AN 587: DPRIO and Multiple Instances SDI Application

© August 2009 AN-587-1.0

Introduction

This application note discusses how the dynamic reconfiguration controller is used to control multiple instances of the Altera® Serial Digital Interface MegaCore® function. Reconfiguration of SDI instances is simple if your design requires only basic standard. However, complex designs may require multiple dynamic configuration controllers to reconfigure multiple SDI instances across a number of transceiver blocks.

The SDI MegaCore function supports transmitting and receiving uncompressed videos such as SDI standard definition (SD), SDI high definition (HD), and SDI 3 gigabits per second (3G). The multi-rate SDI MegaCore function detects incoming video data rate change, while the dynamic partial reconfigurable I/O (DPRIO) feature of the ALT2GXB_RECONFIG megafunction automatically reconfigures the transceiver and locks to the new video data rate.

The SDI example design discussed in this application note is the Stratix® II GX example design provided with the SDI MegaCore function in the **ip/sdi/example/s2gx_tr** directory. This example design uses two instances of the SDI MegaCore function targeting a Stratix II GX device.

The complexity level of using DPRIO and the implementation of the SDI example design is discussed in the following sections:

- Parameter Settings
- Dynamic Reconfiguration Controller
- User Logic for ALT2GXB_RECONFIG Megafunction
- Merging SDI Receiver and Transmitter in the Same Physical Channel
- Stratix II GX LRIO Resources Limitation

For more information about the Stratix II GX example design, refer to *AN 339: Serial Digital Interface Demonstration for Stratix II GX Devices*. For more information about the SDI MegaCore function, refer to the *SDI MegaCore Function User Guide*.

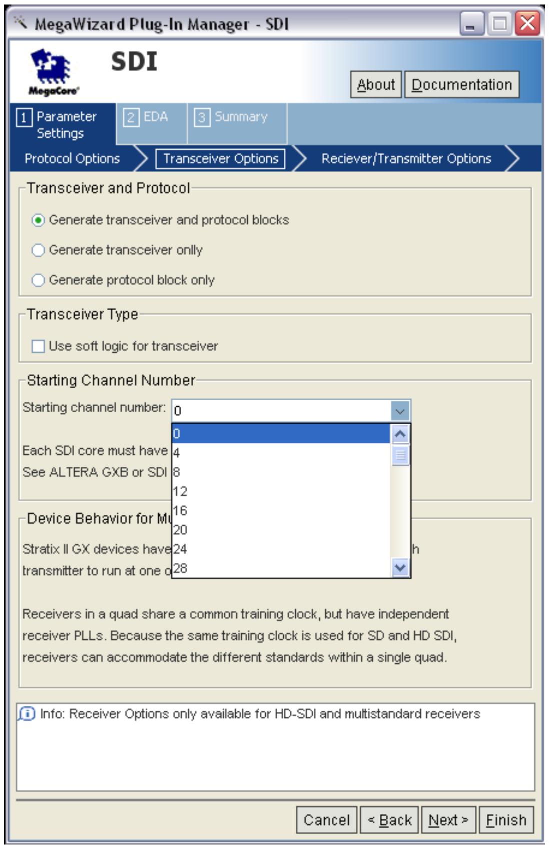
Parameter Settings

The following sections describe how to specify the appropriate parameter settings in the MegaWizard™ Plug-In Manager for the SDI MegaCore function and the ALTGXB_RECONFIG megafunction.

Starting Channel Number

When parameterizing your SDI MegaCore function, you must specify the starting channel number. The setting for the starting channel number ranges from 0 to 156, in multiples of four, refer to Figure 1 on page 2.

Figure 1. The SDI MegaWizard Interface


MegaWizard Plug-In Manager - SDI

SDI

Transceiver and Protocol

● Generate transceiver and protocol blocks

○ Generate transceiver only

○ Generate protocol block only

Transceiver Type

☐ Use soft logic for transceiver

Starting Channel Number

Starting channel number: 0

0, 4, 8, 12, 16, 20, 24, 28

Each SDI core must have

See ALTERA GXB or SDI

Device Behavior for Mu

Stratix II GX devices have … h

transmitter to run at one o

Receivers in a quad share a common training clock, but have independent receiver PLLs. Because the same training clock is used for SD and HD SDI, receivers can accommodate the different standards within a single quad.

Info: Receiver Options only available for HD-SDI and multistandard receivers

To control multiple instances of the SDI MegaCore function with a single dynamic reconfiguration (ALT2GXB_RECONFIG) controller, you must specify each SDI instance with a set of consecutive channel numbers beginning with a unique number in multiples of four. If any two or more instances have the same starting channel number, the Quartus® II software reports an error during compilation.

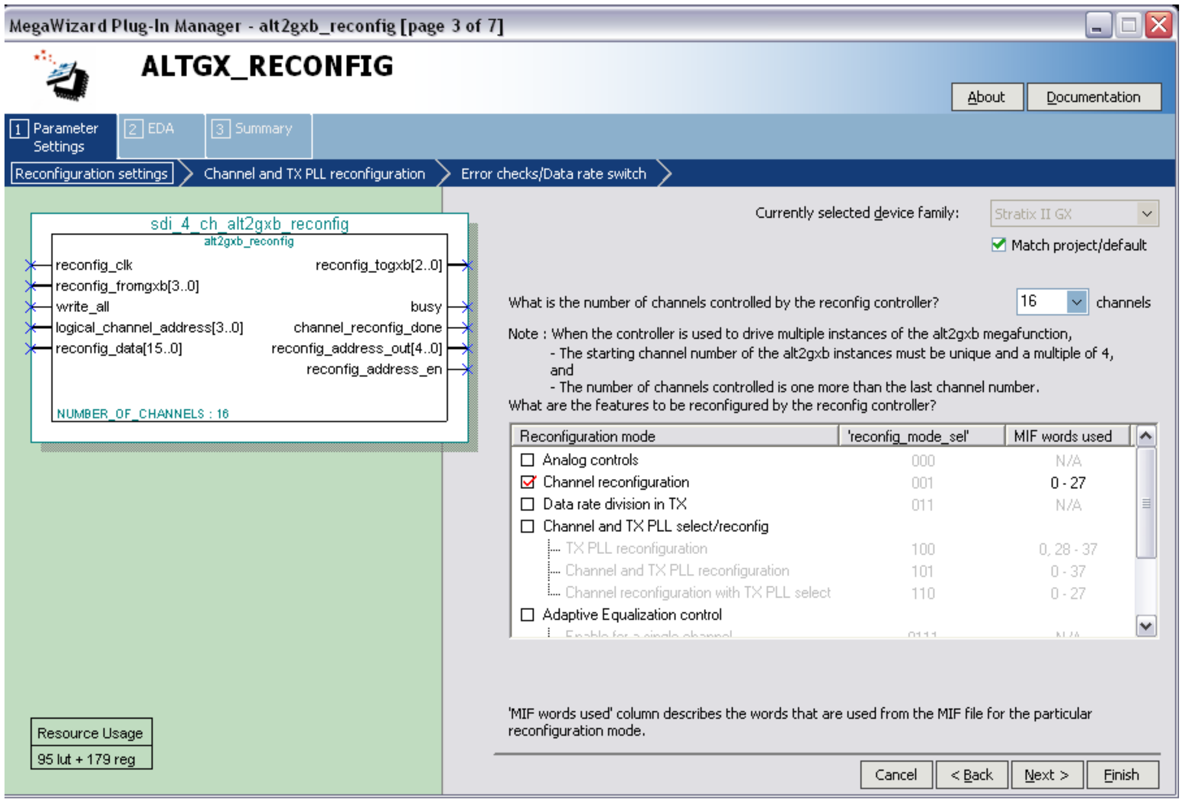
The starting channel number acts as a unique ID representing a specific SDI instance. This starting channel number is important for the ALT2GXB_RECONFIG controller to identify and reconfigure a proper SDI instance via the logical channel address.

ALT2GXB_RECONFIG Megafunction Settings

When instantiating the ALT2GXB_RECONFIG megafunction, you must specify the number of channels based on the requirements of your design. For example, the SDI example design requires 16 channels.

The ALT2GXB_RECONFIG megafunction supports various reconfiguration modes. For the SDI example design, you must select the **Channel reconfiguration** option to enable dynamic reconfiguration, refer to Figure 2 on page 3. The reconfiguration mode you specify for the ALT2GX transceiver inside the SDI MegaCore function and the ALT2GXB_RECONFIG megafunction must be the same.

Figure 2. The ALTGX_RECONFIG MegaWizard Interface



Currently selected device family: Stratix II GX

[✓] Match project/default

What is the number of channels controlled by the reconfig controller? 16 channels

Note : When the controller is used to drive multiple instances of the alt2gxb megafunction,
- The starting channel number of the alt2gxb instances must be unique and a multiple of 4, and
- The number of channels controlled is one more than the last channel number.

What are the features to be reconfigured by the reconfig controller?

Reconfiguration mode	'reconfig_mode_sel'	MIF words used
☐ Analog controls	000	N/A
☑ Channel reconfiguration	001	0 - 27
☐ Data rate division in TX	011	N/A
☐ Channel and TX PLL select/reconfig		
TX PLL reconfiguration	100	0, 28 - 37
Channel and TX PLL reconfiguration	101	0 - 37
Channel reconfiguration with TX PLL select	110	0 - 27
☐ Adaptive Equalization control		
Enable for a single channel	0111	N/A

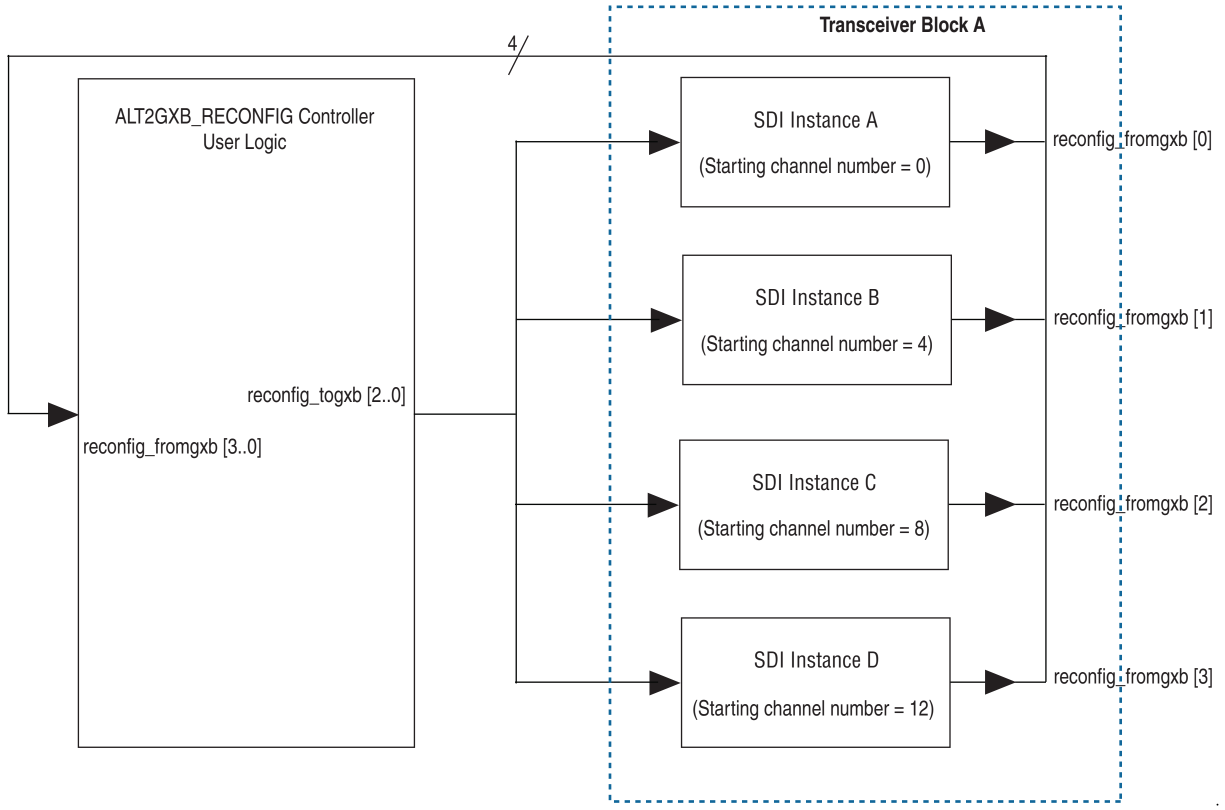
'MIF words used' column describes the words that are used from the MIF file for the particular reconfiguration mode.

Dynamic Reconfiguration Controller

A single dynamic reconfiguration controller (`sdi_tr_reconfig_multi`) can control a maximum of four SDI duplex instances from the same transceiver block or from multiple transceiver blocks. Assign the starting channel number based on the bits of the data bus, refer to Figure 3 and Figure 4.

Figure 3 shows how the ALT2GXB_RECONFIG controller controls and configures four SDI duplex instances in the same transceiver block.

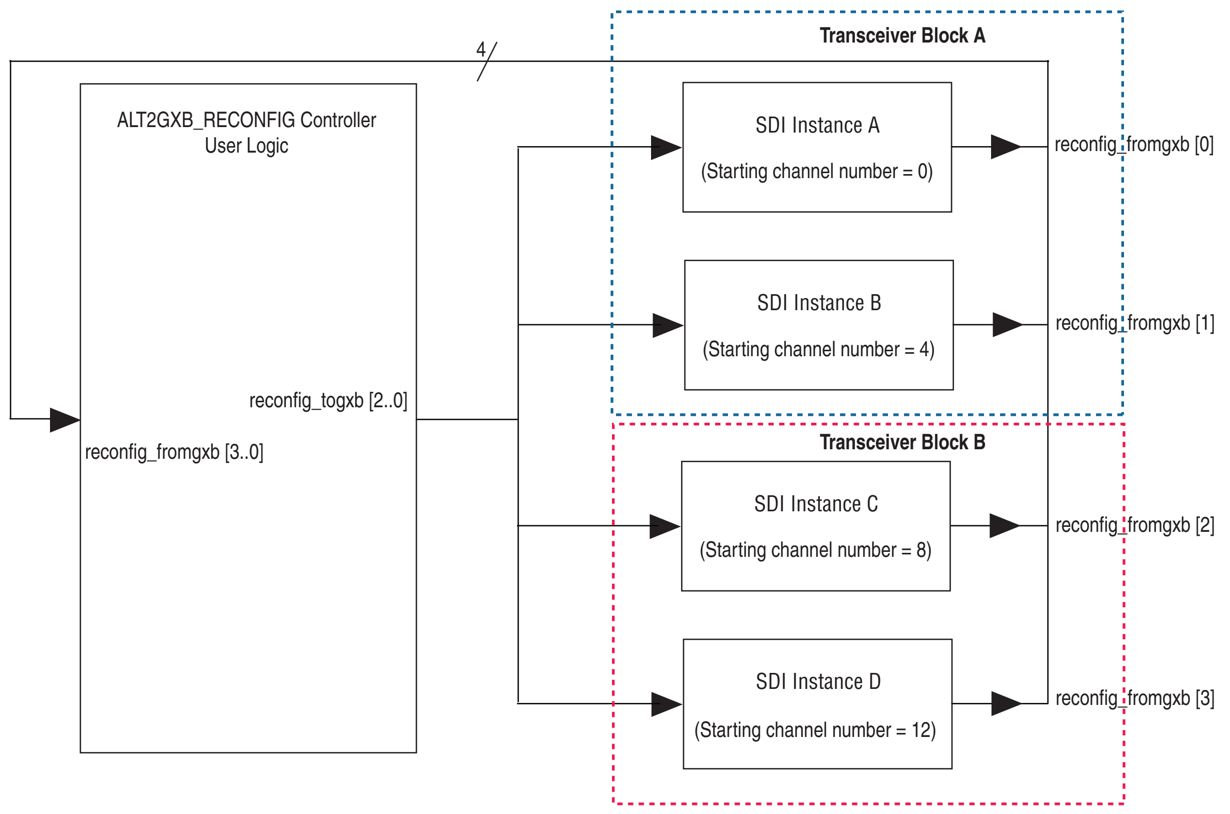
Figure 3. Connecting ALTGXB_RECONFIG Controller to SDI Instances From Single Transceiver Block



The number of ALT2GXB_RECONFIG controllers required in a design depends on the number of SDI duplex instances used. The starting channel number must be unique if you want to connect the SDI instances to one ALT2GXB_RECONFIG controller. However, you can reuse these starting channel numbers if you are connecting the SDI instances to a second or third ALT2GXB_RECONFIG controller. For example, if your design has 8, 12, 16, or 20 SDI instances, reuse the same pattern as illustrated in Figure 3.

Figure 4 shows how a single ALT2GXB_RECONFIG controller is used to control and configure duplex SDI instances from two different transceiver blocks.

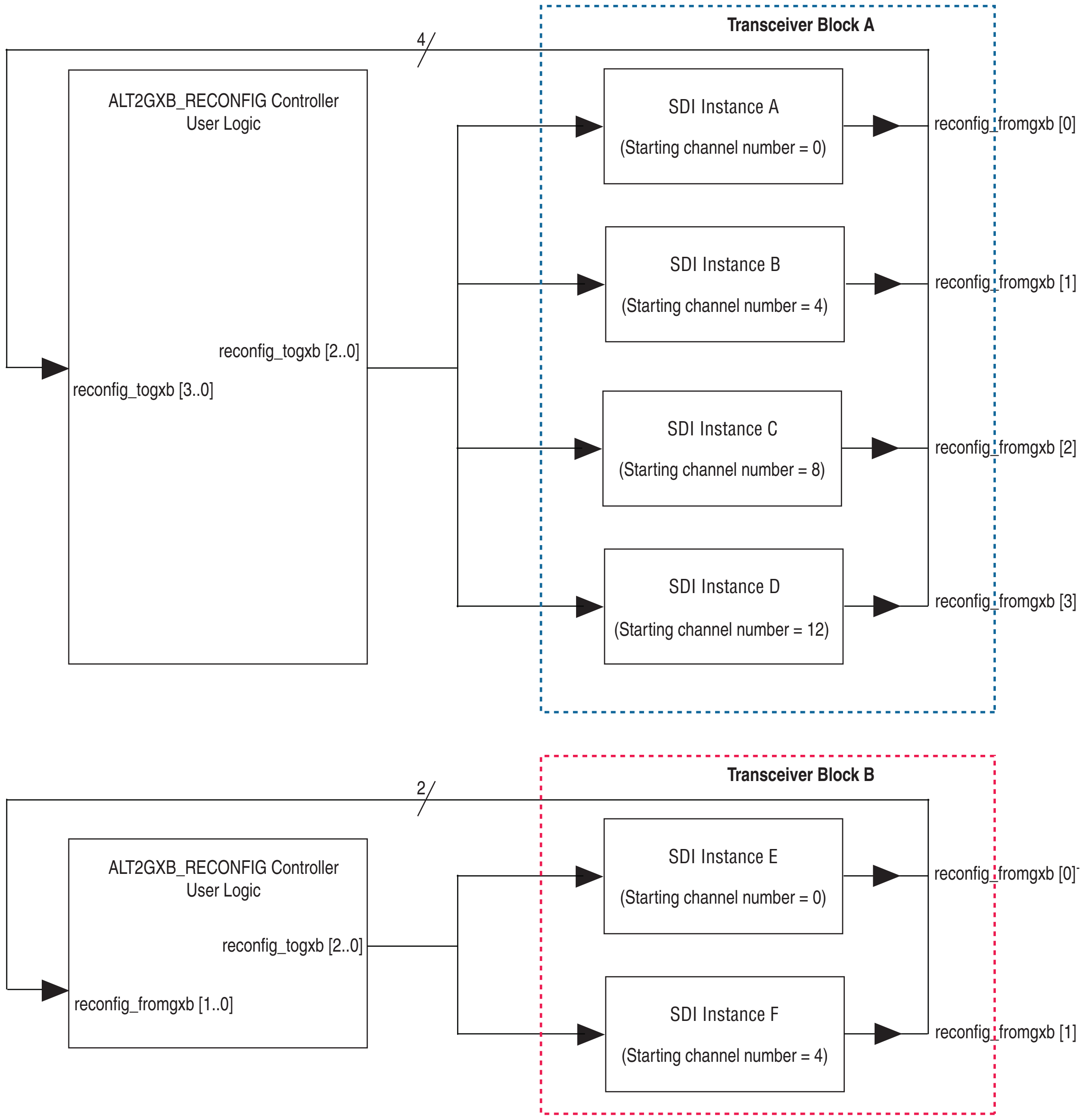
Figure 4. Connecting ALTGXB_RECONFIG Controller to SDI Instances From Multiple Transceiver Blocks



If your design requires more than four duplex SDI instances, you need multiple dynamic reconfiguration controllers. You can use multiple ALT2GXB_RECONFIG controllers to configure multiple instances of the SDI MegaCore located at different transceiver blocks.

Figure 5 shows how multiple ALT2GXB_RECONFIG controllers are connected to multiple SDI instances from multiple transceiver blocks.

Figure 5. Connecting Multiple ALTGXB_RECONFIG Controllers to SDI Instances From Multiple Transceiver Blocks



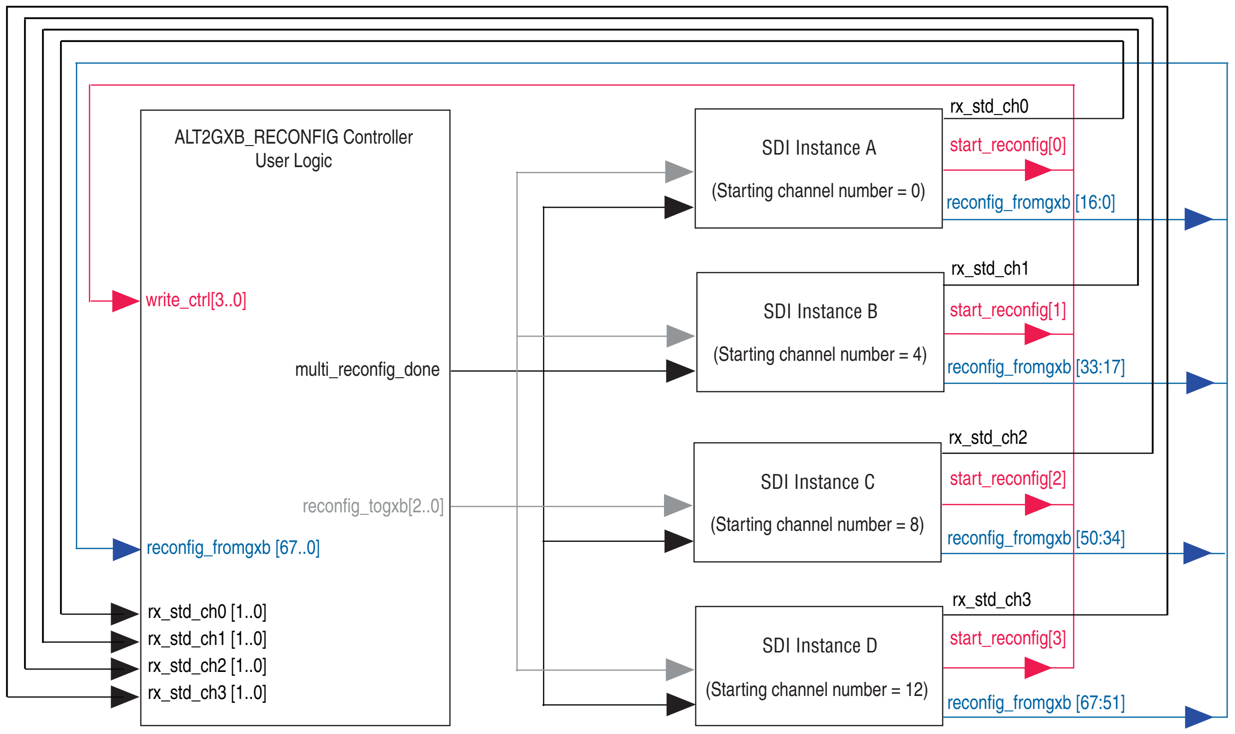
WARNING Two ALT2GXB_RECONFIG controllers cannot control the same SDI instance. The Quartus II software cannot merge multiple transceiver channel instances to become a transceiver block if multiple dynamic reconfiguration controllers are used, even if the channels are configured to the same protocol functional mode and data rate.

Connecting the ALT2GXB_RECONFIG Controller to the SDI Duplex Instances

To configure the SDI MegaCore function using the ALT2GXB_RECONFIG megafunction, you must provide some connecting user logic to implement the handshaking between the two blocks.

Figure 6 shows how you can connect the ALT2GXB_RECONFIG controller to duplex SDI instances in the same transceiver.

Figure 6. Connecting ALT2GXB_RECONFIG Controller to Duplex Instances in Same Transceiver



To connect the ALT2GXB_RECONFIG controller to duplex SDI instances, you need to execute the following steps:

1. Connect the `reconfig_fromgxb` signal from the SDI instance to the `reconfig_fromgxb` signal in the ALT2GXB_RECONFIG controller. Start connecting the lowest starting channel number of the SDI instance to the least significant bit and so on.
2. Connect the `reconfig_togxb` signal from the ALT2GXB_RECONFIG controller to the corresponding `reconfig_togxb` signal in all the SDI instances.

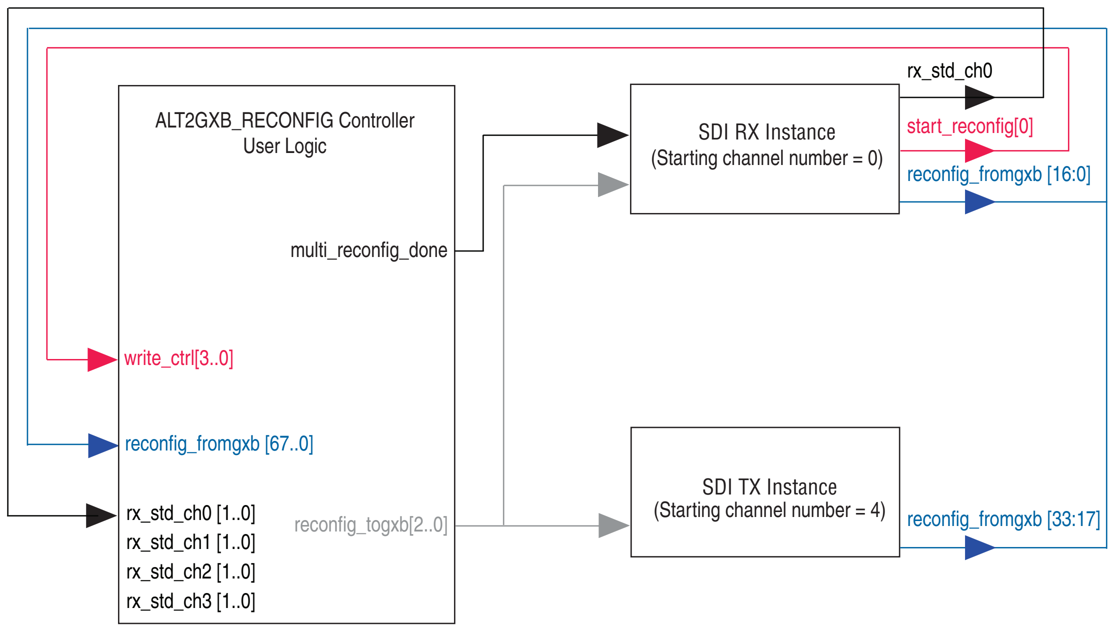
For example, the following codes are used in the SDI example design to execute steps 1 and 2:

```
sdi_tr_reconfig_multi sdi_tr_reconfig_multi_inst
(
.rst                   (reset),
.write_ctrl            (sdi_start_reconfig[3..0]),
.rx_std_ch0            (rx_std_ch0),
.rx_std_ch1            (rx_std_ch1),
.rx_std_ch2            (rx_std_ch2),
.rx_std_ch3            (rx_std_ch3),
.reconfig_clk          (sdi_reconfig_clk),
.reconfig_fromgxb      (reconfig_fromgxb),
.sdi_reconfig_done     (multi_reconfig_done),
.reconfig_togxb        (reconfig_togxb)
);
```

Connecting the ALT2GXB_RECONFIG Controller to the SDI Receiver and SDI Transmitter

Figure 7 shows how to connect the ALT2GXB_RECONFIG controller to the independent SDI receiver and transmitter in the same transceiver.

Figure 7. Connecting ALT2GXB_RECONFIG Controller to SDI Receiver and SDI Transmitter Instances in Same Transceiver



To connect the ALT2GXB_RECONFIG controller to the independent SDI receiver and transmitter, use the following code:

```
sdi_tr_reconfig_multi sdi_tr_reconfig_multi_inst
(
.rst                  (reset),
.write_ctrl           ({3'b000, sdi_start_reconfig[0]}),
.rx_std_ch0           (rx_std_ch0),
.rx_std_ch1           (2'b00),
.rx_std_ch2           (2'b00),
.rx_std_ch3           (2'b00),
.reconfig_clk         (sdi_reconfig_clk),
.reconfig_fromgxb     (reconfig_fromgxb),
.sdi_reconfig_done    (multi_reconfig_done),
.reconfig_togxb       (reconfig_togxb)
);
```

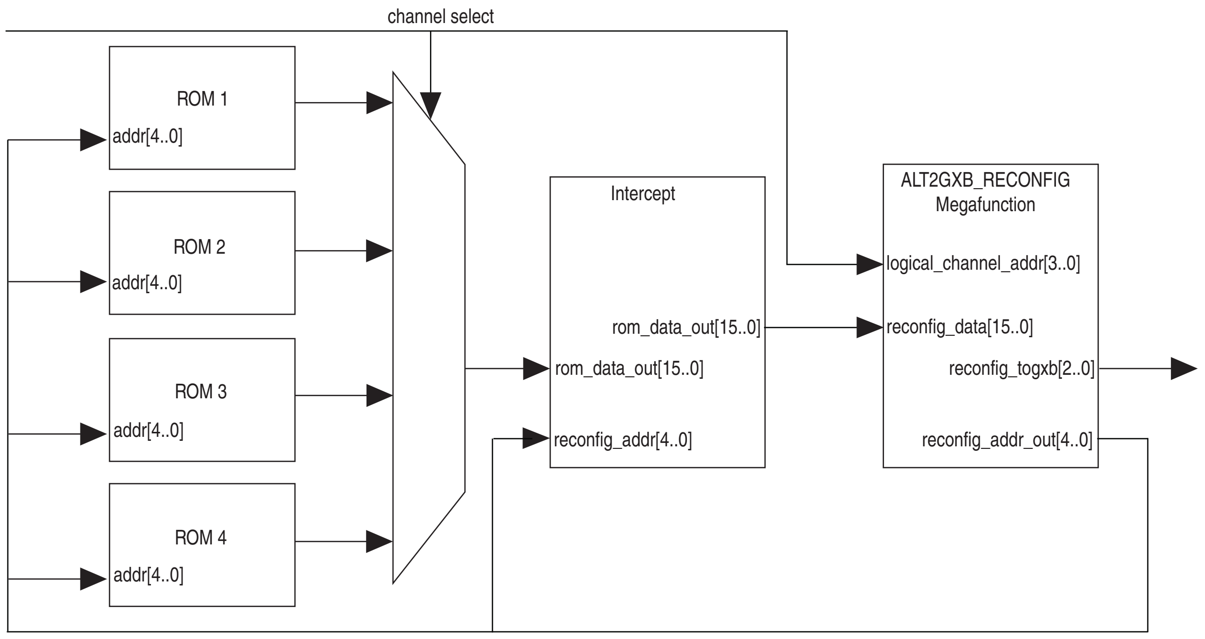
The dynamic reconfiguration in Figure 6 and Figure 7 only involves the SDI receiver and duplex SDI instances. The `SDI_START_RECONFIG` and `SDI_RECONFIG_DONE` signals handle the handshaking mechanism between the dynamic reconfiguration controllers and the SDI receiver or duplex instances. The SDI MegaCore function triggers the `SDI_START_RECONFIG` signal to the ALT2GXB_RECONFIG controller to start the dynamic reconfiguration process. The ALT2GXB_RECONFIG controller triggers the `SDI_RECONFIG_DONE` signal to the SDI MegaCore function to notify the completion of the dynamic reconfiguration process. The SDI duplex receiver or transmitter is at reset mode during the reconfiguration process.

The SDI transmitter is not required in the dynamic reconfiguration process. However, you must connect the `reconfig_togxb` and `reconfig_fromgxb` signals to the transmitter.

User Logic for ALT2GXB_RECONFIG Megafunction

This section describes how the local channel address is mapped in the user logic. Figure 8 shows the user logic for the ALT2GXB_RECONFIG controller in the SDI example design.

Figure 8. User Logic for ALT2GXB_RECONFIG Controller



The SDI example design provides the following code to control the ALT2GXB_RECONFIG megafunction:

```
sdi_4_ch_alt2gxb_reconfig sdi_alt2gxb_reconfig_inst
(
.logical_channel_address({ch_select,2'b00}),
.reconfig_clk(reconfig_clk),
.reconfig_data(reconfig_data),
.reconfig_togxb(reconfig_togxb),
.reconfig_address_en(rom_clk_enable),
.reconfig_address_out(rom_address),
.channel_reconfig_done (reconfig_done),
.write_all(reconfig_userlogic_write),
.reconfig_fromgxb(reconfig_fromgxb),
.busy(busy)
);
```

The user logic determines the logical channel address for dynamic reconfiguration. The logical channel number specifies the logical transceiver channel to be reconfigured by the ALT2GXB_RECONFIG controller. The `logical_channel_address` signal is asserted when the ALT2GXB_RECONFIG megafunction controls more than one channel. The channels are reconfigured one at a time. Assert the `logical_channel_address` signal to provide the necessary logical channel addresses to write the Memory Initialization File (MIF) words so that the channels are successfully reconfigured.

Figure 9 shows the mapping of the logical channel numbers to the SDI instances.

Figure 9. Mapping the Logical Channel Numbers to SDI Instances

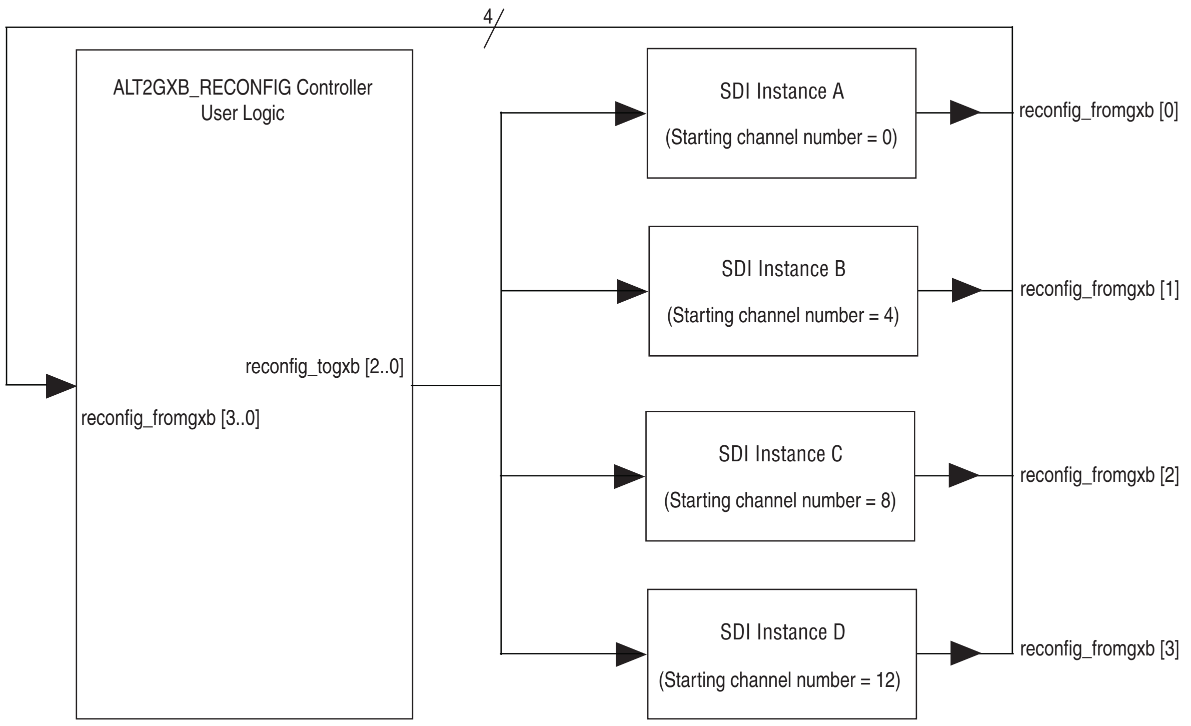


Figure 9 illustrates the mapping of the following logical channel addresses:

- `0000` maps to SDI instance A with channel number 0
- `0100` maps to SDI instance B with channel number 4
- `1000` maps to SDI instance C with channel number 8
- `1100` maps to SDI instance D with channel number 12

The value of the logical channel address depends on the value you assign to the **Starting channel number** option in the SDI MegaWizard interface. However, the value of the logical channel address is not affected by the physical placements of the transceiver channel.

The SDI instances A, B, C, and D in Figure 9 receive the serial signal through the `reconfig_togxb` signal. Each SDI instance compares the logical address to its own starting channel number. The SDI instance that matches its starting channel number to the logical address triggers the reconfiguration process, while the other SDI instances remain unchanged.

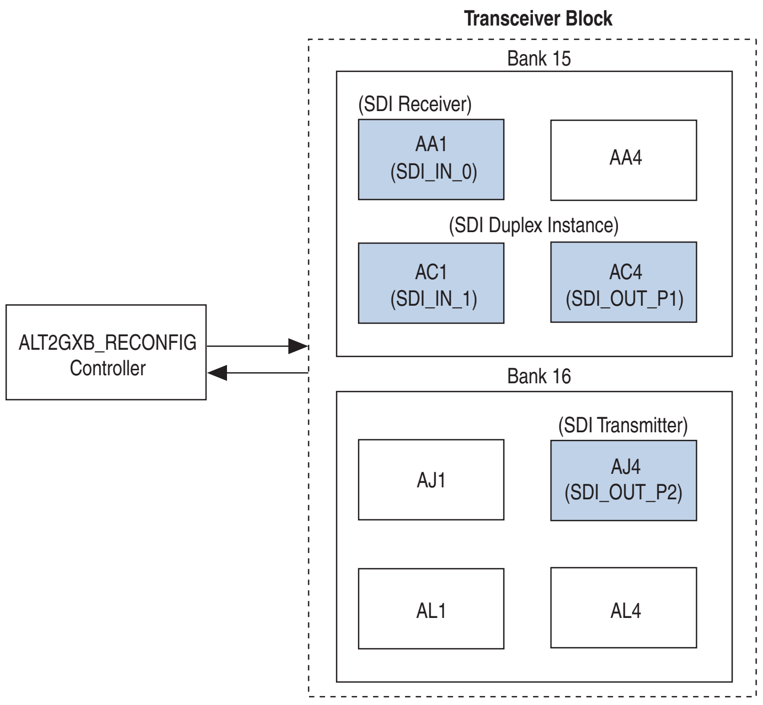
The logical channel address exists as `0000`, `0100`, `1000`, or `1100` and is controlled by the `ch_select` variable. The logical channel address of the ALT2GXB_RECONFIG megafunction maps to the starting channel number of each SDI instance.

SDI Example Designs With ALT2GXB_RECONFIG Controller

This section illustrates a couple of ways the ALT2GXB_RECONFIG controller is used in SDI applications.

In Figure 10, a single ALT2GXB_RECONFIG controller is used in a design where an SDI duplex instance and an independent SDI receiver reside in the same transceiver block.

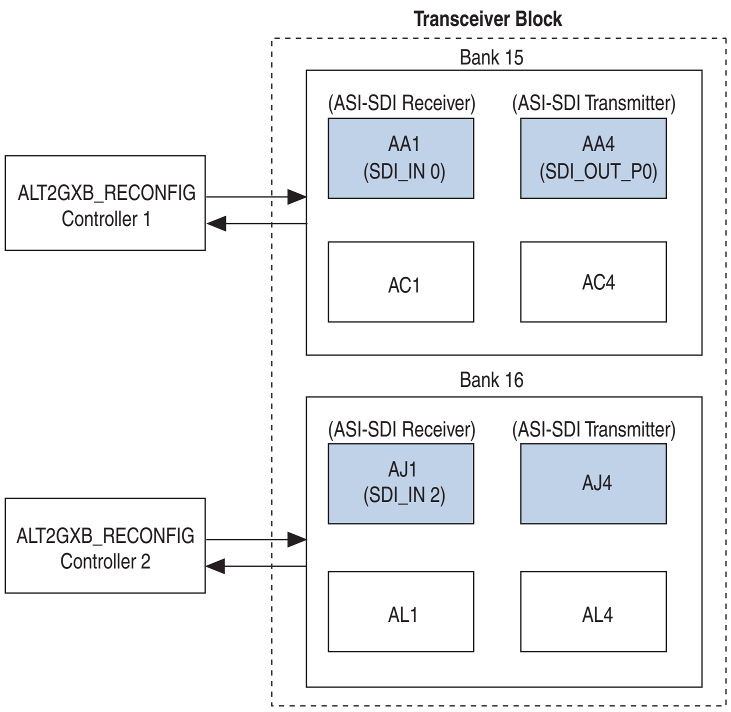
Figure 10. Design With Single ALT2GXB_RECONFIG Controller



For more information about this SDI design, refer to *AN 339: Serial Digital Interface Demonstration*.

In Figure 11, two ALT2GXB_RECONFIG controllers are used in two design modules where the ASI-SDI receivers reside in different transceiver blocks.

Figure 11. Design With Two ALT2GXB_RECONFIG Controllers

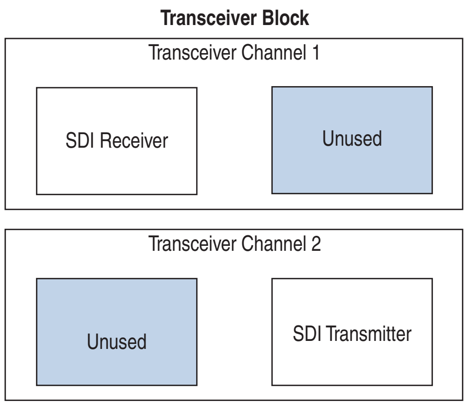


For more information about this SDI design, refer to *AN 551: ASI-SDI Auto Detect Reference Design for Stratix II GX Devices*.

Merging SDI Receiver and Transmitter in the Same Physical Channel

You can configure the SDI MegaCore as an independent receiver, independent transmitter, or duplex instance. The independent SDI receiver and transmitter can reside in different transceiver channels as shown in Figure 12, but cannot reside in the same transceiver channel.

Figure 12. SDI Transmitter and Receiver in Separate Transceiver Channels



If you place the independent SDI transmitter and receiver in the same channel, the Quartus II software shows the following error messages:

```
Error: Can't place GXB Central Control Unit
Error: Can’t fit design in device
```

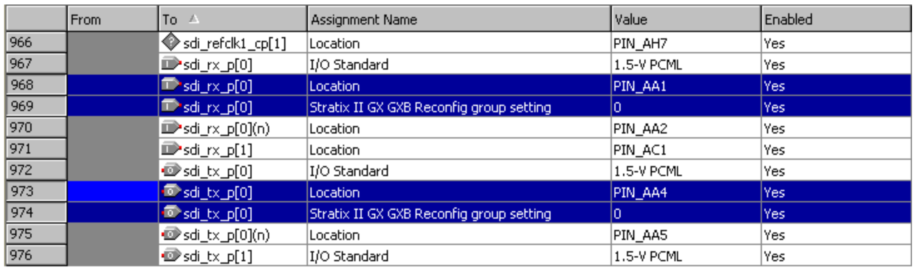
However, if your design requires the independent SDI transmitter and receiver to share the same transceiver channel, you can physically place the SDI transmitter and receiver into the same channel with additional configuration.

In the SDI example design, the receiver (`SDI_IN 0`) and the transmitter (`SDI_OUT_P2`) are placed in different channels. To physically place the transmitter (`SDI_OUT_P0`) in the same channel as the receiver, perform the following steps:

1. On the Assignments menu, click **Assignment Editor**.
2. In the **To** column, look for **sdi_tx_p**, and change the pin name from `PIN_AJ4` to `PIN_AA4` in the corresponding **Value** column.
3. In a new row, type **sdi_tx_p** in the **To** column and select **Stratix II GX GXB Reconfig group setting (Accepts wildcards/groups)** from the list, and type `0` in the corresponding **Value** column.
4. In a new row, type **sdi_rx_p** in the **To** column and select **Stratix II GX GXB Reconfig group setting (Accepts wildcards/groups)** from the list, and type `0` in the corresponding **Value** column.

Figure 13 illustrates the results of completing steps 1 to 4 in the Assignment Editor window.

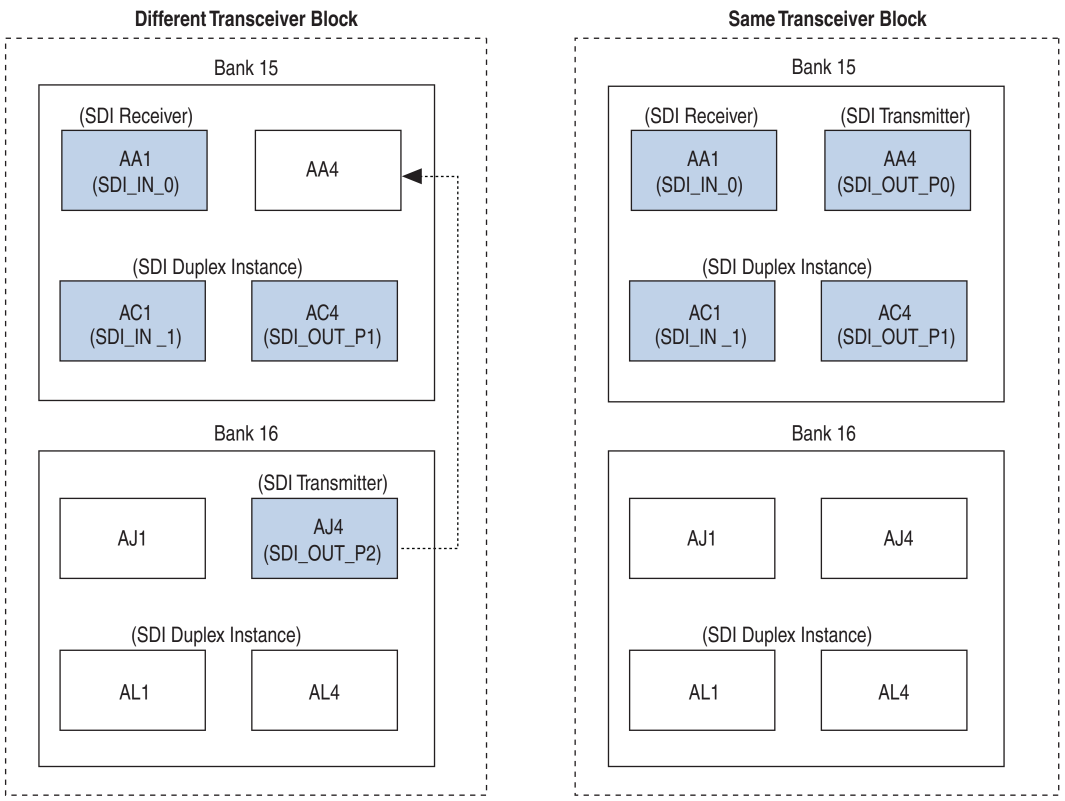
Figure 13. Assignment Editor Window



	From	To	Assignment Name	Value	Enabled
966		sdi_refclk1_cp[1]	Location	PIN_AH7	Yes
967		sdi_rx_p[0]	I/O Standard	1.5-V PCML	Yes
968		sdi_rx_p[0]	Location	PIN_AA1	Yes
969		sdi_rx_p[0]	Stratix II GX GXB Reconfig group setting	0	Yes
970		sdi_rx_p[0](n)	Location	PIN_AA2	Yes
971		sdi_rx_p[1]	Location	PIN_AC1	Yes
972		sdi_tx_p[0]	I/O Standard	1.5-V PCML	Yes
973		sdi_tx_p[0]	Location	PIN_AA4	Yes
974		sdi_tx_p[0]	Stratix II GX GXB Reconfig group setting	0	Yes
975		sdi_tx_p[0](n)	Location	PIN_AA5	Yes
976		sdi_tx_p[1]	I/O Standard	1.5-V PCML	Yes

By merging the SDI transmitter and receiver in the same physical channel, you free one of the transceiver channels as shown in Figure 14.

Figure 14. Merging SDI Transmitter and Receiver in Same Physical Channel



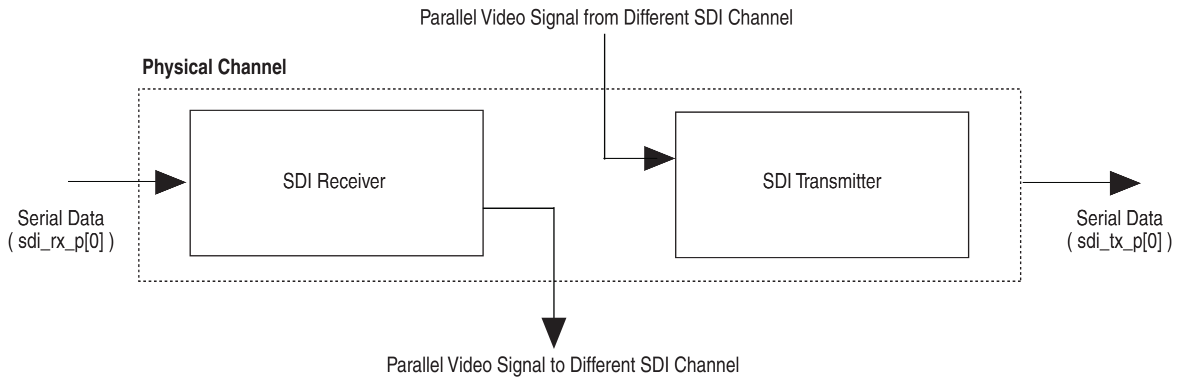
Altera recommends that you use a duplex SDI instance if you want to fully optimize the transceiver channel for designs targeting Stratix II GX devices. If you use too many instances that are merged from independent receiver and transmitter, your design consumes more device clock resources. This limitation does not affect the Stratix IV GX devices.

Receiver and Transmitter in the Same Physical Channel

The SDI transmitter and receiver in the same physical channel work independently. In normal operation, the transmitter and the receiver in the physical channel operate without affecting each other.

Figure 15 shows how a receiver and a transmitter work in the same physical channel.

Figure 15. SDI Transmitter and Receiver in Same Physical Channel


Parallel Video Signal from Different SDI Channel
Physical Channel
SDI Receiver
SDI Transmitter
Serial Data (sdi_rx_p[0])
Serial Data (sdi_tx_p[0])
Parallel Video Signal to Different SDI Channel

The following sections discuss the behaviors of the transmitter and receiver in the following processes:

- Dynamic Reconfiguration
- Change in Video Data Rate Transmission
- Transmitter and Receiver Reset

Dynamic Reconfiguration

The SDI receiver detects the change in the incoming video data rate, and triggers the channel reconfiguration process. The transmitter is not affected during the reconfiguration process.

Figure 16 and Figure 17 show how the SDI receiver internally triggers the `rx_analogreset` and `rx_digitalreset` signals to the GXB transceiver. The SDI receiver continues processing the video signal after the `rx_digital_reset` signal is deasserted.

Figure 16. During Dynamic Reconfiguration Process

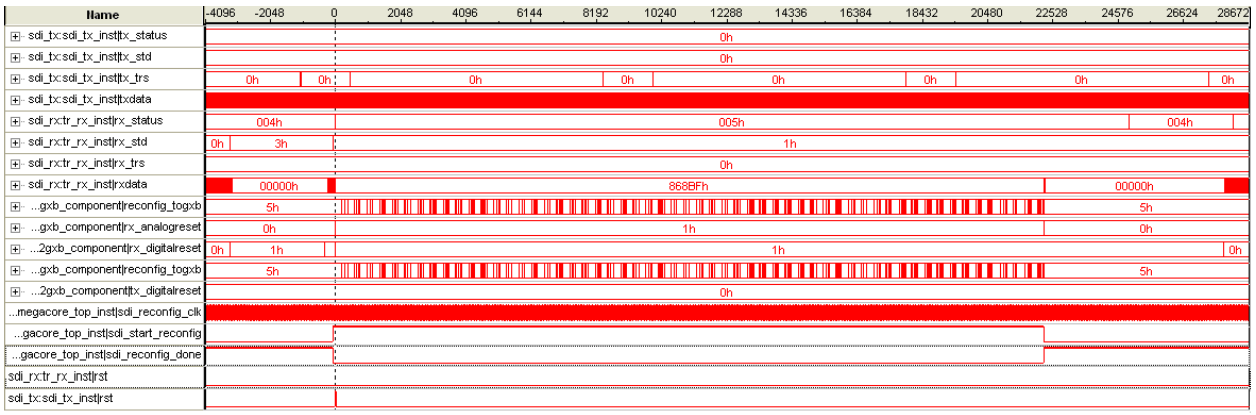
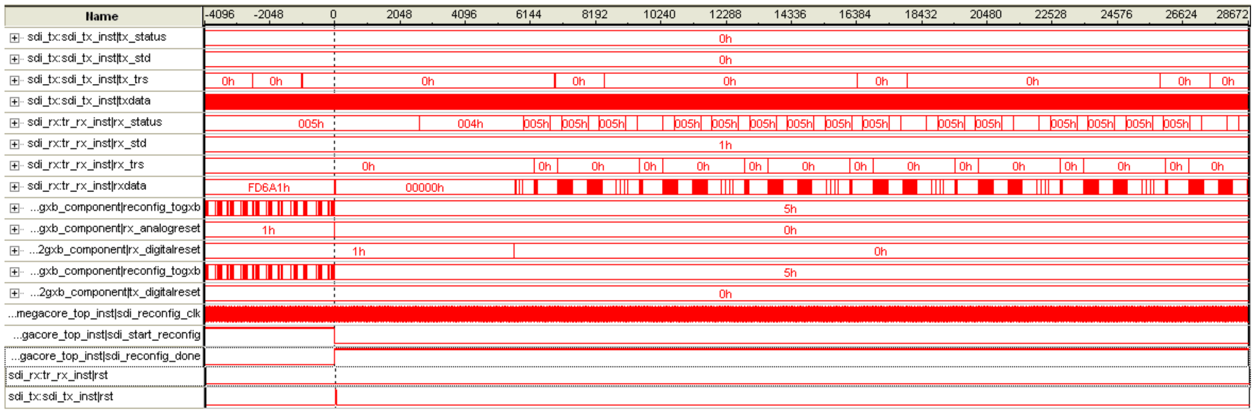

Name
-4096
-2048
0
2048
4096
6144
8192
10240
12288
14336
16384
18432
20480
22528
24576
26624
28672
sdi_tx:sdi_tx_inst|tx_status
sdi_tx:sdi_tx_inst|tx_std
sdi_tx:sdi_tx_inst|tx_trs
sdi_tx:sdi_tx_inst|txdata
sdi_rx:tr_rx_inst|rx_status
sdi_rx:tr_rx_inst|rx_std
sdi_rx:tr_rx_inst|rx_trs
sdi_rx:tr_rx_inst|rxdata
...gxb_component|reconfig_togxb
...gxb_component|rx_analogreset
...2gxb_component|rx_digitalreset
...gxb_component|reconfig_togxb
...2gxb_component|tx_digitalreset
...megacore_top_inst|sdi_reconfig_clk
...gacore_top_inst|sdi_start_reconfig
...gacore_top_inst|sdi_reconfig_done
sdi_rx:tr_rx_inst|rst
sdi_tx:sdi_tx_inst|rst
0h
004h
005h
3h
1h
00000h
868BFh
5h

Figure 17. After Dynamic Reconfiguration Process

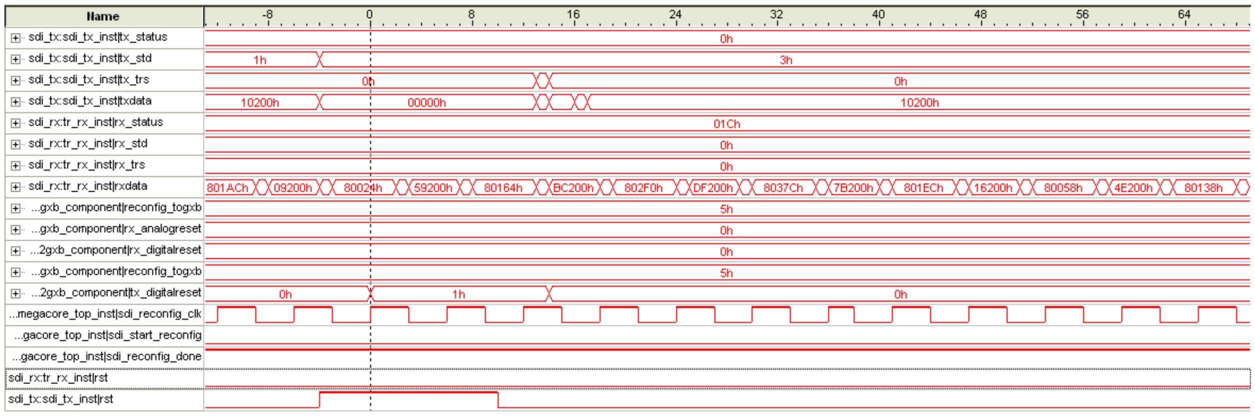


Change in Video Data Rate Transmission

The SDI transmitter does not trigger the reset function internally when the transmission of the video data rate changes. You must add user logic to trigger the SDI transmitter reset function while the transmitter changes the transmission of the video format.

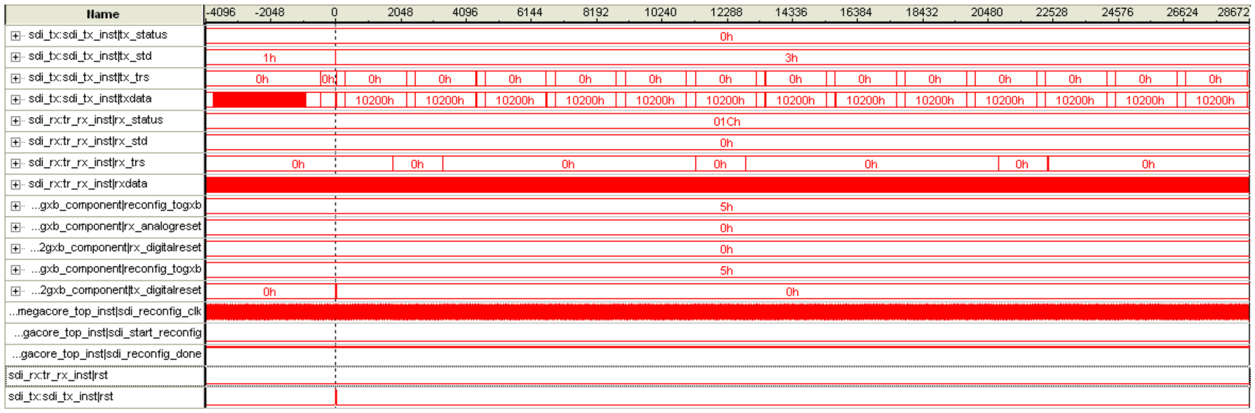
Figure 18 shows when the transmitter is reset when the transmission of the video format changes.

Figure 18. Triggering SDI Transmitter Reset



The SDI receiver is not affected during the change in the video data rate transmission. Figure 19 shows that the `rx_status` signals—`alignment locked`, `trs locked`, and `frame locked`—are locked to the incoming video data.

Figure 19. rx_status Signals Locked to Incoming Video Data


Name
sdi_tx:sdi_tx_inst|tx_status
sdi_tx:sdi_tx_inst|tx_std
sdi_tx:sdi_tx_inst|tx_trs
sdi_tx:sdi_tx_inst|txdata
sdi_rx:tr_rx_inst|rx_status
sdi_rx:tr_rx_inst|rx_std
sdi_rx:tr_rx_inst|rx_trs
sdi_rx:tr_rx_inst|rxdata
...gxb_component|reconfig_togxb
...gxb_component|rx_analogreset
...2gxb_component|rx_digitalreset
...gxb_component|reconfig_togxb
...2gxb_component|tx_digitalreset
...megacore_top_inst|sdi_reconfig_clk
...gacore_top_inst|sdi_start_reconfig
...gacore_top_inst|sdi_reconfig_done
sdi_rx:tr_rx_inst|rst
sdi_tx:sdi_tx_inst|rst

Transmitter and Receiver Reset

You can separately reset the SDI transmitter and receiver in the same physical channel.

Figure 20 shows that when you trigger the reset function for the SDI receiver, the transmitter is not affected. The `rx_status` signal indicates that when the receiver is being reset, the `rxdata` signal becomes zero and the `rx_trs` signal is low.

Figure 20. SDI Receiver Reset

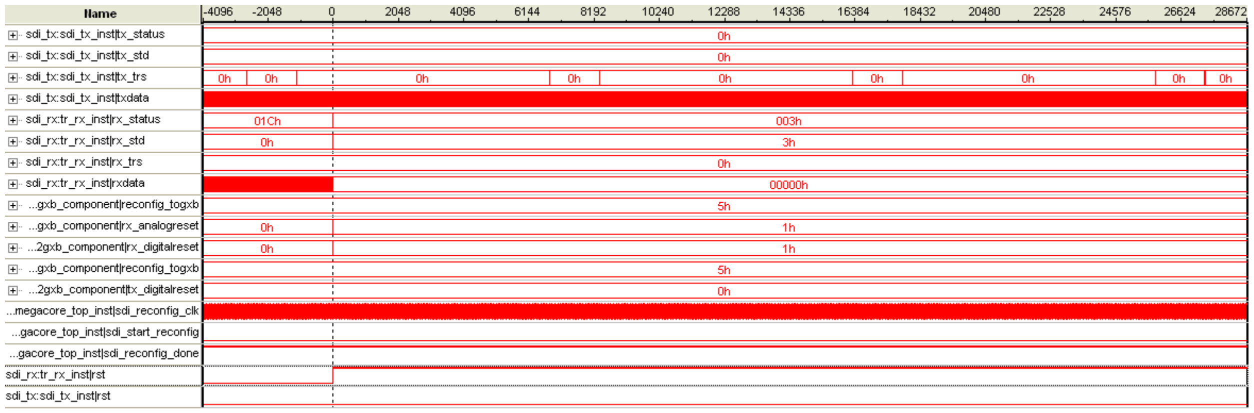
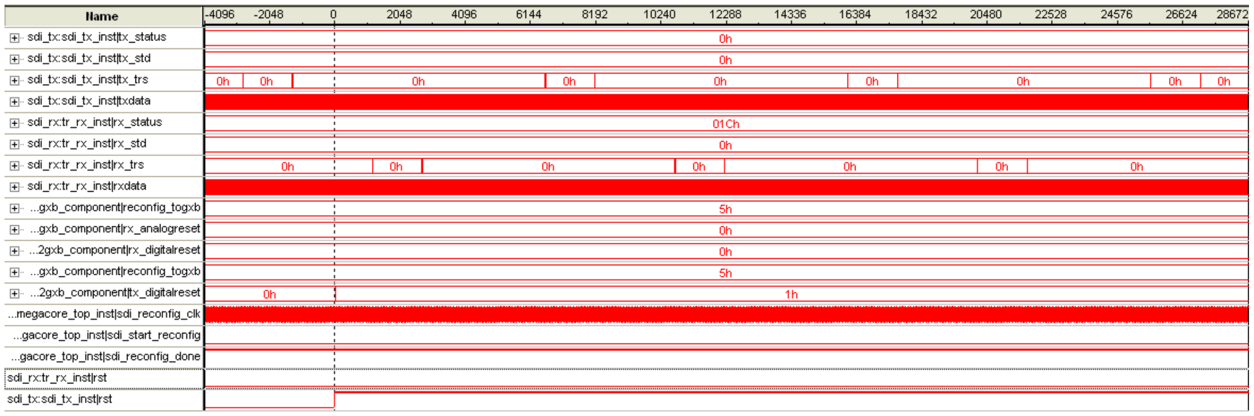

Name
sdi_tx:sdi_tx_inst|tx_status
sdi_tx:sdi_tx_inst|tx_std
sdi_tx:sdi_tx_inst|tx_trs
sdi_tx:sdi_tx_inst|txdata
sdi_rx:tr_rx_inst|rx_status
sdi_rx:tr_rx_inst|rx_std
sdi_rx:tr_rx_inst|rx_trs
sdi_rx:tr_rx_inst|rxdata
...gxb_component|reconfig_togxb
...gxb_component|rx_analogreset
...2gxb_component|rx_digitalreset
...gxb_component|reconfig_togxb
...2gxb_component|tx_digitalreset
...megacore_top_inst|sdi_reconfig_clk
...gacore_top_inst|sdi_start_reconfig
...gacore_top_inst|sdi_reconfig_done
sdi_rx:tr_rx_inst|rst
sdi_tx:sdi_tx_inst|rst

Figure 21 shows that the SDI receiver is not affected when the SDI transmitter is reset. The SDI receiver `rx_status` signals—`alignment locked`, `trs locked`, and `frame locked`—are locked to the incoming video data.

Figure 21. SDI Transmitter Reset


Name
-4096
-2048
0
2048
4096
6144
8192
10240
12288
14336
16384
18432
20480
22528
24576
26624
28672
sdi_tx:sdi_tx_inst|tx_status
sdi_tx:sdi_tx_inst|tx_std
sdi_tx:sdi_tx_inst|tx_trs
sdi_tx:sdi_tx_inst|txdata
sdi_rx:tr_rx_inst|rx_status
sdi_rx:tr_rx_inst|rx_std
sdi_rx:tr_rx_inst|rx_trs
sdi_rx:tr_rx_inst|rxdata
...gxb_component|reconfig_togxb
...gxb_component|rx_analogreset
...2gxb_component|rx_digitalreset
...gxb_component|reconfig_togxb
...2gxb_component|tx_digitalreset
...megacore_top_inst|sdi_reconfig_clk
...gacore_top_inst|sdi_start_reconfig
...gacore_top_inst|sdi_reconfig_done
sdi_rx:tr_rx_inst|rst
sdi_tx:sdi_tx_inst|rst
0h
01Ch
5h
1h

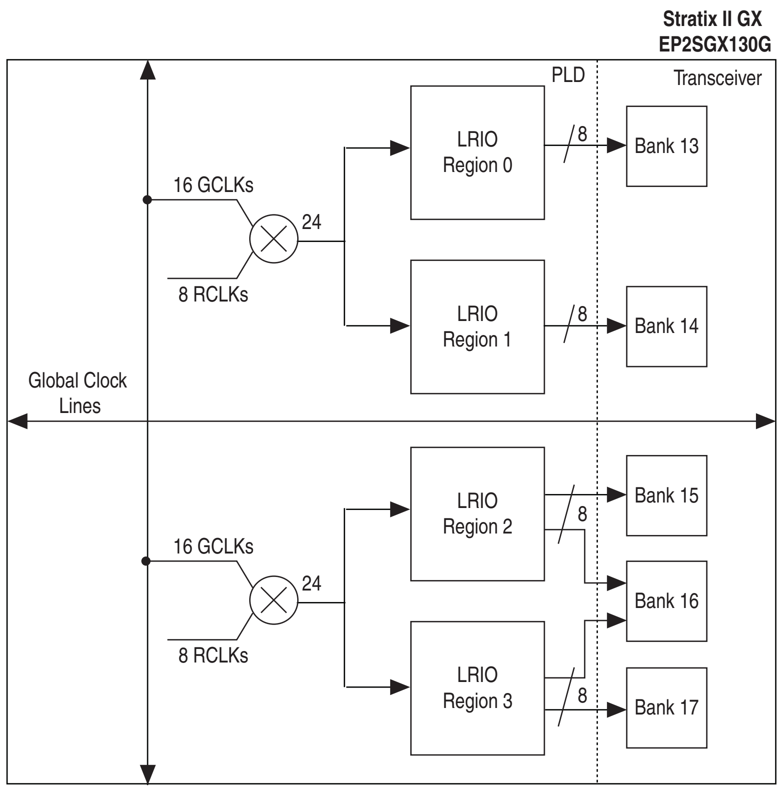
Stratix II GX LRIO Resources Limitation

If you want to route the regional or global clock network into the transceiver, you need a local route I/O (LRIO) channel. Each LRIO clock region has up to eight clock paths and each transceiver block has a maximum of eight clock paths to connect to the LRIO clock. These LRIO resources are limited and they determine the number of clocks that can be used between the FPGA and transceiver blocks.

The LRIO resources are device dependent. If your design requires multiple transceiver blocks, you must include the device's LRIO resources in your design plan to avoid no-fit device issue.

For example, the Stratix II GX EPS2GX130G device has five transceiver blocks; banks 13, 14, 15, 16 and 17, refer to Figure 22 on page 20.

Figure 22. Stratix II GX EPS2GX130G LRIO Resource



From the FPGA clock network, you can drive a maximum of 16 transceiver clocks for each upper half (banks 13 and 14) and each lower half (banks 15, 16, and 17) of the Stratix II GX EPS2GX130G device. The transceiver blocks in the lower half of the Stratix II EPS2GX130G device share LRIO regions. The LRIO region to the transceiver block mapping is hardwired in the device and is non-reconfigurable.

Because the independent transmitter and receiver require two instances and the duplex instance require only one, the independent transmitter and receiver consume more LRIO resources in an SDI design. Altera recommends that you use more SDI duplex instances in your design to optimize the LRIO resources. The upper half of the Stratix II GX EPS2GX130G device has more LRIO resources compared to the lower half. Altera recommends that you begin placing the independent SDI receiver and transmitter in the upper half of the Stratix II GX EPS2GX130G device first.

This limitation does not affect the Stratix IV GX devices.

For information about Stratix II GX devices and dynamic reconfiguration, refer to the *Stratix II GX Dynamic Reconfiguration* chapter in the *Stratix II GX Device Handbook*. For more information about the LRIO resources limitation, refer to the PLD Clock Resource section in the *Stratix II GX Transceiver Architecture Overview* chapter in the *Stratix II GX Device Handbook*.

Conclusion

Using the SDI example design as a guide, you can now reconfigure multiple SDI instances across a number of transceiver blocks using multiple dynamic reconfiguration controllers. Even though the SDI example design discussed in this application note is specifically for Stratix II GX devices, the issues discussed are applicable to all device families supported by the SDI MegaCore function.

Document Revision History

Table 1 shows the revision history for this application note.

Table 1. Document Revision History

Date and Revision	Changes Made	Summary of Changes
August 2009, version 1.0	Initial Release.	—