White Paper

MorphIO: An I/O Reconfiguration Solution for Altera Devices

Introduction

Altera developed the MorphIO software to help designers use the I/O reconfiguration feature in Altera® devices. It is written in tool command language (Tcl) and thus can be used as an extension to the Quartus® II software. The MorphIO software supports all current Altera devices that support the `CONFIG_IO` instruction. Beginning with Stratix™ and Cyclone™ devices, Altera introduces I/O standard setting reconfiguration through the Joint Test Action Group (JTAG) boundary-scan test (BST) chain. The JTAG chain can update the I/O standard for all input, output, and bidirectional pins any time before or during user mode. The I/O reconfiguration feature enables:

- *JTAG Testing Before Configuration*: When pins drive or receive signals from other devices on the board using voltage-referenced standards, designers can use this feature for JTAG testing before configuration. Since the device may not be configured before JTAG testing, the I/O pins may not be configured for the appropriate electrical standards for chip-to-chip communication. Programming those I/O standards via JTAG allows designers to fully test the I/O connection with other devices.
- *Real-time I/O Standard Reconfiguration in User Mode*: Without reconfiguring the entire device, designers can use this feature to change, on the fly, the I/O standard of any input, output, or bidirectional pin to other I/O standards supported by that particular pin on that device. This feature includes changing a single-ended I/O standard to a differential I/O standard or vice versa.
- *Real-Time Drive Strength Reconfiguration in User Mode*: Designers can use this feature to change the LVTTL, LVCMOS, TTL, and CMOS drive strength of any output pin during user mode, on the fly, without reconfiguring the entire device. This feature enables the designer to control overall power consumption by increasing and decreasing the drive strength.

I/O Configuration Shift Register

The I/O configuration shift register (IOCSR) is a long shift register implemented in Stratix and Cyclone devices to facilitate the I/O reconfiguration feature. Each I/O element (IOE) contains a number of configuration bits that control the IOE characteristics. In an IOE, the configuration bits are connected to a series of registers. By linking the registers in all adjacent IOEs, the IOCSR is formed. Thus, designers can achieve I/O reconfiguration by shifting new configuration data into the IOCSR to update the I/O standard settings.

➢ The IOCSR is not the same as the boundary-scan register used in boundary-scan testing. Also, the new data shifted in overwrites the current I/O standard settings. Any invalid settings in the new data can severely damage the device and lead to system failure.

CONFIG_IO JTAG Instruction

Designers can access the IOCSR by using the `CONFIG_IO` JTAG instruction via the JTAG BST chain. The instruction code is `00 0000 1101` (binary) or `00D` (hexadecimal) for Stratix and Cyclone devices (refer to the device datasheet). When it is executed, the `TDI` pin connects to the beginning of the IOCSR and the `TDO` pin connects to the end. All I/O pins are tri-stated and the `nSTATUS` pin drives low. The `CONF_DONE` pin is not affected. If the device is in user mode, it will retain the configuration data and even the user data.

Through the `SHIFT-DR` state, the new content for the IOCSR can be shifted in, while the old contents are shifted out through `TDO`. Once content shifting is complete, advance the TAP controller to the `UPDATE_DR` state and then back to the `IDLE` state to update the new data. The I/O pins are tri-stated throughout these processes and will remain tri-stated until a new JTAG instruction is issued to enable the new configuration on the I/O pins. If no new JTAG instruction is planned, the designer needs to perform a cycle through the `RESET` state and back to the `IDLE` state to enable the new configuration on the I/O pins.

The `nSTATUS` pin goes low after the `CONFIG_IO` instruction is executed and remains low throughout the entire reconfiguration process, even after a new JTAG instruction is issued. It only goes high when the cycle through the `RESET` state and back to the `IDLE` state is performed by the designer.

The `CONFIG_IO` instruction can be executed before, during, and after configuration (i.e., user mode). If it is executed before configuration, the device behaves as stated above. If it is executed during configuration, the configuration process stops and the `nSTATUS` pin is held low to reset the configuration device. The designer then needs to reconfigure the whole device from scratch. When executing in user mode, the device behaves as stated above. All operations stop within the device upon executing of the instruction due to the tri-stated I/O pins. However, user-mode behavior resumes after the I/O reconfiguration is complete.

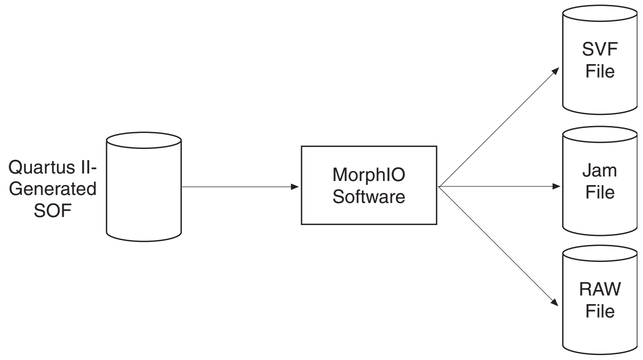
The third attribute stated above enables real-time I/O standards reconfiguration in user mode without losing both configuration data and even user data. For example, if an externally clocked counter is implemented in the device, it stops counting when the `CONFIG_IO` instruction is executed. This action occurs because the I/O pins are being tri-stated and thus effectively cutting off the external clock source from the core. After I/O reconfiguration is complete, the counter continues to count from where it stopped.

The MorphIO Software

The MorphIO software uses the Quartus II-generated SRAM Object File (**.sof**) as the input file. The current release supports Serial Vector Format File (**.svf**), Jam^{TM} File (**.jam**), and raw binary file (**.raw**) format as output files. The default output file is SVF, but can be customized to output any format stated above. The output files in both SVF and Jam formats contain the necessary routines to reconfigure Altera device I/O pins in real time. Designers can use these two formats without any modification.

The output file in RAW format is designed for users whose equipment does not support SVF or Jam formats, and can be used in an embedded environment. The RAW format does not contain any routine. It only contains the new IOCSR data, in packed binary form, that is to be shifted into the IOCSR during I/O reconfiguration. Therefore, the designer must develop his own routine to reconfigure Altera device I/O pins. The routine algorithm is very simple and will be discussed in the next section. Figure 1 shows the MorphIO work flow.

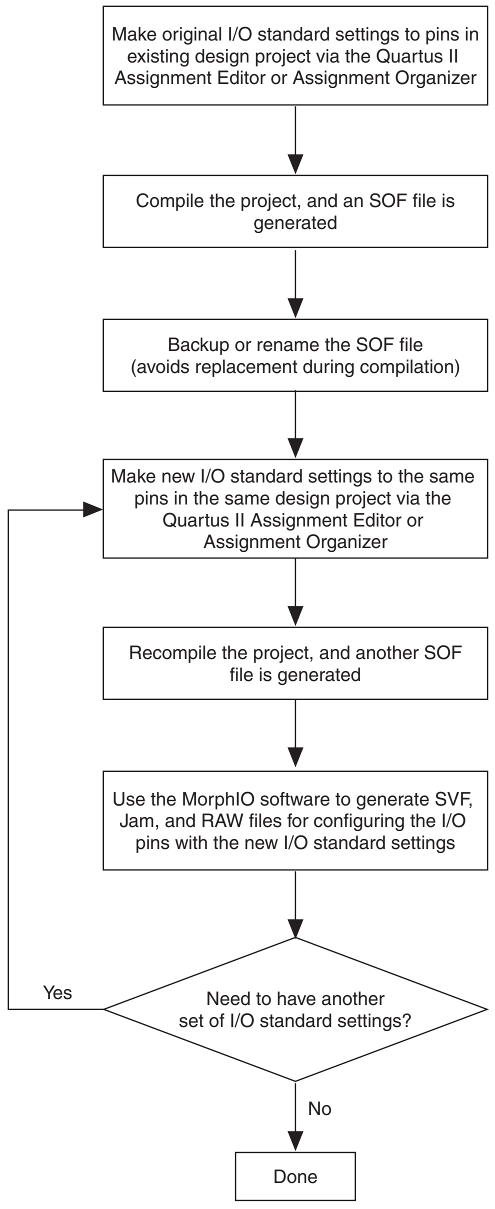
Figure 1. MorphIO Work Flow



Using MorphIO in a Design Flow

Figure 2 shows how to use MorphIO in a design flow.

Figure 2. MorphIO Software Design Flow


Make original I/O standard settings to pins in existing design project via the Quartus II Assignment Editor or Assignment Organizer
Compile the project, and an SOF file is generated
Backup or rename the SOF file (avoids replacement during compilation)
Make new I/O standard settings to the same pins in the same design project via the Quartus II Assignment Editor or Assignment Organizer
Recompile the project, and another SOF file is generated
Use the MorphIO software to generate SVF, Jam, and RAW files for configuring the I/O pins with the new I/O standard settings
Need to have another set of I/O standard settings?
Yes
No
Done

Designers make the original I/O standard settings to the appropriate pins in the project via the Quartus II Assignment Editor or Assignment Organizer. These I/O standard settings are what will be configured into the device and initially used. The project is then compiled, and an SOF is produced. The SOF should be renamed or backed up since the SOF with the same project name will be rewritten during compilation.

Using the Quartus II Assignment Editor or Assignment Organizer, make the new I/O standard settings. This new set of I/O standard settings is what will be reconfigured into the device. Compile the project, and another SOF is generated. Using the SOF as the input file to the MorphIO software, the desired output file is generated. This process is repeated for every new set of I/O standard settings required. A new SOF is required even if the designer wishes to reconfigure only a single pin.

How to Use MorphIO

The MorphIO syntax is as follows:

morphio <*SOF input file.sof*> <*output filename*> ?options?

where

<*SOF input file.sof*> is the SOF input file together with the path leading to it

<*output filename*> is the output filename together with the path leading to it

?options?

-s or –S	generate SVF file
-j or –J	generate Jam file
-r or –R	generate RAW file

Since the MorphIO software appends the appropriate extension to the output files, applying an extension to the <*output filename*> field is not necessary:

- The generated SVF file will be named <*output filename*>.svf
- The generated JAM file will be named <*output filename*>.jam
- The generated RAW file will be named <*output filename*>@< *number of bits*>.raw

The <*number of bits*> field in the generated RAW file signify the length of the IOCSR. Since a byte (8 bits) is the smallest unit a computer can write to the file system, the <*number of bits*> field serves as the exact number of bits the designer needs to clock into the IOCSR during I/O reconfiguration. Any variation in the number of bits clocked in will damage the device. This field is not found in the output filename of SVF and Jam format because the routines inside already specify the exact number of bits to be clocked.

Examples

The following example generates a default output file **design1.svf.**

morphio design1.sof design1

This example generates **dsgn2.jam** and **dsgn2@13646.raw** as output files. Using optional switches will override the default settings (i.e., generating an SVF file). Also, the output filename does not need to be the same as the input SOF filename:

morphio design2.sof dsgn2 –j –r

This example generates **test.svf** and **test.jam** as output files in their respective paths. Because Tcl handles paths differently, a forward slash is used in the path versus a backslash:

morphio "c:/My Design/design3.sof" "d:/Test Design/test" –s -j

Using the SVF Output File

The following code shows the contents of the generated SVF file:

```
TRST ABSENT;
ENDDR IDLE;
ENDIR IDLE;
STATE IDLE;
SIR 10 TDI (00D);
SDR 13646 TDI (...);
STATE RESET;
STATE IDLE;
```

The first line of code denotes that the `TRST` pin is unused. The two subsequent lines prompt the SVF parser to bring the JTAG state machine back to the `IDLE` state after every instruction and data. The I/O reconfiguration process starts in the `IDLE` state and proceeds to execute the `CONFIG_IO` instruction (with code `00D`). This instruction is followed by the long IOCSR data being shifted into the device (shown by "…"). Since there are no more JTAG instructions that follow, a cycle through the `RESET` state and then back to `IDLE` state is executed.

The algorithm is very straight forward. To perform I/O reconfiguration, the designer can copy and embed this routine into any existing SVF-based test routine.

Using the Jam Output File

The following code shows the contents of the generated SVF file:

```
NOTE MAX_FREQ "10000000";
ACTION CONFIG_IO = EXECUTE;
PROCEDURE EXECUTE;
BOOLEAN X = 0;
DRSTOP IDLE;
IRSTOP IDLE;
STATE IDLE;
IRSCAN 10, $00D;
DRSCAN 13646,
$...;
STATE RESET;
STATE IDLE;
EXIT 0;
ENDPROC;
```

The first line of code denotes that the recommended maximum frequency for `TCK` is 10 MHz. The action name for the I/O reconfiguration procedure is `CONFIG_IO`. The rest is the same as stated in "Using the SVF Output File" on page 6.

Again, the designer can copy and embed this procedure into any existing Jam-based test routine to perform I/O reconfiguration. To use it with the Jam Player, type:

```
jam –aCONFIG_IO <filename>.jam
```

Using the RAW Output File

The data in the RAW output file is in packed binary format, meaning that each bit in the file is IOCSR data. The RAW output file is generated in little endian order. Therefore, the least significant bit (LSB) of the first byte read from the file is the IOCSR LSB. To reconfigure I/O pins, the IOCSR LSB is the first bit-shifted into the IOCSR.

The <*number of bits*> field in the filename should be considered when the last byte is read from the file. It is the LSB portion, not the most signigicant bit (MSB), of the last byte to be shifted into IOCSR. For example, if there are three bits remaining that need to be shifted into the IOCSR, the designer should read the three LSBs of the last byte to shift them into the IOCSR.

Executing the MorphIO Software

The Quartus II software contains a built-in Tcl interpreter and can be used to run the MorphIO software. Designers can use the MorphIO software:

- Interactively in the Quartus II Tcl Console window
- Interactively in any Tcl interpreter shell such as Tclsh and Wish
- DOS or UNIX command-line prompt via the **quartus_cmd.exe** file
- DOS or UNIX command-line prompt via any Tcl interpreter shell

Using MorphIO Interactively

In the Quartus II Tcl Console window or any Tcl interpreter shell, type:

```
source morphio.tcl
morphio <SOF input file.sof> <output filename> ?options?
```

The designer can also embed the above lines in a batch job Tcl script to automate generations.

Using MorphIO via Command Line Prompt

The designer must first create a Tcl script containing the above text. Then, type the following:

`quartus_cmd -f <script.tcl>` if using quartus_cmd.exe,
or
`tclsh <script.tcl>` if using any Tcl interpreter shell.

The designer can also embed it in a batch job Tcl script to automate generations.

Conclusion

With its ease-of-use, the MorphIO software is designed to let designers exploit the powerful I/O reconfiguration feature in Stratix and Cyclone devices.

101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
www.altera.com