Stratix FPGA Configuration

1. Summary

This document discusses the various options for configuring Altera Stratix FPGAs, as used in the SCUBA-2 multichannel electronics.

2. References

'Stratix FPGA Family Data Sheet', DS-STXFAMLY-3.0, Altera Corp.

'AN 208: Configuring Stratix & Stratix GX Devices', AN-208-2.2, Altera Corp.

'ByteBlasterMV Parallel Port Download Cable Data Sheets', DS-BYTBLMV-3.3, Altera Corp.

'AN 217: Using Remote System Configuration with Stratix & Stratix GX Devices', AN-217-2.1, Altera Corp.

3. Introduction

The Altera Stratix FPGAs to be used in the design of the multichannel electronics use SRAM based cells to store the required configuration for the device. This means that the configuration information is lost when the powered is turned off and needs to loaded into the device when the power is turned on.

4. Configuration options

4.1 General

Stratix devices can be programmed in any of five different basic configuration schemes as shown in table 1 (see Altera Stratix data sheet and application notes AN-208).

Configuration Scheme	Data Source
Configuration device	Enhanced or EPC2 configuration device
Passive serial (PS)	ByteBlasterMV or MasterBlaster download cable or serial data source
Passive parallel asynchronous (PPA)	Parallel data source
Fast passive parallel (FPP)	Parallel data source
JTAG	MasterBlaster or ByteBlasterMV download cable or a microprocessor with a Jam or JBC file

Table 1

The choice of scheme is dependent on the particular requirements of a given design, such as the time required to configure the device(s).

4.2 Hardware programmers

During the development phase configuration is usually carried out using a hardware programmer, such as the Altera ByteBlasterMV (see ByteBlasterMV Parallel Port Downlaod Cable Data Sheets). This allows configurations to be easily updated and tested.

The programmer is plugged into a header on the card which connects to the device(s) to be programmed. Either the PS or JTAG modes can be selected depending on how the header is wired to

the device(s). JTAG mode allows the SignalTap embedded logic analyser feature of the Stratix device to be used. SignalTap allows internal signals to be viewed at normal operating speed without having to bring signals out to I/O pins.

4.3 Configuration Devices

Altera supply standard and enhanced configuration devices which can be use to configure an FPGA. The configuration devices basically consist of a FLASH memory with a built in controller. The controller reads the configuration information from the FLASH memory and transfers it to the FPGA.

The configuration devices themselves can be programmed via a JTAG interface either by a hardware programmer, e.g. ByteBlasterMV, or other JTAG controller, e.g. embedded processor.

4.4 JTAG

In JTAG mode a number of devices can be linked together on the JTAG bus to form a daisy chain. Any device(s) in the chain can be selectively configured as required. A suitable controller is needed to drive the JTAG bus and transfer the configuration information to the relevant device(s).

Devices can be programmed in JTAG mode either by a hardware programmer, e.g. ByteBlasterMV, or other JTAG controller, e.g. embedded processor. Altera provide software to implement a JTAG controller to run on a standard PC or an embedded processor. Two packages are provided: i) the JAM player software which has been adopted as a JEDEC standard for in system programmability; and ii) the JRunner software which is proprietary JTAG controller. Both packages need to be modified to interface to the particular set of hardware in use.

4.5 Serial and parallel modes

The serial and parallel modes are proprietary to Altera and as the names suggest use serial and parallel data transfer, respectively, to configure the FPGAs. Multiple devices can be daisy chained together but in this mode all devices in the chain must be configured sequentially.

For the serial mode the data source can be an Altera hardware programmer, Altera configuration device, or other serial source such as an embedded processor. Parallel mode is similar except that this mode is not supported by Altera hardware programmers.

5. Estimated configuration times

Some of the timings shown in this section are based in part on measurements of the configuration time for an APEX EP20K100 device using a JAM player running on a 450MHz Pentium III PC in conjunction with a ByteBlasterMV hardware programmer. Other timings are based on measurements, using the same PC, of the proposed SCUBA-2 protocol (see document).

Device	**Configuration file size (bits)**	**Configuration time (s)**					
		JAM player + ByteBlasterMV *	**JAM player + standard SCUBA-2 protocol ***	**JAM player + simple protocol ****	**SCUBA-2 fibre optic limit**	**JTAG/ PS limit**	**FPP limit**
EP20K100	985000	5	758	16	0.1	0.1	0.01
EP1S10	3534640	18	2722	57	0.4	0.4	0.04
EP1S25	7894144	40	6078	126	0.8	0.8	0.08
EP1S40	12389632	63	9540	198	1.2	1.2	0.12

* - based on measured timings on a 450MHz Penitum III PC
** - using standard SDSU protocol

It is clear that running the JAM player on the PC is not a satisfactory option for loading configurations whenever the electronics is powered on or reset. However, it may be acceptable for updating configurations on an occasional basis.

The reason that the JAM player is so slow when run on the PC has to do with the details of the JAM player operation. The problem is that the JAM player can only ever send/receive a single bit of information to/from the PC. The overheads associated with this process e.g. packet size on the data link, result in the whole process being very slow.

6. Proposed configuration scheme

To get round the problems associated with the initial preferred scheme of running the JAM player on the PC, the following configuration scheme is proposed.

Each Stratix device will have an associated EPC16 configuration device which will download the configuration data into the Stratix device at power on or system reset. The EPC16 can transfer the configuration data in either FPP (preferred) or PS modes. This gives a configuration time of less than one second for all devices.

The EPC16 configuration devices, and indeed the Stratix devices, can be programmed via a JTAG bus which connects to all EPC16 and Stratix devices in a daisy chain fashion. Active bus switches e.g. SN74CBTLV125, on the bus backplane are used to bypass empty slots and maintain the continuity of the JTAG chain. When a card is plugged into the backplane the JTAG bypass switch is disabled and the JTAG chain flows through the devices on the card.

The JTAG bus can be controlled in two ways: (i) by means of a ByteBlasterMV connected to a JTAG header on the face plate of the clock card, and (ii) by the Stratix device on the clock card.

Option (i) is typically used during development and allows the configuration of any device(s) on any card to be updated. It would also be possible to use switches on each card to allow the JTAG bus to be locally controlled via a ByteBlasterMV connected to a separate header on each card. However, there is no real advantage in doing this and indeed there are two main disadvantages: (i) unless the switch on the card can automatically detect whether or not the ByteBlasterMV is attached, there is always the possibility that a switch is left in the wrong position leaving that card disconnected from the main JTAG chain, and (ii) the ByteBlasterMV connection has to be moved around depending on which card is being worked on.

Option (ii) is used for updating the configuration information stored in each EPC16 device via the fibre optic link connected to the data acquisition PC. This mode utilises the remote configuration capability of the Stratix device (see application note AN-217). The clock Stratix device is setup in remote configuration mode. At power up a default, or boot, configuration is loaded into the clock card Stratix device by the EPC16. This configuration checks the fibre optic link for configuration update requests. If such a request is received the configuration data is downloaded from the PC into the SRAM on the card, the Stratix device takes control of the JTAG bus, and then a NIOS processor is used to run the JRunner or JAM player software to configure the relevant device(s) using the data stored in the SRAM. Once all the configuration requests are complete the PC sends a start request which the Stratix device uses to force the EPC16 to load the main operating configuration and the system is then ready for normal operation. The EPC16 has a 64K byte area of FLASH memory which can contain the code needed by the NIOS processor, in our case the JRunner or JAM player software. We can choose whether or not implement the FLASH memory interface on other cards as necessary, e.g. it may be useful for storing filter weighting constants on the readout card.

Depending on the resources needed for the boot and operating configurations, these could be combined into a single configuration if desired. However, there is some benefit in keeping the two configurations separate as they can then be treated as independent firmware tasks.

This configuration scheme is versatile in that we could choose to have a boot configuration such that a JAM player on the PC is used to update the configuration of devices as required, assuming the time required for the update was acceptable.

7. Version control

It is important that all cards of the same type, whichever electronics subrack they are mounted in, use the same version of the FPGA configuration. Provision must be made to ensure that this is case. In addition it must be possible to check the configuration version for each card to check that it is correct.

If we use the mode where the clock card boot configuration uses a NIOS processor and JRunner software to configure other devices, the same software could be used to interrogate the JTAG chain and determine the version numbers of the various FPGA configurations.