Channel Bonding with HOTLink II™ - AN014

AN014

Author: Naveen Srinivasan
Associated Project: No
Associated Application Notes: None

Abstract

The HOTLink II™ family of devices are point-to-point or point-to-multipoint communications building block that provide serialization, deserialization and framing functions. They can transport serial data at rates between 0.2 and1.5 Gigabits per second (Gbps) per channel and are compatible with communication standards such as Gigabit Ethernet, Fibre Channel, SMPTE-259M, SMPTE-292M, DVB-ASI and ESCON®.

Introduction

HOTLink II Background

The CYP15G0401DXB is a quad channel HOTLink II device that includes 8B/10B encoding and decoding functionality. It also has the capability to perform channel bonding of parallel data channels across multiple channels and multiple devices to enable wider paths of data transfer like 16 bits, 32 bits, 64 bits, 128 bits, etc.

The purpose of this application note is to discuss the following:

1. Channel bonding overview
2. Configuration guidelines
3. Channel bonding process
4. Layout guidelines for bonding communication signals in multidevice bonding.

Channel Bonding Overview

What is channel bonding?

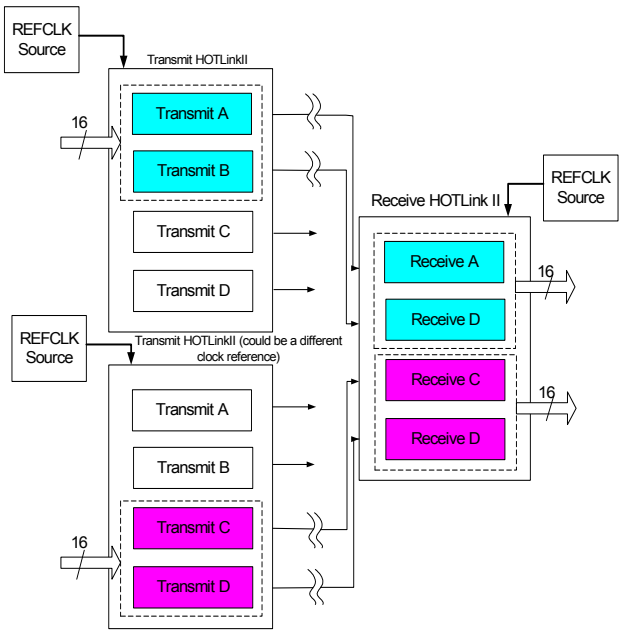
Channel bonding is a method used to combine and align multiple data channels in order to form a wider TX/RX data path. This feature provides the capability to align receive parallel data across multiple channels/devices with respect to a common clock. The multiple channels are aligned using a bonding sequence transmitted from the transmitting device to the receiving device across all channels. Based on the arrival of the bonding sequence across different channels, the elasticity buffer present in the receive side logic will realign the parallel data across multiple channels.

The CYP15G0401DXB allows configuration for different channel bonding options as follows:

1. Dual-channel bonding

In the dual-channel bonding mode, channels A, B and C, D are bonded to form two16-bit wide data buses. In this case, the bonding function of channels A, B is independent from that of channels C, D. The incoming serial streams for channels A and B need to be from a common reference (0 ppm offset). Similarly, the incoming streams for channels C and D need to be from a common reference. From the transmitter end the parallel data of the bonded channels should be clocked using the same clock (TXCKSEL = LOW or HIGH). An example block-level representation of dual-channel bonding is shown in Figure 1.

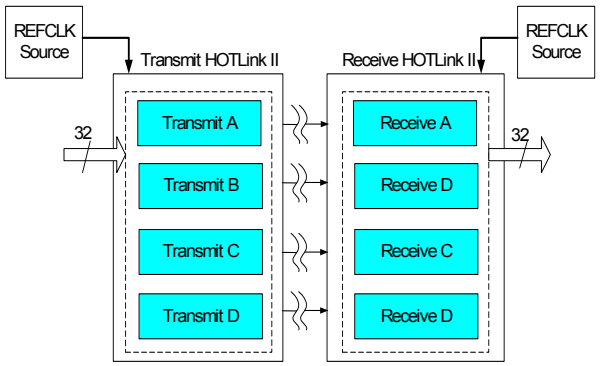
Figure 1. Dual Channel Bonding



2. Quad-channel bonding

In the quad-channel bonding mode, channels A, B and C, D are bonded within one device to form a 32-bit data bus. The incoming serial streams for channels A, B and C, D need to be from the same clock reference; in other words, the incoming streams should have a frequency offset of 0 ppm. From the transmitter end, the parallel data of the bonded channels should be clocked using the same clock (TXCKSEL = LOW or HIGH). A block-level representation of quad-channel bonding is illustrated in Figure 2.

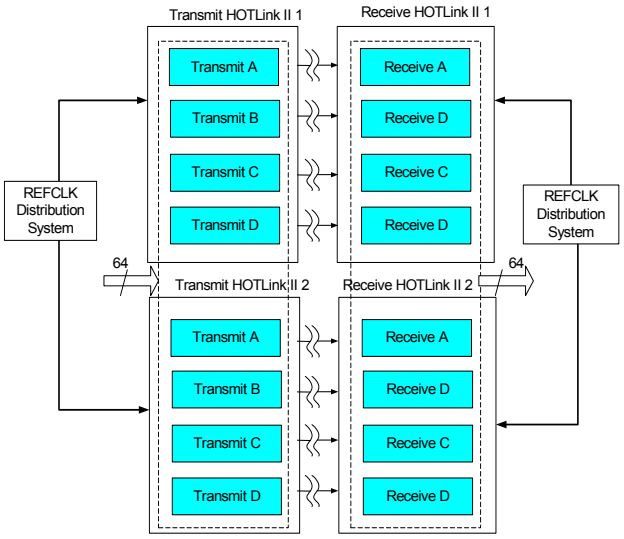
Figure 2. Quad Channel Bonding



3. Multidevice bonding

One of the unique features of channel bonding with HOTLink II is that the quad-channel bonding mode can be expanded to work in a multidevice environment wherein all channels of multiple devices are bonded. In this case, data-path widths of 64 bits or higher can be achieved. A block-level representation of multidevice channel bonding using the two device channel bonding case is shown in Figure 3.

Figure 3. Two-Device Channel Bonding

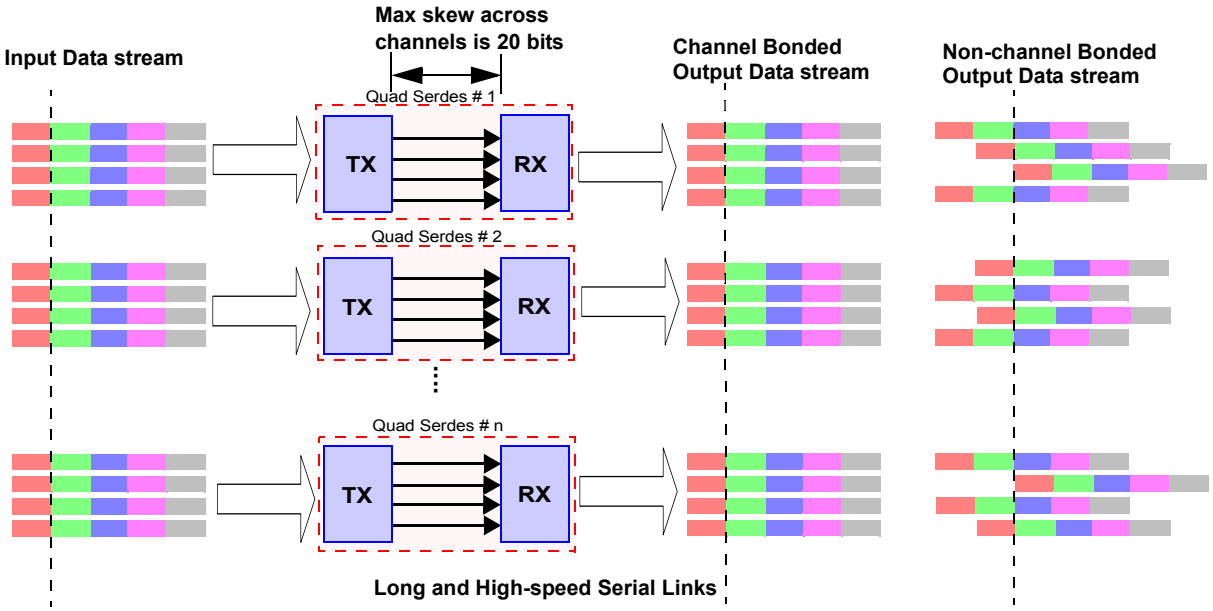


Why is channel bonding needed?

The main need to bond multiple parallel channels together is to transfer wider databuses from one point to another point with a few serial links, thereby achieving a very high point to point bandwidth.

If there is no channel bonding, there might be misalignment in data received in the RX bus across multiple channels with respect to the data transmitted in the TX bus. This misalignment might be caused due to differences in signal propagation across multiple serial links and/or asymmetric skew among channels present in the communication system. The CYP15G0401DXB can align channels skewed by up to twenty serial bit times, measured at the serial input interfaces as shown in Figure 4 on page 3.

Figure 4. Difference Between Channel Bonded and Non-channel Bonded Output



Where in HOTLink II is the channel bonding implemented?

The channel bonding is implemented in the elasticity buffer in the receiver sections of the CYP15G0401DXB. The offset of the elasticity buffer is independently adjusted in each channel in order to achieve the channel bonding functionality.

What are the main application areas of channel bonding?

Channel bonding has applications mainly where there is a need for transporting data paths wider than eight bits. Another area of application is in certain networking applications, where multiple data channels are required to be aligned at the receiver end. With intrachip channel bonding the CYP15G0401DXB can be used to transport 16- or 32-bit-wide data paths. With multi-chip bonding, the data paths can be as wide as 64, 96,128 bits, etc., depending on how many devices are bonded together. For example, customers may implement the multidevice bonding design feature to implement a wireless base station for 64-bit-wide data transfer from one card to another card using eight serial links.

Table 1. Required Settings for Channel Bonding

Signal Name	Setting	Comments
RXMODE[1]	MID or HIGH	MID is for dual channel bonding. HIGH is for quad/multidevice channel bonding.
RXMODE[0]	LOW or HIGH	LOW is for Status A reporting and HIGH is for Status B reporting.
RXCKSEL	LOW or HIGH	When LOW, all four output registers are clocked using REFCLK and RXCLKA± and RXCLKC± present buffered and delayed forms of REFCLK. For multi-device channel bonding it is necessary to configure this input as LOW. When LOW and dual channel bonding is enabled, the insertion and deletion of K28.5 characters is controlled by the master channel in each bonding pair which is selected by using RXCLKB+ and RXCLKD+ as shown in Table 2 on page 5. When LOW and quad channel bonding is enabled, the master channel for generating BONDST[1:0] status is determined by settings of RXCLKB+ and RXCLKD+ as shown in Table 3 on page 5. When HIGH and dual channel bonding is enabled (RXMODE[1] = MID) one of the recovered clocks from the bonding pair is used to clock the parallel bonded data out of the device. RXCLKA± presents the recovered clock from either channel A or B depending on the setting of RXCLKB+, and RXCLKC± presents the recovered clock from either channel C or D depending on the setting of RXCLKD+, as shown in Table 2 on page 5. The same table also shows the selection of master channel. When HIGH and quad channel bonding is enabled (RXMODE[1] = HIGH) one of the four recovered clocks of the device is used to clock the parallel bonded data out of the device. RXCLKA± and RXCLKC± present the recovered clock from either channel A, B, C or D depending on settings of RXCLKB+ and RXCLKD+ as shown in Table 3 on page 5.
DECMODE	MID or HIGH	Channel bonding can be done only when the 8B/10B Decoder in the receiver logic is enabled.
RFEN	HIGH (at least until framing is established)	Data in the receiver must be framed before the channel bonding is done. For low-latency framer (RFMODE = LOW) selection, RFEN must be dynamically switched off after the channels have established framing in order to avoid framing to alias framing characters.

Configuration Guidelines

General Configuration for Channel Bonding

Some of the static control signals of the HOTLink II are 3-level select inputs. These inputs can be set as LOW (logic-0), MID (open) or HIGH (logic-1). It is not recommended to control these inputs using a logic device such as an FPGA, since the DC levels for the three level input HIGH and LOW have lesser noise margin than LVTTL DC levels. Moreover, controlling an FPGA output in three-state mode to achieve the MID state for the three level inputs is not a recommended practice.

The required settings for channel bonding are shown in Table 1.

Since the 8B/10B decoders in the HOTLink II receivers are enabled for the channel bonding, it is required for the serial output data coming out of the transmitters to be compatible with the 8B/10B encoding/decoding scheme. The encoders in the transmitters may or may not be enabled as discussed in the following cases.

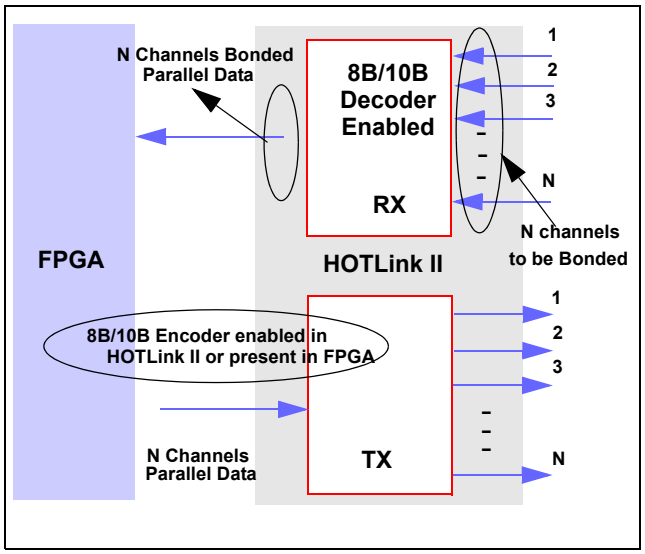
Case 1: 8B/10B Encoding Enabled in the HOTLink II

If the transmit logic in HOTLink II is configured with the 8B/10B encoders enabled, then the TXMODE[1] must be configured as either MID or HIGH.

Case 2: 8B/10B Encoding disabled in the HOTLink II and a compatible 8B/10B Encoder implemented in an ASIC or FPGA

Figure 5 on page 5 illustrates, within the encircled region, that 8B/10B encoding should be enabled within the HOTLink II device or in the upstream FPGA.

Figure 5. Enabling of 8B/10B in channel bonding



The settings discussed in Table 1 on page 4 are required for channel bonding to be implemented. The following section discusses the additional settings for different types of channel bonding.

Configuration for Intrachip Channel Bonding

Control Settings for Dual Channel Bonding

In dual-channel bonding mode, only channels A and B together and channels C and D together can be bonded. Channels A and C, A and D, B and C, and B and D cannot be bonded in this mode.

In this case, the CYP15G0401DXB needs to be configured as follows:

1. RXMODE[1] should be set to MID.
2. If RXCKSEL is configured as HIGH, then RXCLKA± outputs the recovered clock from either receive channel A or channel B, as selected by RXCLKB+. RXCLKC± outputs the recovered clock from either receive channel C or channel D, as selected by RXCLKD+. Note. RXCLKB+ and RXCLKD+ will be treated as LVTTL control inputs in this case. Inputs RXCLKB+ and RXCLKD+ also select the master channel, as shown in Table 2.
3. If RXCKSEL is configured as LOW, then RXCLKA± and RXCLKC± output a buffered version of REFCLK. Inputs RXCLKB+ and RXCLKD+ select the master channel for each bonding pair, as shown in Table 2.
4. The pins BOND_ALL and BONDST[1:0] are not used in dual channel bonding mode. These pins should be left open.
5. The signal $\overline{\text{MASTER}}$ is not interpreted in dual channel bonding mode. This pin should be left open.

Table 2. Dual Channel Bonding Recovered Clock Select

RXCLKB+[1]	RXCLKD+[1]	Clock Source		Master Channel
		RXCLKA±	RXCLKC±	
0	X	RXCLKA		A
1	X	RXCLKB		B
X	0		RXCLKC	C
X	1		RXCLKD	D

Control Settings for Quad Channel Bonding

In quad channel bonding mode, channels A, B, C, and D are bonded together. In this case, the CYP15G0401DXB needs to be configured as follows:

1. RXMODE[1] should be set to HIGH.
2. If RXCKSEL is configured as HIGH, then RXCLKA± and RXCLKC± output the recovered clock from receive channel A, B, C or D, as selected by the combined settings of RXCLKB+ and RXCLKD+. The same setting of RXCLKB+ and RXCLKD+ also decides the master channel. Refer to Table 3.
3. When RXCKSEL is configured as HIGH, the inter-device bonding functionality is disabled. In this case, the $\overline{\text{MASTER}}$, BOND_ALL, and BONDST[1:0] should be left open. Please refer to case 1 in Figure 6 on page 6.
4. When RXCKSEL is configured as LOW, then RXCLKA± and RXCLKC± output a buffered version of REFCLK. Inputs RXCLKB+ and RXCLKD+ select the master channel as shown in Table 3.
5. When RXCKSEL is configured as LOW in Quad Channel Bonded mode, the open-drain IO pins BOND_ALL and BONDST[1:0] become active, and each one requires a strong external pull-up resistor (36 ohms) to operate properly. Please refer to case 2 in Figure 6 on page 6.
6. When RXCKSEL=LOW and Quad Channel Bonding is enabled, the signal $\overline{\text{MASTER}}$ should be driven low for Single Device Quad Channel Bonding mode.

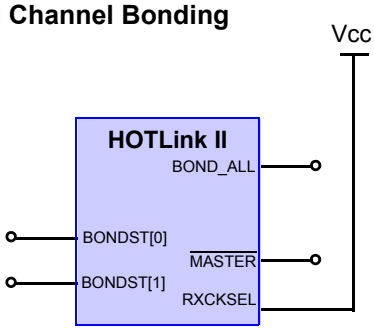
Table 3. Quad Channel Bonding Recovered Clock Select

RXCLKB+[1]	RXCLKD+[1]	Clock Source	Master Channel
		RXCLKA±/RXCLKC±	
0	X	RXCLKA	A
1	X	RXCLKB	B
X	0	RXCLKC	C
X	1	RXCLKD	D

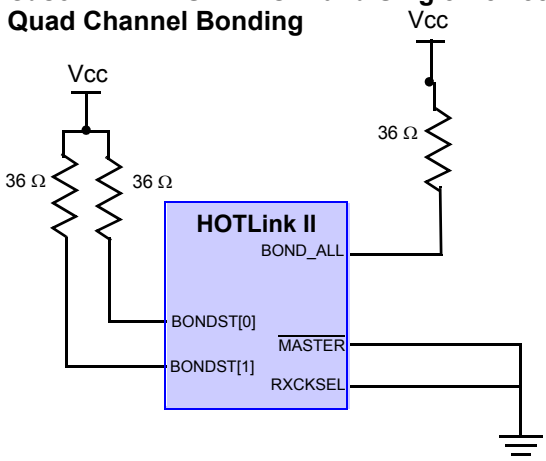
Note
1. RXCLKB+ and RXCLKD+ are treated as LVTTL inputs when channel bonding is enabled.

Figure 6. Bonding Setting for Quad Channel Bonding

***Case 1*: RXCKSEL=HIGH and Single Device Quad Channel Bonding**



***Case 2*: RXCKSEL=LOW and Single Device Quad Channel Bonding**



Configuration for Multi-Device Channel Bonding

The primary requirement for multi-chip channel bonding is that all the CYP15G0401DXB devices that are bonded together must be configured in quad channel bonding mode, with RXCKSEL = LOW.

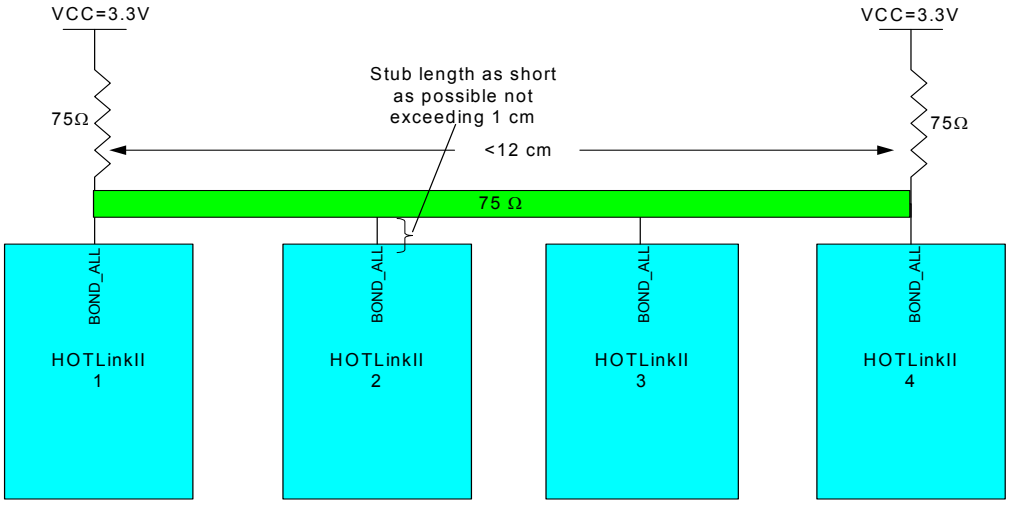
In this configuration, only one of the CYP15G0401DXB devices must be configured as the master device. One of the four channels of this device will be the master channel depending on the settings shown in the Table 4. For this device, $\overline{\text{MASTER}}$ must be configured as LOW. The rest of the CYP15G0401DXB devices must be configured as slaves by setting their $\overline{\text{MASTER}}$ pin to HIGH. For the master device, the BONDST[1:0] are used as outputs going into the BONDST[1:0] of the slave devices as inputs. The bidirectional BOND_ALL IO signals of all devices should be tied together for inter chip communication.

The $\overline{\text{BOND_INH}}$ pin must be equally configured for all the devices that are bonded. When set to LOW, $\overline{\text{BOND_INH}}$ inhibits bonding between the channels unless all channels in the configuration can be bonded.

Table 4. Multi-device Bonding Master Channel Select

RXCLKB[2]	RXCLKD[2]	Master Channel
0	0	A
0	1	B
1	0	C
1	1	D

Figure 7. Layout of BOND_ALL Signals in Interdevice Bonding of Two or More Devices



Note

2. RXCLKB+ and RXCLKD+ are treated as LVTTL inputs when channel bonding is enabled.

Layout Guideline for Bonding Communication Signals in Multidevice Bonding

The bonding communication signals BOND_ALL, BONDST[1:0] have very fast edge rates and should be treated as high-speed signals. The traces for these signals need to be designed taking high-speed board design guidelines into consideration. For an introduction to high-speed board design, please refer to the application note entitled High-speed Board Design Guidelines (using CYS25G0101DX as an example).

The next section covers the specific layout guidelines for BOND_ALL and BONDST[1:0], when multidevice bonding is enabled.

BOND_ALL:

The requirements for the BOND_ALL transmission line that connects the BOND_ALL pins of all devices that are bonded are as follows:

1. The line that connects all the BOND_ALL pins should have a uniform characteristic impedance of 75 ohms.
2. Since the BOND_ALL signal has very fast edge rates the line should be terminated at both ends using the characteristic impedance. Termination at both ends is required in all cases of multi-device bonding including two device bonding, due to the fact that the BOND_ALL signal is a bidirectional I/O. The recommended termination is two 75-ohm pull-up resistors, one at each end of the transmission line.
3. The stubs between the BOND_ALL pin of each device and the transmission line should be as short as possible not exceeding 1 cm.
4. The overall length of the BOND_ALL transmission line connecting all the BOND_ALL pins should as short as possible and should not exceed 12 cm.

An example of BOND_ALL layout for four-device bonding is shown in Figure 7 on page 6. More specific BOND_ALL trace layout depending on the number of devices bonded is shown in the appendix.

BONDST[1:0]:

The requirements for the signals in the BONDST[1:0] bus are as follows:

1. The two lines that connect all the BONDST[1] of all bonding devices and BONDST[0] pins of all bonding devices should have a uniform characteristic impedance of 75 ohms.

Figure 8. Layout of BONDST[1:0] Signals When the MASTER Device is Present in One End of the Transmission Line

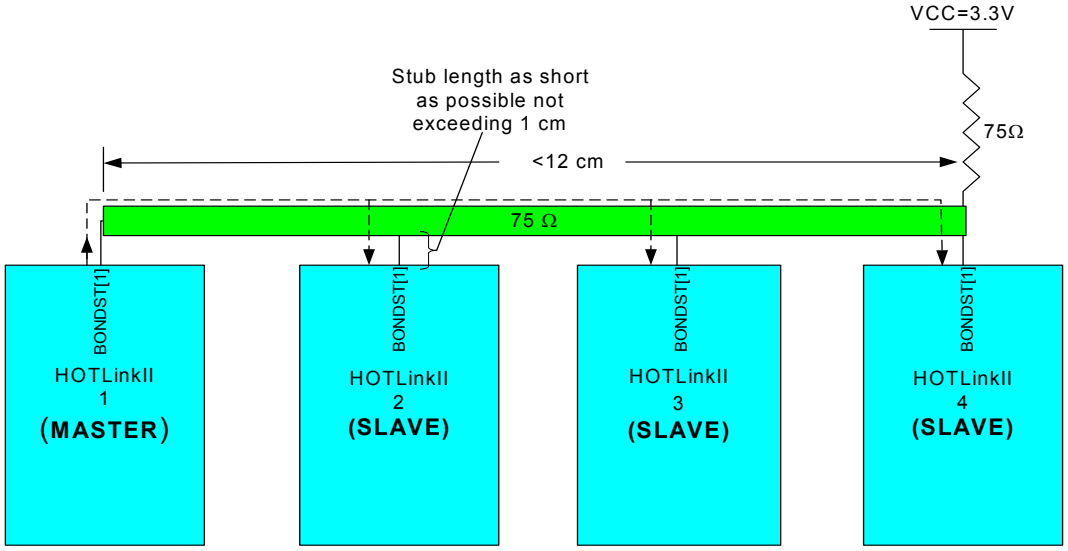
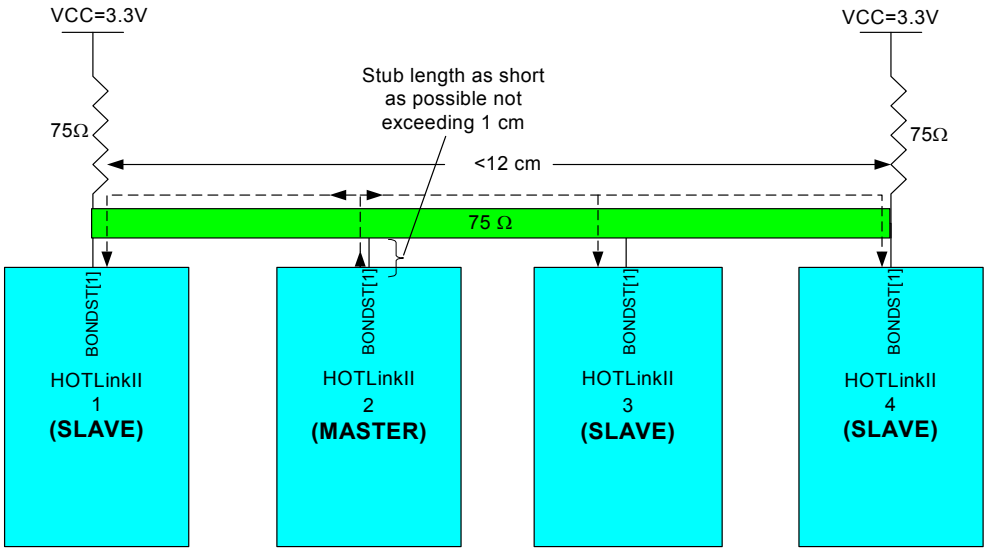


Figure 9. Layout of BONDST[1:0] Signals When the MASTER is Not Present in the End of the Transmission Line



2. Since the BONDST[1:0] signals have very fast edge rates the transmission lines should be terminated at the receiving end using a pull up resistor with a value equal to the characteristic impedance. When the master device is located at the end of the transmission line, termination is needed only at the other end of the transmission line as shown in Figure 8 on page 7. It is recommended to keep the master device at the end of the transmission line whenever possible. In the case of three or more devices bonded together, termination is necessary at both ends whenever the master device is not located at the end of the transmission line as shown in Figure 9. The recommended termination is a 75-ohm pull-up resistor.
3. The stubs between the BONDST[1:0] pins of each device and the corresponding transmission lines should be as short as possible not exceeding 1 cm.
4. The overall length of the transmission lines connecting all the BONDST[1:0] pins of all devices together should be as short as possible not exceeding 12 cm.

Channel Bonding Process

Requirements

The basic requirements for channel bonding are:

1. The incoming serial streams to channels that are bonded must have 0-ppm frequency offset. In other words, all the transmitting channels/devices should be timed using the same REFCLK source.
2. The skew between the incoming serial data across any two channels should be less than 20 serial bit times.
3. The skew between the REFCLK inputs to the transmitting devices bonded together should be a minimum. Similarly, the skew between the REFCLK inputs of the receiving devices bonded together should also be a minimum. Cypress HOTLink II devices were tested with a skew of 50 ps between the REFCLK inputs of the devices that were bonded.

Successful channel bonding requires that the data that is being transmitted fulfills certain requirements.

1. Check that $\overline{\text{LFI}}$ reports an inactive (HIGH) status in all receiving channels. This will ensure that all RX CDR PLLs are locked to the incoming data stream.
2. If low latency framer is used, framing should have been established by sending the appropriate framing character as defined by FRAMCHAR input. This will ensure that the characters that are received have the proper 10-bit boundaries before being decoded as 8-bit characters.
3. Bonding will be attempted each time at least four framing characters followed by a valid data character is received in all the channels in the same bonding domain. Note that this sequence should be received across all bonding channels within the deskew window, which is two characters. The bonding sequence could also be a word sync sequence followed by a valid data character. The word sync sequence is a sequence of sixteen K28.5 characters that can be generated from the transmitting devices. Refer to the data sheet for more details on the description of word sync sequences and the settings for generating them (based on TXMODE[1:0], TXCTx[1:0] and SCSEL).

In dual-channel bonded mode, when the transmitter is set in either TX Modes 5 or 8, word sync sequences can be generated at the same time in channels A and B when TXCTB[0] is sampled HIGH. Word sync sequences can be generated in channels C and D when TXCTD[0] is sampled HIGH.

In quad-channel bonded mode, when the transmitter is set at either TX Modes 5 or 8, word sync sequences can be generated across all four channels in the device when TXCTB[0] is sampled HIGH.

4. When RXCKSEL = LOW, due to minor frequency offsets between the reference clocks used for the transmit devices and receive devices, the CYP15G0401DXB may add or delete a framing character into the data stream in order to keep the receiver Elasticity Buffer from reaching an overflow or underflow state. When necessary, a framing character may be inserted after one is detected in the data stream and a framing character may be deleted when detected. It is necessary, therefore, to include enough framing characters interspersed within the data to tolerate the PPM difference between the REFCLK and the recovered clock. For one PPM difference, one framing character will be deleted or inserted into the datastream every million clock cycles. Therefore, the number of framing characters that need to be sent over every million clock cycles equals the maximum PPM difference between the transmit REFCLK and the receive REFCLK. If the selected framing character is also a special command (like a data packet delimiter) for upstream logic, an additional framing character should be sent along with each framing character to compensate for a possible deletion. When channel bonding is enabled, the addition and deletion of framing characters to compensate for PPM differences takes place across all channels at the same time.

Bonding Process Overview

The following few paragraphs discuss how the channel bonding process works and what status can be expected out of the RXSTx[2:0] for the corresponding receive channels. The assumption for this discussion is that the RXMODE[1:0] is selected to report Status-A.

At the detection of four consecutive framing characters on a receive channel, the specific channel goes to RESYNC state. During this state, the RXSTx[2:0] in that particular channel will report a 111. These channels continue to report a RESYNC while framing characters are continuously being received. If a channel that is in the RESYNC state receives a valid data character within the deskew window (two characters), then the channel goes to IN_SYNC state. If all channels move from RESYNC to IN_SYNC within the deskew window (two characters) all channel bonding will be established. In the IN_SYNC state, the status bits RXSTx[2:0] will report the status of the characters received in the stream (000–Valid Data, 001–Special code detected, 011–Framing character detected, 100–Codeword violation, 110–Running disparity error).

The behavior of the status bits when channel bonding is successful is shown in Figure 10 on page 11. In this example, each channel has a eight-bit counter output being transmitted with the same count value across each channel. Before each count cycle begins, six K28.5 characters (HEX value of K28.5 is “BC” when DECMODE is HIGH) are transmitted to enable channel bonding.

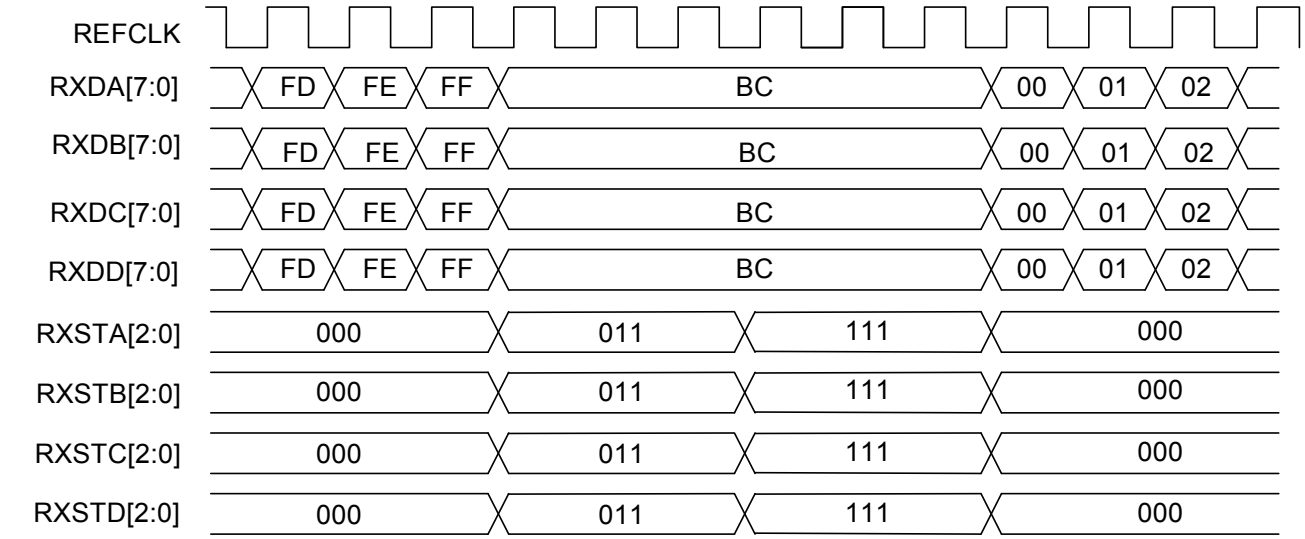
If the deskew window expires even in one channel before the arrival of a valid data character, then a 101 (COULD_NOT_BOND) is reported in the corresponding channel for one clock cycle. After the COULD_NOT_BOND condition, the RXSTx[2:0] for that channel will move to IN_SYNC state and begin to report the normal status of the data that is being received in that channel. For example, in a quad bonding case, suppose channels A, C, and D receive the bonding sequence before the deskew window expires, but channel B does not, then channel B will not bond with channels A, C, and D. The COULD_NOT_BOND status is reported for only one character clock cycle and after that the channel will go to the IN_SYNC state, even though it did not establish the channel bonding as shown in Figure 11.

The channels that are in the IN_SYNC state will bond according to the selected bonding configuration. For some of the channels, users may expect to see framing characters inserted or deleted to/from the bonding sequence as part of the deskew process. Note that this insertion or deletion of framing characters is independent of the insertion/deletion that happens to tolerate frequency offsets between the recovered clock and REFCLK. This insertion or deletion of framing characters from the bonding sequence in order to align the channels can also happen when the receive parallel data is clocked using one of the recovered clocks.

Table 5. Summary of Requirements for Channel Bonding

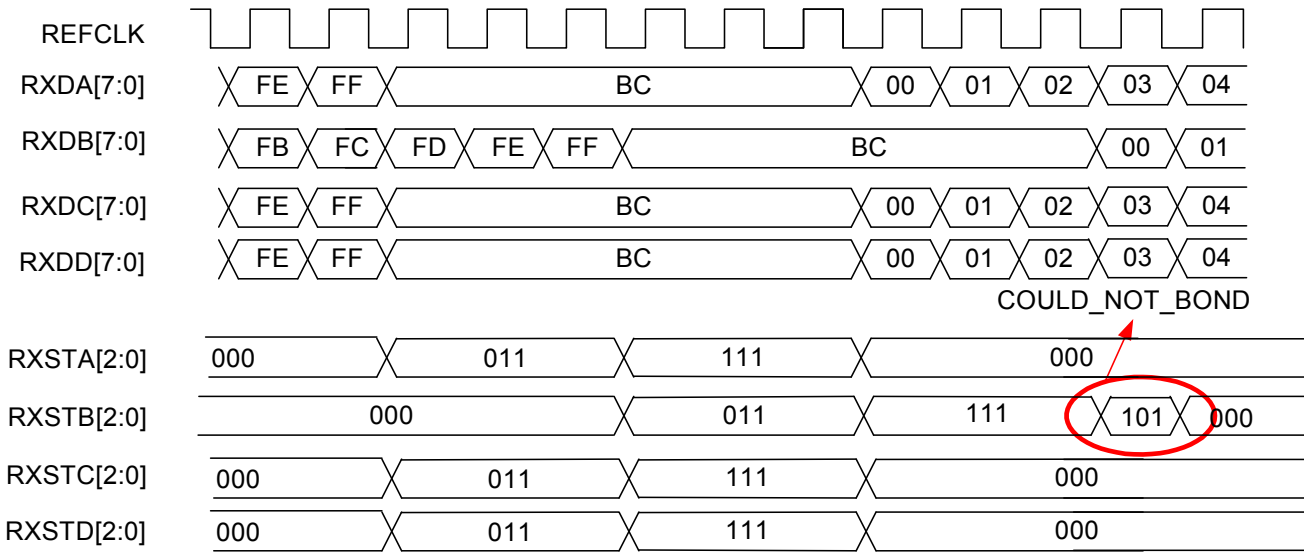
Parameter	Dual-channel Bonding	Quad-channel Bonding (Single Device)	Multidevice Bonding
Serial stream skew at receivers that are bonded	20 serial bit times	20 serial bit times	20 serial bit times
REFCLK skew across multiple devices that are bonded	Same REFCLK per bonded pair. Multi-device bonding not possible	Same REFCLK per bonded pair.	Skew between REFCLK inputs to multiple devices that are bonded together should be as low as possible. HOTLink II was tested with a skew of 500 ps at a clock frequency of 70 MHz. For more information on worst case timing constraints refer to APPENDIX B on page 13.
RXCKSEL setting	LOW or HIGH, depending on whether the RXCLKx follows REFCLK or Recovered clock	LOW or HIGH, depending on whether the RXCLKx follows REFCLK or recovered clock	LOW
BOND_ALL signal	Should be left open	Should be left open when RXCKSEL = HIGH When RXCKSEL = LOW, the BOND_ALL signal should be externally pulled up using a strong pull up like 36 ohms.	The BOND_ALL signals from all bonded devices should be connected together with a 75-ohm transmission line. The two ends of the transmission line should be terminated with 75-ohm pull up resistors. The length of the transmission line should be as short as possible and less than 12 cm. For more information on timing constraints introduced due to the transmission line delay refer to APPENDIX B on page 13.
BOND_ST[1:0] signals	Should be left open	Should be left open when RXCKSEL = HIGH When RXCKSEL = LOW, the BONDST[1:0] signals should be externally pulled up using a strong pull up like 36 ohms.	The BONDST[1:0] signals of all devices should be connected together using 75-ohm transmission lines. The receiving end(s) of these transmission lines should be terminated using a 75-ohm pull up resistor. The length of the transmission line should be as short as possible and less than 12 cm. For more information on timing constraints introduced due to the transmission line delay refer to APPENDIX B on page 13.
BOND_INH	Set to LOW, if bonding is desired only when bonding sequence is received across all channels that are desired to be bonded When HIGH, all channels that have detected a bonding sequence are allowed to align the elasticity buffer pipelines.	Set to LOW, if bonding is desired only when bonding sequence is received across all channels that are desired to be bonded When HIGH, all channels that have detected a bonding sequence are allowed to align the elasticity buffer pipelines.	Set to LOW, if bonding is desired only when bonding sequence is received across all channels that are desired to be bonded. All bonding devices should have same setting for this input. When HIGH, all channels that have detected a bonding sequence are allowed to align the elasticity buffer pipelines.
MASTER	Should be left open	Should be left open when RXCKSEL = HIGH When RXCKSEL = LOW, the MASTER pin should be driven LOW for Single Device Quad Channel Bonding.	One device per bonding domain is set as the MASTER by driving this pin LOW. All other devices in this domain will be slaves and should have this pin driven HIGH.

Figure 10. Behavior of Status Bits When Channel Bonding is Successful (Status Type A with $\overline{\text{BOND_INH}}$ = LOW)



RXDx are displayed as HEX values decoded with setting DECMODE=HIGH

Figure 11. Example of a Situation When Channel Bonding Fails (Status Type A with $\overline{\text{BOND_INH}}$ = LOW)



$\overline{\text{BOND_INH}}$ also plays an important role when the device is configured in the multichip bonding mode, since it impacts the behavior of the Status-A and Status-B state machine as described in the data sheet. Most channel bonding applications are strict on all channels being aligned and therefore prefer to select $\overline{\text{BOND_INH}}$ as LOW.

Summary

The requirements for different modes of channel bonding are summarized in Table 5 on page 10.

Appendix A

Figure 12. Optimal Layout of BOND_ALL Trace for Two-device Bonding

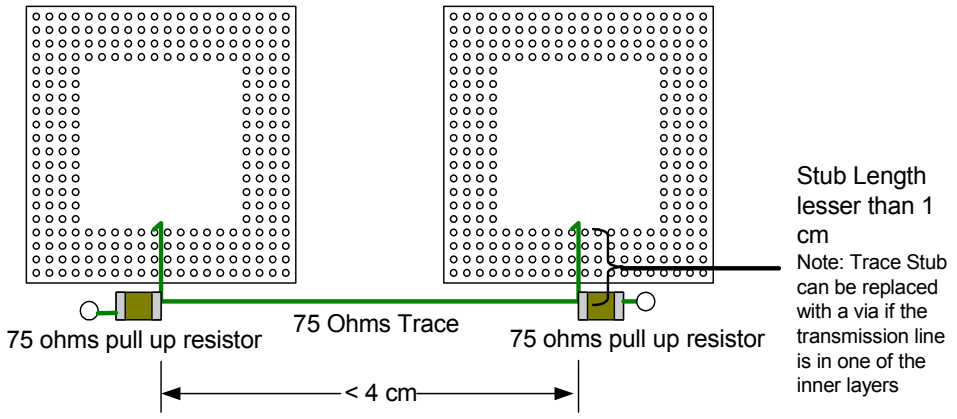
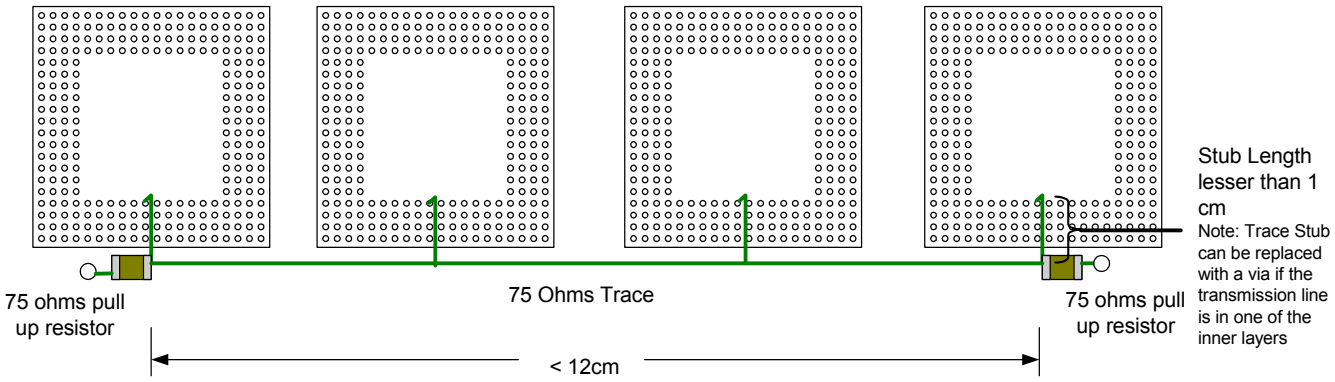


Figure 13. Optimal Layout for Channel Bonding Four Devices



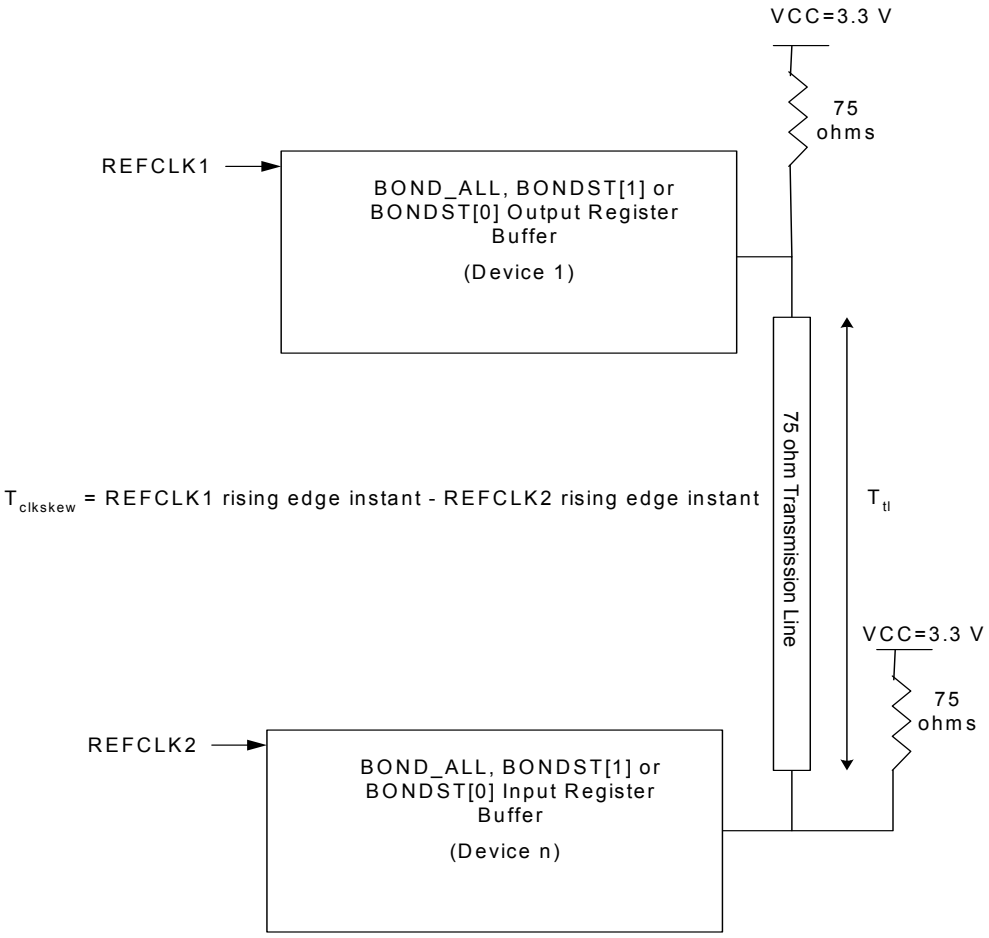
Appendix B

Timing Constraints for Multidevice Bonding

The last column of Table 5 on page 10 gives the typical requirements for successful multichannel bonding. This appendix gives the actual timing constraints for multidevice channel bonding.

Figure 14 shows the timing for the bidirectional open drain buffers in a multidevice bonding environment.

Figure 14. Timing Budget for Multidevice Bonding



Timing constraint for valid bonding communication from one device to another bonding device is

$T_{clkskew} + T_{tl} <$ Clock period – 5 ns.

$T_{clkskew}$ = REFCLK1 rising edge instant – REFCLK2 rising edge instant (inclusive of any jitter present).

T_{tl} is the delay due to the 75-ohm transmission line. For PCB's with FR4 dielectric, propagation delay per inch is 160 ps.

Note that the clock skew should also adhere to constraints posed by set-up and hold time of the upstream device that receives the bonding data with respect to RXCLKA or RXCLKC of either device. Refer to data sheet timing parameters $t_{REFADV–}$, $t_{REFADV+}$ or $t_{REFCDV+}$, $t_{REFCDV–}$ depending on whether RXCLKA or RXCLKC is used to clock the bonded data out of the device.

Document History

Document Title: Channel Bonding with HOTLink II™ - AN014
Document Number: 001-15051

Revision	ECN	Orig. of Change	Submission Date	Description of Change
**	1009340	NVNS	04/23/2007	Existing Application Note in the web - Added Spec No. and new disclaimer and also updated the copyright date Please post in the web- overwrite the existing AN014 file
*A	3248216	NVNS	05/04/2011	Updated in new template.
*B	4384630	NVNS	05/20/2014	No technical updates. Completing Sunset Review.

ESCON is a registered trademark of International Business Machines. HOTLink is a registered trademark, and HOTLink II is a trademark, of Cypress Semiconductor. All other trademarks or registered trademarks referenced herein are the property of their respective owners.

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
http://www.cypress.com

© Cypress Semiconductor Corporation, 2003-2014. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

This Source Code (software and/or firmware) is owned by Cypress Semiconductor Corporation (Cypress) and is protected by and subject to worldwide patent protection (United States and foreign), United States copyright laws and international treaty provisions. Cypress hereby grants to licensee a personal, non-exclusive, non-transferable license to copy, use, modify, create derivative works of, and compile the Cypress Source Code and derivative works for the sole purpose of creating custom software and or firmware in support of licensee product to be used only in conjunction with a Cypress integrated circuit as specified in the applicable agreement. Any reproduction, modification, translation, compilation, or representation of this Source Code except as specified above is prohibited without the express written permission of Cypress.

Disclaimer: CYPRESS MAKES NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Cypress reserves the right to make changes without further notice to the materials described herein. Cypress does not assume any liability arising out of the application or use of any product or circuit described herein. Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress' product in a life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.

Use may be limited by and subject to the applicable Cypress software license agreement.