Clock Card Dual Configuration Tech-Note

1. Summary

This document discusses the configuration scheme to be adopted for the Clock Card (CC) FPGA in the Multi-Channel Electronics (MCE).

2. References

[1] 'Multichannel Electronics Block Diagram', SC2_ELE_S560_001.
[2] 'Stratix FPGA Configuration', SC2/ELE/S500/21.
[3] 'MCE Engineering Description', SC2/ELE/S565/000
[4] 'AN 217: Using Remote System Configuration with Stratix & Stratix GX Devices', AN-217-2.1, Altera Corp.
[5] 'Multichannel Electronics Requirements and Recommendations', SC2/ELE/S500/11.
[6] 'FPGA Configuration Block Diagram', SC2_ELE_S560_003.
[7] 'Clock Card Configuration Scheme Schematic Capture' SC2/ELE/S565/103/001
[8] 'AN 208: Configuring Stratix & Stratix GX Devices', AN-208-2.2, Altera Corp
[9] 'Stratix FPGA Family Data Sheet', DS-STXFAMLY-3.0, Altera Corp.
[10] 'ByteBlasterMV Parallel Port Download Cable Data Sheets', DS-BYTBLMV-3.3, Altera Corp.
[11] 'Using the JAM Language for ISP & ICR via an Embedded Processor' Altera Application Note 088
[12] 'CC Dual-EPC16 Configuration Block Diagram', SC2_ELE_S565_003_000

3. Introduction

Stratix FPGAs can be configured in a variety of different ways, as discussed in [2]. Using the modes that are available to us, it is our intention to implement a configuration scheme which is robust – one with which the MCE can always recover from an internal hardware glitch or firmware bug.

This document details the particular configuration scheme which has been chosen for the CC. We intend on using two configuration devices on the CC – one to hold an application configuration for normal operation at JCMT, and another to hold a stable factory configuration which the CC FPGA can revert to if the application configuration encounters difficulties.

4. Clock Card Dual-Configuration Capability

The intention for the CC is for it to have a non-corruptible factory configuration which it can always fall back upon to reload a working application configuration from its RT-Linux PC. There are three ways to achieve this goal:

1) Run the CC Stratix FPGA in Remote FPP Configuration Mode [4], and use the PGM[2..0] signals with external some logic to switch between the two EPC16s.
2) Run the CC Stratix FPGA in Local FPP Configuration Mode [4], still using the PGM[2..0] signals and external logic to switch between the two EPC16s.
3) Run the CC FPGA in Standard or Stratix Remote/Local [4] FPP Configuration Mode, and implement external logic which is independent of the PGM[2..0] signals to switch between the two EPC16s.

The Stratix FPGAs have a Remote/Local Configuration Mode [4] using a single EPC16 which implements much of what we want for a dual factory/application configuration, except the factory configuration page in a single EPC16 may still become corrupted when programming one of the

application pages. The first two solutions, above, hi-jack the PGM[2..0] signals to switch between two EPC16s. This would work as follows:

4.1 Using PGM[2..0] Signals to Select Between EPC16s

The CC FPGA would be configured through hardware to be in Remote or Local FPP system configuration mode (i.e. RUnLu = 1 for Remote or 0 for Local, MSEL[2..0] = 100; refer to [8] p.3). In Remote/Local FPP mode, the CC can select which of 8 configurations to load using its PGM[2..0] pins. The configuration interface between the FPGA and the EPC16 includes the following pins:

EPC16	Stratix
DATA[7..0]	DATA[7..0]
DCLK	DCLK
nCS	CONF_DONE
nINIT_CONF	nCONFIG
OE	nSTATUS
PGM[2..0]	PGM[2..0]

For details on the requirements for the intervening logic between the CC FPGA and its two configuration devices on the pins above, see [4] and [8].

We would add some external logic to force the use of the factory EPC16 when the Stratix FPGA requests configuration from page 0 (the default or 'factory' page). Other pages would switch to the application EPC16. The Stratix FPGA contains all the mechanisms to support multiple configurations, including a programmable user watchdog timer. This watchdog timer must be reset in the main operating configuration at a rate determined by the factory configuration. If the watchdog timer is not reset and allowed to trigger, either due to a firmware fault or intentionally, the Stratix will request reload of the factory configuration from the EPC16 into the clock card Stratix device.

Choosing Remote Configuration Mode (RUnLu = 1) causes the factory configuration to load on power-up. The factory configuration would contain the configuration engine and other essential services. When the CC receives a command from the RT-Linux computer to switch to the application configuration, the factory firmware enables the watchdog timer and triggers reconfiguration of the Stratix FPGA using the application configuration.

Choosing Local Configuration Mode (RUnLu = 0) causes the application configuration to load on power-up, saving the time to load first the factory then the application configuration. The idea behind this is that the application configuration would be a super-set of the factory configuration, with the factory configuration being just a fall-back. Note that the Stratix watchdog timer is disabled in Local Configuration Mode, and the factory configuration will only be loaded if the application configuration is corrupted (i.e. generates a CRC error). If the configuration engine (which permits firmware upgrades from the RT-Linux computer) is not contained in the application configuration, then Local Configuration Mode cannot be used.

In order to keep the paging capability of Remote/Local Configuration Mode, we must retain the selection of EPC16 pages using the FPGA's PGM[2..0] signals. For this to work, the factory configuration must fit in one page of the EPC16, and the application configuration must fit within the remaining seven pages. If a large Stratix FPGA is used (so that its configuration cannot fit within one page of the EPC16), then just the FPGA's PGM0 signal may be used to switch between two EPC16s which have their PGM[2..0] all tied low, foregoing the paging capability.

4.2 Using Dedicated Logic to Switch Between EPC16s (refer to [12])

The third solution to achieve a non-corruptible factory configuration adopts an EPC16-switching strategy that is completely independent of the PGM[2..0] signals and the Stratix FPGA's remote-configuration capabilities. The Stratix FPGA may be configured in hardware (via is RUnLu and MSEL[2..0] signals) for Standard or Remote/Local Configuration Mode [4] without any difference to the EPC16-switching circuitry. (In fact, this solution works with non-Stratix FPGAs which may not have Remote/Local Configuration Mode.) The factory and application EPC16s may be independently

partitioned (in terms of pages, FLASH usage for NIOS or other processors, etc.) however the FPGA design sees fit. The strategy switches whole EPC16 devices, and is essentially transparent to the FPGA except that the FPGA can choose which EPC16 its next reconfiguration data comes from.

Document [12] illustrates one way to implement the dual-configuration mechanism. In that document, the circuit provides the following capabilities:

- The CC FPGA's "program" may initiate reconfiguration into either factory or application mode. Using the Stratix FPGA's built-in Remote/Local Configuration Mode [4] capabilities, a specific page from the active EPC16 may be selected.
- A '**configuration reset**' (initiated by the Power Card via the BRst signal, see [3]) will always reconfigure the CC FPGA from the factory configuration device.
- An INIT_CONFIG JTAG command to the application configuration device will force the CC FPGA into application mode.

4.2.1 Circuit Design

The switching of the FPGA between two EPCs is straightforward with the exception of the OE/nSTATUS signal. This signal is an open-drain signal, and it is also bidirectional. The EPC will hold OE low to signal to the FPGA that it is about to be configured. The FPGA may also pull nSTATUS low to indicate to the EPC that it wishes to be configured. Additionally, an external device may hold OE low to defer the EPC's configuration sequence.

The use of FETs preserves the open-drain nature of the OE/nSTATUS signal. If we consider the paths from the Q/nQ outputs of the flip-flop to the OE pins for the moment, the inactive EPC must be held off by holding its OE pin low. This is accomplished by connecting an open-drain FET, gated by the Q and nQ outputs of the flip-flop, to each EPC16's OE pin. Additionally, the FPGA must be able, with its nSTATUS signal, to pull down the EPC16s' OE pin, and this is accomplished via the Schottky diodes labelled "optional" (a 74LVC4066 may be used in place of the FET_Schottky diod combination); the EPC that is inactive is already held off and the diodes provide that only the low level provided by the nSTATUS signal reaches the active EPC16's OE pin. Finally, the active EPC (and only the active one) must be able to pull down the FPGA's nSTATUS pin, and the series-pass FETs provide this capability. Think of them as another open-drain device stacked upon the EPC's internal one, similar to a cascade configuration.

Using so-called Digital FETs (e.g. FDC6030N, from Fairchild Semiconductor) having $V_{GS(th)} \leq 1.5V$, the FETs will fully enhance under all conditions in a 3.3V circuit. In fact, they will enhance and bypass the optional Schottky diodes even when it is the FPGA pulling down on nSTATUS, because their body diodes (and/or the optional Schottky diodes) will initially conduct until the FET source becomes low enough that the FET's entire channel enhances. Note that, in order to prevent the circuit from becoming self-latching, the pull-down current must be sunk by either the OE pin of the EPC or the nSTATUS pin of the FPGA. We cannot use tri-state buffers such as a 74xx125/126 because the output low-level (sunk by the buffer's internal low-side FET) will feed back to its input and become self-reinforcing; the circuit will stay in a low state even if both the EPC and FPGA release their OE/nSTATUS pins.

The on-board power-on-reset (POR) circuit is both good practice and required in this design. Originally, the BRst signal (originally an active-low signal) from the Power Card could qualify that all power supply voltages were within specification and stable; the active-low nature of the original BRst signal allowed for reset to be asserted even before the power supply voltages had all come up to full voltage. However, due to noise, isolation, and benign-failure requirements, linear regulators were added to each card. Therefore, the qualification of "power-good" can only be done on-board each card, past the linear regulator. Having an on-board POR circuit also affords some degree of fault-tolerance with respect to the BRst circuits themselves.

Since we are implementing functionality to switch between EPC16s independently of the Stratix FPGA's Remote/Local Configuration feature, we must provide a separate watchdog that is capable of resetting the selection circuitry to a known state (that which selects the Factory EPC). In general, the use of an external watchdog also guarantees watchdog functionality in the case of a mis-configured on-chip watchdog or even catastrophic failure of the FPGA. The watchdog can reset other circuitry

critical to the health of the cryostat, for example, even if the FPGA is destroyed. For an in-depth discussions of the merits of external watchdogs, and of proper watchdog designs, please refer to:

http://www.embedded.com/design_library/OEG20021211S0032
http://www.embedded.com/story/OEG20030115S0042
http://www.embedded.com/design_library/OEG20030220S0037

The 74LVC541 tri-state bus drivers are required because the EPC16s' DATAx signals are always driven.

The DCLK signals may be combined simply using an OR gate, because the inactive EPC idles its DCLK output low.

The PGM[2..0] signals are connected between the FPGA and the EPC16s to support Remote/Local Configuration Mode [4]. If the FPGA is not (MSEL2 = 0) configured for Remote/Local Configuration, then the PGM[2..0] signals from the FPGA are general I/O signals and should be pulled down with resistors strong enough to overcome the FPGA's pull-ups in case they are enabled.

4.2.2 Circuit Operation

A power-on reset or BRst reset will reset the flip-flop to use the Factory EPC. The low-going pulse on the OE/nSTATUS signal will also initiate a configuration sequence.

A private JTAG command to the Application EPC will assert its nINIT_CONFIG signal, setting the flip-flop to use the Application EPC. The FPGA, upon seeing its nCONFIG input become active, request reconfiguration through its nSTATUS and CONFIG_DONE signals. nPOR is not asserted for this type of reconfiguration.

The firmware in the FPGA may initiate a reconfiguration by setting up the desired level on its nEPC_SEL output and asserting the nRECONF output. In combination with the Stratix Remote/Local Configuration Mode, the FPGA firmware designer may reconfigure from page 0 of either the Factory or Application EPC using nRECONF, then reconfigure from another page within the selected EPC using the Stratix Remote Update Control Register. nPOR is not asserted for this type of reconfiguration.

In any case, should the active FPGA configuration not reset the external watchdog timer, the watchdog will emulate a power-on reset and reconfigure the FPGA from page 0 of the Factory EPC. The watchdog is not enabled until the FPGA is configured. Upon being enabled, the watchdog initialises itself to the untriggered state and begins counting up to its designed timeout period, within which the FPGA firmware should start resetting it.