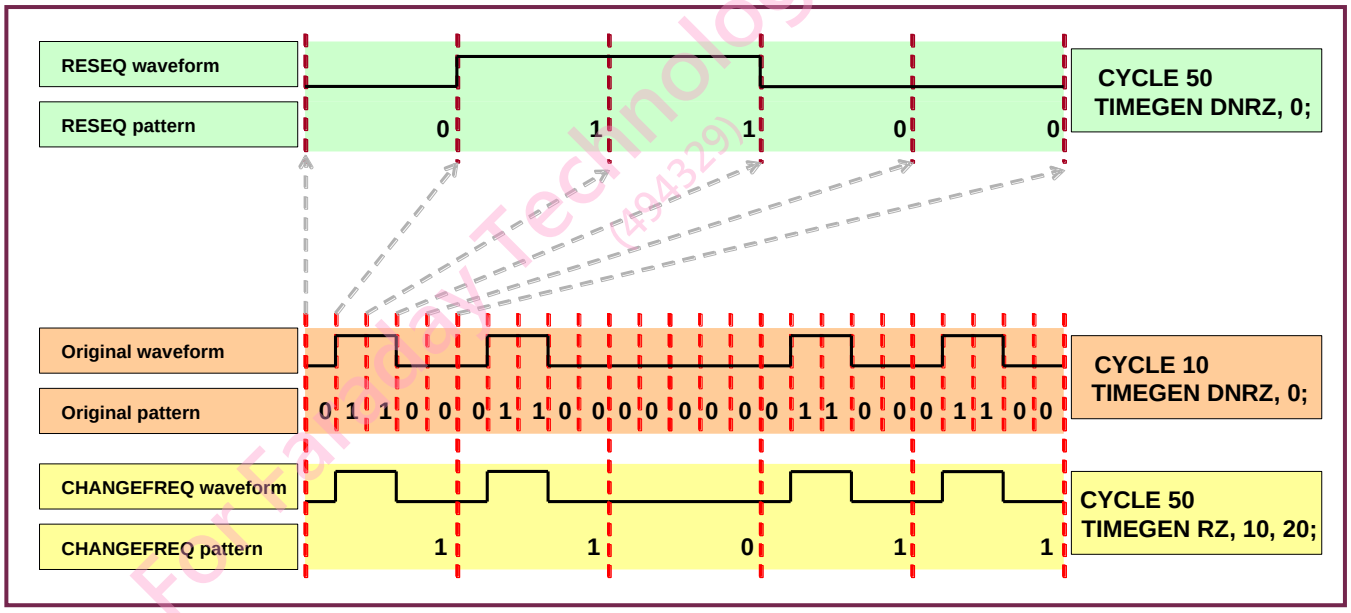


**Figure 4-3. Difference between CHANGEFREQ and RESEQ Patterns**

## 4.4 CHECK Mode

Similar to Section 3.6, **ftlutil** can help users check the sequence and consistency of the ftl pattern. When using the option, CheckGroup, users can specify a golden *ftl* file in the header section to check the *ftl* files in the CheckGroup section.

## 4.5 CYCLE Mode

Similar to Section 3.7, **ftlutil** can help users check the number of cycles in each pattern. For this function, different types of ATE testers have different limitations in number of cycles. Users can use this to check whether the total tester cycle of the pattern is larger than the limitation or not.

## 4.6 A More Convenient UI Script: ftlutil_ui

For different usages of the **ftlutil** utility, users should write different types of command files. ftlutil_ui is a convenient way to help users in writing a command. When using **ftlutil** to modify a pattern, such as re-sequence, mask, check a pattern, and check the *ftl* pattern or the cycle number of a pattern, users must write the command file for each pattern. If more patterns need to perform **ftlutil**, users must generate more command files. However, the difference between these command files is the input file names. ftlutil_ui can also help this situation. ftlutil_ui has four modes that users can query by typing: ftlutil_ui –h.

```
ftclnx43:[149] ftlutil_ui -h
Now executing /home/ddinteg/designkit/major/EXE/ftlutil.lnx
-----------------------------------------------------------------
  Faraday ASIC Design Kit (UI) : ftlutil Version  200802.1.2

   Copyright (c) 2002 - 2007 by Faraday Technology Corp.

  Syntax : ftlutil_ui [re_seq | mask | chks | chk_cycle | rename | -h | -v]

  Funciton: FTL Manipulating Utilities

-----------------------------------------------------------------
```

1. re_seq mode: ftlutil_ui helps users generate the RESEQ command file to perform the re-sequence function. Users can use regular expression in the input and output names in the command file, ftlutil_ui can also help perform the RESEQ command to all the files at the user working directory. A RESEQ command file generated by ftlutil_ui is listed below.

```
Header {
     Mode = RESEQ ;
     InputFile    = "ori_*.ftl";
     OutputFile   = "*.ftl";
     TargetHeader = "TargetHeaderFile";
     TimeGen = Source; //[Source|Target]
     Conversion = None; //[Compress|Expand|None]
     DefaultInputValue = '0' ;
     DefaultInoutValue = 'X' ;
     DefaultOutputValue = 'X' ;
}

ControlValueGroup {
       TARGET_SIG =  '1' ;
}

MapSignalGroup {
       SOURCE_SIG = TARGET_SIG;
       SOURCE_BI_SIG = {TARGET_IN_SIG,TARGET_OUT_SIG,TARGET_EN_SIG:OUTIF1} ; //[OUTIF1|OUTIF0]
}
```

2. Mask mode: ftlutil_ui helps users generate the mask command file to perform the mask function. Users need to add InputFile and OutputFile in the command file to perform ftlutil_ui for masking the ftl pattern. An example of the generated mask command file by using ftlutil_ui is listed below.

```
Header {
     Mode = MASK ;
     InputFile    = "source.ftl";
     OutputFile   = "output.ftl";
     Conversion = None; //[Compress|Expand|None]
}
MaskGroup {
       MismatchFile = "mismatch.out" -mode MASK; //[MASK|REPLACE]
       MaskRegion TARGET_SIGNAL -begin 0 -end 10 -value '0';
       MaskCondition TARGET_SIGNAL -value '0' -when COND_SIGNAL = value;
}
PeriodMaskGroup {
       RefSignal TARGET_SIGNAL -Trigger Posedge;//[Level0|Level1|Posedge|Negedge]
       Period  8 -begin 0 -end 1000;
       MaskSignal TARGET_SIGNAL1 -Mask "00011001";
       MaskSignal TARGET_SIGNAL2 -Mask "00011011";
}
```

3. chks mode: ftlutil_ui helps users generate the CHECK command file to perform the checking of the ftl function. ftlutil_ui will automatically search for the ftl patterns at the current working directory and generate a command file. Users can use this command file to perform ftlutil_ui again to check these ftl patterns. An example of the generated command file is listed below:

```
Header {
     Mode = CHECK ;
     InputFile    = "CP_MCP100_ATPG.ftl";
}

CheckGroup {
    CheckFile = "CP_MCP100_ATPG.ftl" ;
    CheckFile = "CP_MCP100_mb10.ftl" ;
    CheckFile = "CP_MCP100_mb1.ftl" ;
    CheckFile = "CP_MCP100_mb2.ftl" ;
    CheckFile = "CP_MCP100_mb3.ftl" ;
    CheckFile = "CP_MCP100_mb4.ftl" ;
    CheckFile = "CP_MCP100_mb5.ftl" ;
    CheckFile = "CP_MCP100_mb6.ftl" ;
    CheckFile = "CP_MCP100_mb7.ftl" ;
    CheckFile = "CP_MCP100_mb8.ftl" ;
    CheckFile = "CP_MCP100_mb9.ftl" ;
    CheckFile = "CP_MCP100_ScanCh.ftl" ;
    CheckFile = "CP_MCP100_SEQ.ftl" ;
}
```

4. chk_cycle mode: ftlutil_ui helps users generate the CYCLE command file to perform a checking of the cycle function. ftlutil_ui automatically searches for the patterns, including the ftl pattern (*.ftl* files) and the stil pattern (*.stil* files), under current working directory and generate the corresponding command file. Users can use this command file to perform ftlutil_ui again to check the cycle number of these ftl and stil patterns. So far, only the chk_cycle command supports the stil format. For other related usages of the stil pattern, please refer to the design kit **stilutil**. An example of the generated command file is listed below:

```
Header {
     Mode = CYCLE ;
}

CycleGroup {
    CheckFile = "AReCo_Scan_0.ftl" ;
    CheckFile = "AReCo_Scan.ftl" ;
    CheckFile = "ATPG_SEQ_5.stil" ;
    CheckFile = "FA_SEQ_5.stil" ;
}
```

5. Rename mode: In order to use re-sequence multiple patterns with ftlutil_ui, users need to add prefix "ori_" to all patterns. ftlutil_ui also supports this function. To add this prefix, users need to execute "ftlutil_ui rename +ori_" after all the FTL files at the working directory have added the "ori_" prefix. The help message in the re-name mode is listed below.

```
ftclnx43:[215] ftlutil_ui rename
Now executing /home/ddinteg/designkit/dev/EXE/ftlutil.lnx
-----------------------------------------------------------------
  Faraday ASIC Design Kit (UI) : ftlutil Version  200802.1.2

   Copyright (c) 2002 - 2007 by Faraday Technology Corp.

  Funciton: FTL Manipulating Utilities

-----------------------------------------------------------------
 Syntax : ftlutil_ui rename [+|-]prefix [force]
```

# Chapter 5

# Message List



This chapter contains the following sections:

5.1 Messages Related to Command Argument
5.2 Messages Related to Control File
5.3 Messages Related to FTL
5.4 Messages Related to Simulation Mismatch File

The messages listed below are generated by **ftlutil**.

## 5.1 Messages Related to Command Argument

"ARG-1" "No control file specified"
"ARG-2" "No log file specified, use stdout instead"
"ARG-3" "Unknown command line argument <%s>"
"ARG-4" "Can't open log file <%s>, use stdout instead"
"ARG-5" "No control file specified"
"ARG-6" "Can't open control file <%s>"

## 5.2 Messages Related to Control File

"CTRL-1" "%s: Syntax error %s"
"CTRL-2" "Pin<%s> of the target header has not mapped pin nor default value\n =>Force %s=%c"
"CTRL-3" "Bidirectional pin<%s> had been mapped to an input pin<%s>"
"CTRL-4" "Bidirectional pin<%s> had been mapped to an output pin<%s>"
"CTRL-5" "Pin Type Mismatch:\n Source:[%s] <=> Target:[%s]\n Force %s=?"
"CTRL-6" "Pin<%s> of the source FTL has not mapped pin in the target header file"
"CTRL-7" "Translating pattern error: %s"
"CTRL-8" "Masking pattern error: %s"
"CTRL-9" "Operation mode must be specified in the header section"
"CTRL-10" "Input FTL file must be specified in the header section"
"CTRL-11" "Target header file must be specified in the header section"
"CTRL-12" "Output file must be specified in the RESEQENCE mode"
"CTRL-14" "%s:%d Unsupported operation mode[%s]"
"CTRL-15" "%s:%d Can't assign the default input value to [%c], set the default input value to [0]"
"CTRL-16" "%s:%d Can't assign the default output value to [%c], set the default output value to [X]"
"CTRL-17" "%s=> %s"
"CTRL-18" "Parser source FTL file:%s"
"CTRL-19" "%s=> %s"
"CTRL-20" "Parser header FTL file:%s"
"CTRL-21" "%s:%d Unsupported timegen option[%s]"

"CTRL-22" "%s:%d Unsupported conversion option[%s]"
"CTRL-23" "%s=> can't open file for reading"
"CTRL-24" "%s:%d Can't find the pin <%s[%d]> in the source header file"
"CTRL-24" "%s:%d Can't find the pin <%s> in the source header file"
"CTRL-25" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-25" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-26" "%s:%d The pin <%s[%d]> in the target header file has been mapped before"
"CTRL-26" "%s:%d The pin <%s> in the target header file has been mapped before"
"CTRL-27" "Bidirectional pin<%s> had been mapped to an input pin<%s>"
"CTRL-28" "Bidirectional pin<%s> had been mapped to an output pin<%s>"
"CTRL-29" "Pin type mismatch:\n  Source:[%s] <=> Target:[%s]\n Force %s=?"
"CTRL-30" "%s:%d Can't find the pin <%s[%d]> in the source header file"
"CTRL-30" "%s:%d Can't find the pin <%s> in the source header file"
"CTRL-31" "%s:%d Source pin<%s> is not a bi-directional pin"
"CTRL-32" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-32" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-33" "%s:%d The pin <%s[%d]> in the target header file has been mapped before"
"CTRL-33" "%s:%d The pin <%s> in the target header file has been mapped before"
"CTRL-34" "%s:%d Target input pin<%s> is not a input pin"
"CTRL-35" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-35" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-36" "%s:%d The pin <%s[%d]> in the target header file has been mapped before"
"CTRL-36" "%s:%d The pin <%s> in the target header file has been mapped before"
"CTRL-37" "%s:%d Target output pin<%s> is not a output pin"
"CTRL-38" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-38" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-39" "%s:%d The pin <%s[%d]> in the target header file has been mapped before"
"CTRL-39" "%s:%d The pin <%s> in the target header file has been mapped before"
"CTRL-40" "%s:%d Target enable pin<%s> is not a output pin"
"CTRL-41" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-41" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-42" "%s:%d Can't assign the value[%c] to the input pin[%s], ignore this command!!"
"CTRL-42" "%s:%d Can't assign the value[%c] to the output pin[%s], ignore this command!!"

"CTRL-43" "%s:%d Unrecognized mask mode<%s>"
"CTRL-44" "Parsing file:%s"
"CTRL-45" "%s=> %s"
"CTRL-46" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-47" "%s:%d The \"
"CTRL-48" "%s:%d Can't assign the value[%c] to the input pin[%s], ignore this command!!"
"CTRL-49" "%s:%d Can't assign the value[%c] to the output pin[%s], ignore this command!!"
"CTRL-50" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-50" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-51" "%s:%d Can't find the pin <%s[%d]> in the target header file"
"CTRL-51" "%s:%d Can't find the pin <%s> in the target header file"
"CTRL-52" "%s:%d The \"
"CTRL-53" "%s:%d Can't assign the value[%c] to input pin[%s], ignore this command!!"
"CTRL-54" "%s:%d Can't assign the value[%c] to output pin[%s], ignore this command!!"
"CTRL-55" "%s:%d Can't find the reference signal <%s> in the header file"
"CTRL-56" "%s:%d Can't find the reference signal <%s[%d]> in the header file"
"CTRL-57" "%s:%d Unrecognized reference signal trigger mode<%s>"
"CTRL-58" "%s:%d Invalid trigger type of the referenced signal<%s> with RZ timegen."
"CTRL-59" "%s:%d Invalid trigger type of the referenced signal<%s> with RO timegen."
"CTRL-60" "%s:%d Timegen SBC is not supported on the referenced signal"
"CTRL-61" "%s:%d Can't find the pin <%s[%d]> in the header file"
"CTRL-61" "%s:%d Can't find the pin <%s> in the header file"
"CTRL-62" "%s:%d Mask string length doesn't match the mask period"
"CTRL-63" "Parser error:%s"
"CTRL-65" "The sequence length of [%s] doesn't match the golden FTL file"
"CTRL-66" "Sequence mismatch [%d]:\n\tGolden:[%s] <=> Target:[%s]"
"CTRL-67" "Pin type mismatch [%d]:\n\tGolden:[%s] <=> Target:[%s]"
"CTRL-68" "Checking pattern error: %s"
"CTRL-69" "Cycle number must greater than 0 at %s:%d"
"CTRL-70" "%s: Unexpected control file error: %s: %s"
"CTRL-71" "Conditional mask does not support the scan ftl pattern"

## 5.3 Messages Related to FTL

"HDR-1" "Can't move timegen from %d to %d"
"HDR-2" "The pin:%s has been declared before"
"DATA-1" "%s: Syntax error: %s"
"DATA-2" "%s: TIMEUNIT must be defined in the ftl header section"
"DATA-3" "%s: CYCLE must be defined in the ftl header section"
"DATA-4" "pin:%s[%d] referenced in the SEQUENCE, but not yet declared before"
"DATA-4" "%s:%d SEQUENCE declaration redefined"
"DATA-5" "pin:%s Referenced in the SEQUENCE, but not yet declared before"
"DATA-6" "%s:%d Pattern width mismatch with the previous declared SEQUENCE"
"DATA-7" "%s:%d Pattern width mismatch with the previous declared SEQUENCE"
"DATA-8" "%s:%d Pattern width mismatch with the previous declared SEQUENCE"
"DATA-9" "%s:%d Pin:%s[%d] is not declared previously"
"DATA-9" "%s:%d Pin:%s is not declared previously"
"DATA-10" "%s:%d The value of scan pin:%s on the initial pattern is not set to '-'"
"DATA-11" "%s:%d Scan chain<%s> pattern length is not %d"

## 5.4 Messages Related to Simulation Mismatch File

"OUT-1" "Can't open the out file <%s> for reading"
"OUT-2" "Can't find the mismatch signal<%s> in the out file at line <%d>"
"OUT-3" "Syntax error at line <%d>"