XILINX®

XAPP697 (v1.2) January 7, 2005

Dynamic Phase Alignment Using Asynchronous Data Capture

Author: Catalin Baetoniu and Tze Yi Yeoh

Summary

This application note and its accompanying reference design describe a dynamic phase alignment (DPA) module used in bus interfaces, such as SPI 4.2, using asynchronous data capture techniques. The DPA module can run at 800 Mbps and faster in Virtex-II™ and Virtex-II Pro™ devices. It contains a word-alignment unit that can remove channel-to-channel skew. This document is an extension of XAPP671: High-Speed Data Recovery Using Asynchronous Data Capture Techniques.

Introduction

In high-speed source-synchronous applications, clock and data recovery is essential. The most prevalent method of clock and data recovery using Xilinx devices is oversampling incoming data with multiple phases of the DCM-generated clock. However, using the DCM as a means of data recovery is not always suitable at very high frequencies due to the DCM's inherent jitter. Additional jitter from the DCM causes a corresponding reduction in the data-valid window, limiting the highest possible performance. With asynchronous data capture techniques, data recovery can be performed without using the DCM, thus achieving higher performance. This application note describes a dynamic phase alignment (DPA) module used in a 16-channel SPI 4.2 interface designed using asynchronous data capture techniques. The design techniques described here can be used in any source-synchronous bus applications.

DPA Reference Design Details

Design Facts

Table 1 lists the performance, resource utilization information, and implementation requirements for the DPA reference design (xapp697.zip).

Table 1: **DPA Reference Design Specifics**

Performance	Up to 800 Mbps Double Data Rate (DDR) in Virtex-II FPGAs. Up to 1 Gbps DDR in Virtex-II Pro FPGAs
FPGA Device Implemented	Virtex-II XC2V1000-5 FF896
Resource Utilization	954 Virtex-II slices 1 Block RAM 3 BUFGMUXs
HDL Used	VHDL
Synthesis Tool	Synplify 7.3.4(1)
Implementation Tool	Xilinx ISE 6.1i
Simulation Tool	Modelsim 5.8

1. Note that the reference design uses attributes and directives specific to Synplify. Porting the design to other synthesis tools might require significant code modifications.

Design Overview

The XAPP671 and XAPP697 reference designs illustrate using asynchronous data capture to perform data recovery and alignment. The basic components in these reference designs are:

- Data sampling delay lines
- Data recovery state machine
- Word-alignment unit

Figure 1 shows a high-level view of the DPA reference design.

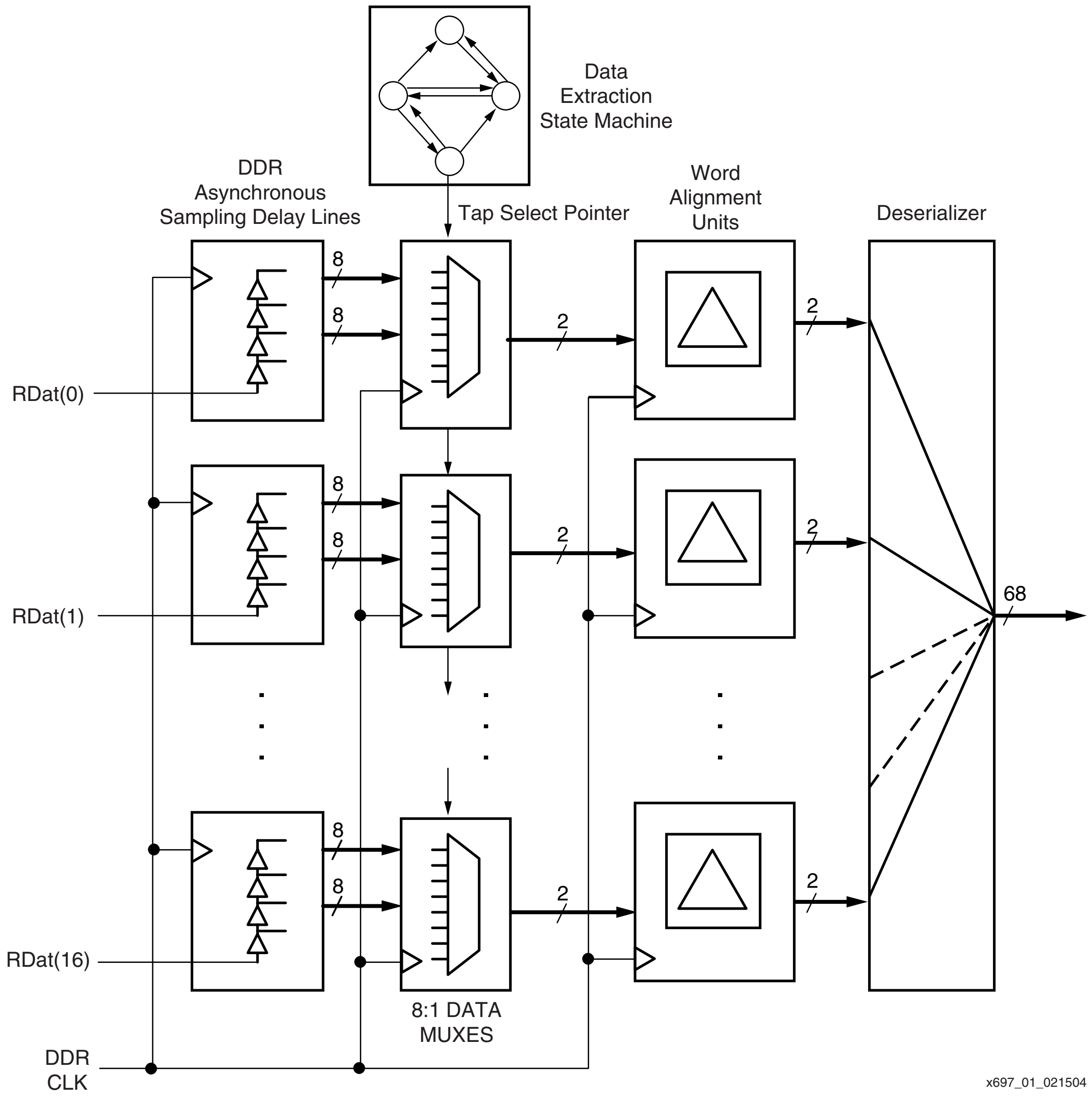
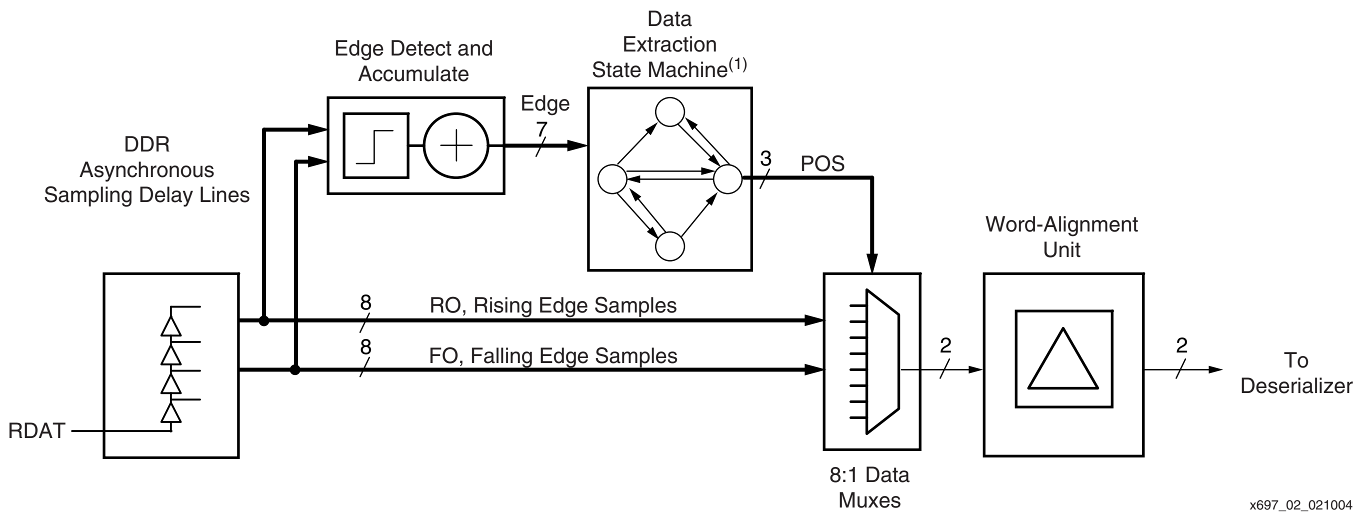


Figure 1: **Top-Level Block Diagram**

Figure 2 shows the block diagram for a single channel.



Notes:

1. In the actual reference design, there is only one data extraction state machine time-shared between all channels. The state machine shown here is for illustrative purposes only.

Figure 2: **Block Diagram for One Channel**

In source-synchronous applications, such as SPI 4.2, the transmitting device forwards a clock along with the data. The DPA module uses the forwarded clock to sample delayed versions of incoming data using a multitap delay line. The data recovery state machine continuously selects the valid sample using edge information from the multitap delay line and forwards the correct sample to the word-alignment unit. The word-alignment unit polls the incoming data, looking for a specific training pattern. In this design, the SPI 4.2 training pattern (10 zeros followed immediately by 10 ones or vice versa) is assumed. When the training pattern is found, the word-alignment unit then can deskew the channels with respect to each other.

Package Files

The package file names for the reference design are as follows:

- `dpa2_v2_pkg.vhd` (Virtex-II devices and packages)
- `dpa2_v2p_pkg.vhd` (Virtex-II Pro devices and packages)

Design Hierarchy

Figure 3 shows the hierarchy of the reference design.

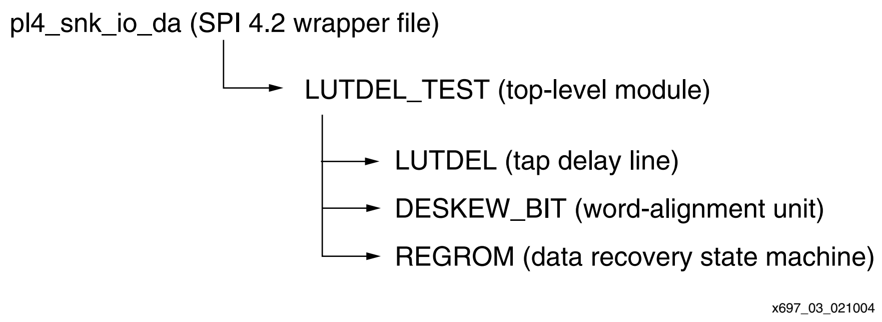

pl4_snk_io_da (SPI 4.2 wrapper file)
LUTDEL_TEST (top-level module)
LUTDEL (tap delay line)
DESKEW_BIT (word-alignment unit)
REGROM (data recovery state machine)
x697_03_021004

Figure 3: **Reference Design Hierarchy Structure**

XAPP671 introduced the concept of using asynchronous data capture techniques for data recovery designs. Its reference design was a 32-independent channel backplane design running up to a maximum data transfer rate of 622 Mbps. Table 2 summarizes the differences between XAPP671 and XAPP697.

Table 2: **Differences Between XAPP671 and XAPP697**

XAPP671	XAPP697
Up to 622 Mbps only	Up to 1 Gbps in Virtex-II Pro devices Up to 800 Mbps in Virtex-II devices
Non-interleaved tap delay line	Selectable interleaved or non-interleaved tap delay line
No channel-to-channel alignment	Word-alignment unit enables channel-to-channel alignment
Does not support bus interfaces	Supports bus interfaces such as SPI 4.2
Each channel has its own state machine (16 Block RAMs used)	One state machine (1 Block RAM) used for the entire design
Designed for asynchronous data transfer (clock is asynchronous to data)	Designed for quasi-synchronous environment (clock is relatively synchronous to data)

Detailed Block Descriptions

Data Sampling Delay Lines

The input data for each channel is sampled asynchronously using 8-tap delay lines. There are two delay lines per channel for DDR operation: one for rising-edge samples and one for falling-edge samples. Each delay line is built from eight look-up tables (LUTs) configured as inverters, ensuring symmetry between rise and fall times and ensuring that the data samples are regularly spaced from each other. The input nets from the input/output blocks (IOBs) to the inputs of the two delay lines and the nets connecting the LUTs in each delay line are the only asynchronous portion of the design. The input net from the IOB must reach both delay lines with very little skew. Consistent routing of these nets for all channels is ensured using directed routing constraints and RLOC constraints for the LUT components.

The timing analysis that determines the tap delays and length of the delay chain is based on two basic timing constraints:

- Best-case timing constraint

  The best-case timing constraint is the minimum value for the entire delay line. It must be greater than the amount of data jitter. This constraint determines the number of taps. The best-case timing constraint is expressed mathematically as:

$$t_{JIT} < t_{TAPMIN} \times ((n-2k)+1) \qquad \text{Eq. 1}$$

- Worst-case timing constraint

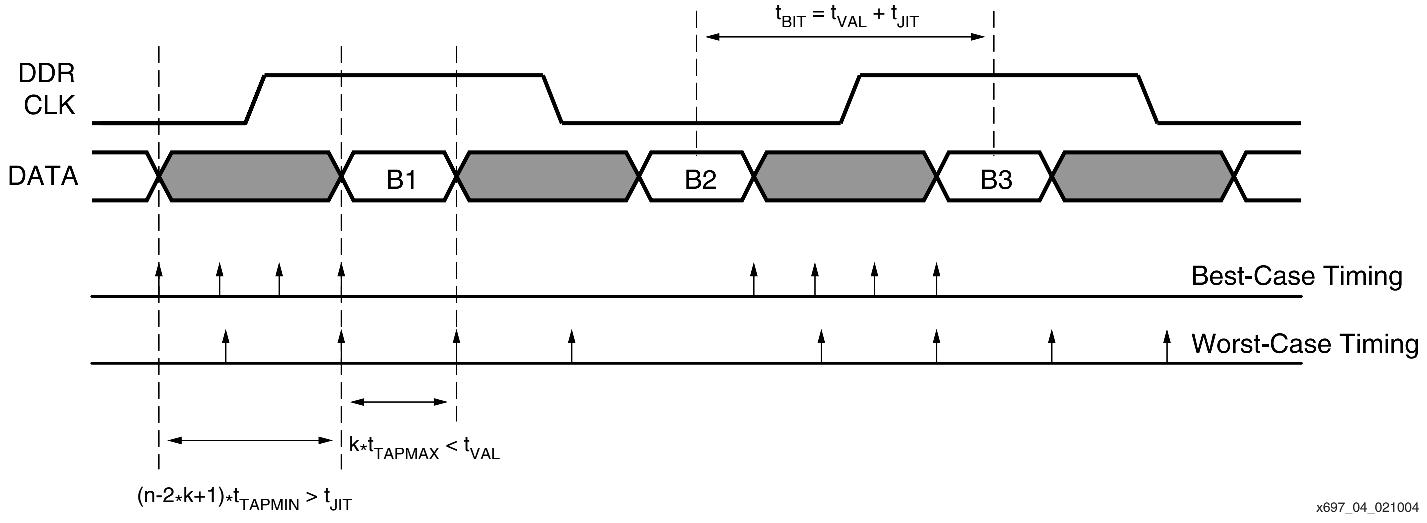
  The maximum value for the tap delay must be less than the data valid time window (equivalent to the Nyquist sampling theorem for analog signals). The worst-case timing constraint determines the selection of the delay line elements. The worst-case timing constraint is expressed mathematically as:

$$t_{VAL} > k \times t_{TAPMAX} \qquad \text{Eq. 2}$$

In this example, *n* is the number of taps, *k* is the minimum number of data samples in the data valid time window, t_{TAPMIN} is the required minimum tap delay, t_{TAPMAX} is the required maximum tap delay, t_{JIT} is the amount of jitter, and t_{VAL} is the data valid time window.

The minimum and maximum tap delay values, t_{TAPMIN} and t_{TAPMAX}, are the result of the derating factor causing a variation in the delay values used in the analysis. Derating factors occur due to variations in process, voltage, and temperature (PVT) and are assumed to be approximately 40%. An example timing analysis is provided in "Appendix A: Timing Analysis."

Figure 4 illustrates the best-case and worst-case constraints.


$t_{BIT} = t_{VAL} + t_{JIT}$
DDR CLK
DATA
B1
B2
B3
Best-Case Timing
Worst-Case Timing
$k * t_{TAPMAX} < t_{VAL}$
$(n-2*k+1)*t_{TAPMIN} > t_{JIT}$
x697_04_021004

Notes:

1. In this figure, n = 4 (the number of taps) and k = 1 (the minimum number of valid data samples).

Figure 4: **Tap Delay Line Timing Constraints**

There are two basic tap delay lines:

- Non-interleaved tap delay lines
- Interleaved tap delay lines

Figure 5 shows a non-interleaved tap delay line. In a non-interleaved tap delay line, the LUTs are chained together one after another in a straightforward fashion. Therefore the tap delay is the delay through the LUT (data sheet parameter t_{ILO}) plus the interconnect delay (t_{NET}). Typically, the interconnect delay is insignificant compared to the LUT delay, so the tap delay is effectively $t_{ILO.}$ In Virtex-II and Virtex-II Pro devices, t_{ILO} is typically 400 ps. Depending on the input conditions (for example, data transfer rate, input jitter), a non-interleaved delay line runs substantially slower than an interleaved delay line.

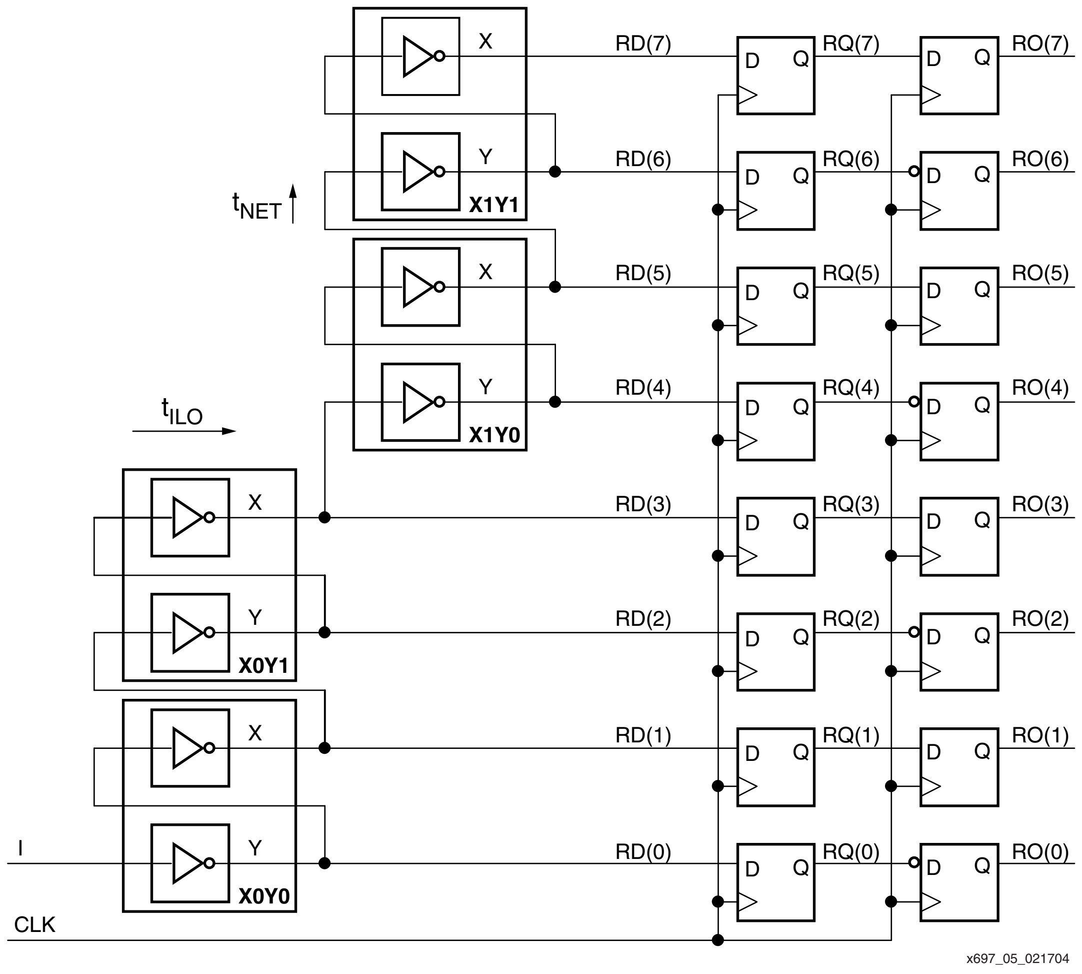


Figure 5: **Non-interleaved Tap Delay Line**

Figure 6 shows the interleaved tap delay line. In an interleaved tap delay line, two separate delay lines are constructed. The second line is routed such that the input interconnect delay is equal to ½ t_{ILO}. In this case, the tap delay is effectively ½ $t_{ILO.}$ Therefore, an interleaved delay line yields a tap delay of 200 ps in Virtex-II and Virtex-II Pro devices. A smaller tap delay generally allows the DPA module to capture data at a much higher rate and is more tolerant of greater amounts of input jitter.

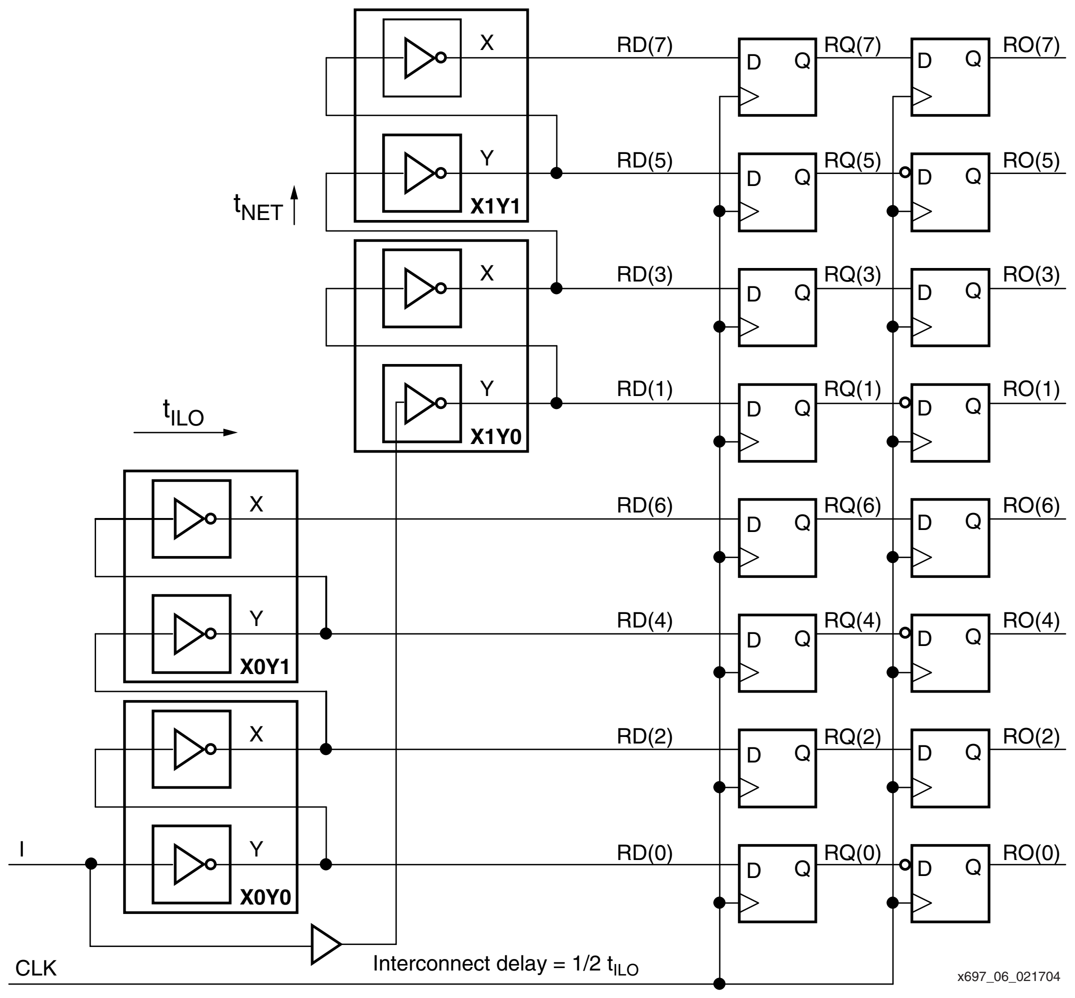


Figure 6: **Interleaved Tap Delay Line**

Another design practice is to add an extra delay element in front of the tap delay line. This method gives the user a way to dynamically add a constant delay to the tap delay line to shift the entire delay line to a different bit area.

In this reference design, the user can select between non-interleaved and interleaved tap delays using the SEL input port. Setting SEL = 0 selects the non-interleaved delay line, and setting SEL = 1 selects the interleaved delay line. In addition, the user also can add an extra delay at the front end of the tap delay line by setting DEL = 1, which shift the delay line to a different bit area. *This option is only available in non-interleaved mode (when SEL = 0)*. A MUXCY element is used as the additional delay element. The data sheet delay through the MUXCY element is approximately 580 ps. Table 3 summarizes the functions of the SEL and DEL signals.

Table 3: **Descriptions of the SEL and DEL Signals**

Signal	Function
SEL	Selects interleaved or non-interleaved tap delay line. SEL = 0: Non-interleaved tap delay line SEL = 1: Interleaved tap delay line
DEL	Selects extra constant delay to be added to front end of tap delay line. *Available only in non-interleaved mode (when SEL = 0)*. DEL = 0: No constant delay added DEL = 1: Constant delay added

Figure 7 shows the block diagram for the tap delay line used in the DPA reference design.

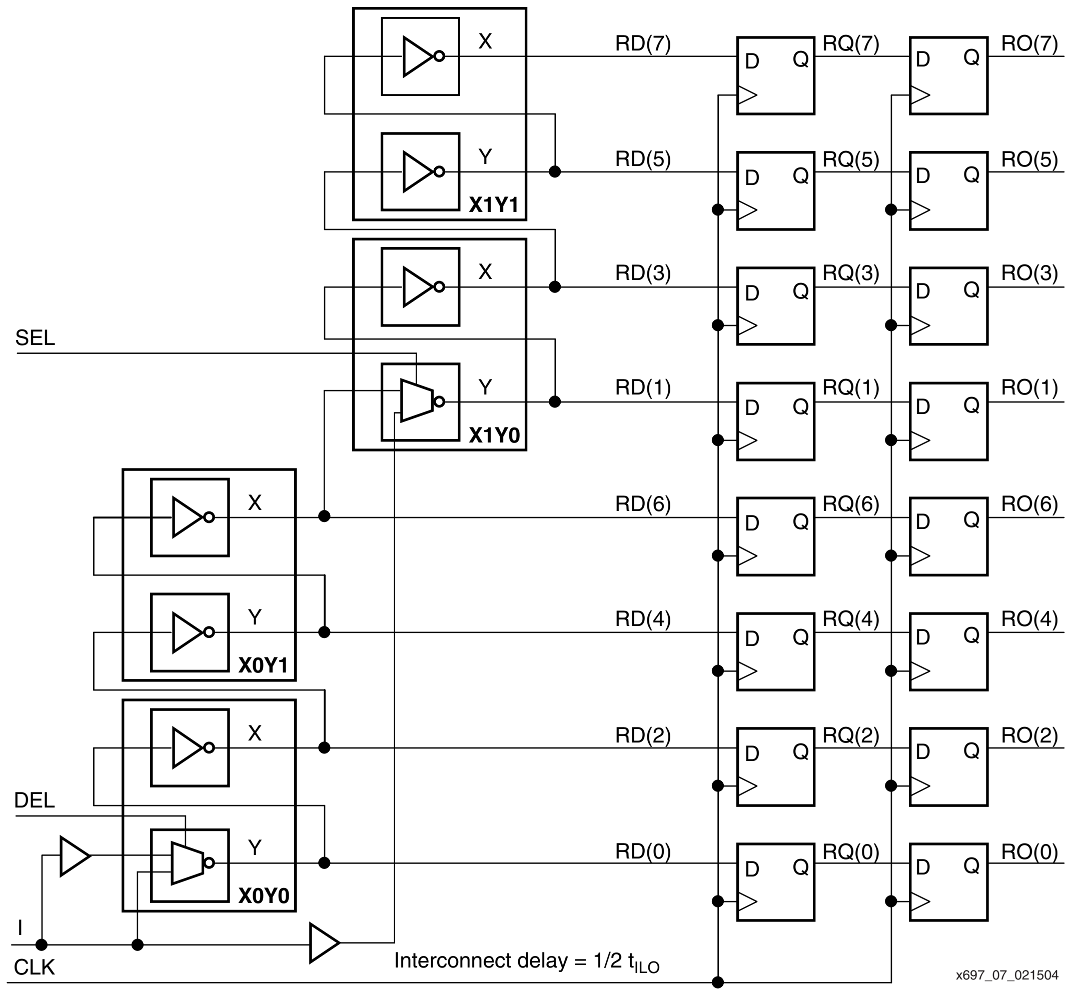


Figure 7: **Block Diagram of the Selectable Tap Delay Line used in the Reference Design**

An improper relative position of an IOB pair to a Relative Placement Macro (RPM) tile can lead to a different routing for this net. If the delays for the two loads of this net are not equal, the design will not work. This situation is indicated through directed routing constraint failures. It is important to check the place-and-route report after each implementation.

Data Recovery State Machine

As data is sampled by the tap delay lines, the data recovery state machine processes these samples and ultimately produces valid output bits. The DPA reference design has just one state machine. The edge-detect mechanism is achieved by rippling the individual edge-detect results across all the channels. Each channel creates its own edge-detect samples by pairwise XORing the eight delay line samples from both the rising-edge and falling-edge tap delay lines. In the process, the falling-edge signals are moved from the CLK180 domain into the CLK clock domain. Then the edge detects for the rising-edge and the falling-edge samples are ORed together with the edge-detect samples from the previous channel, and the result is forwarded to the next channel. At the last channel, the final result is forwarded to the data recovery state machine. Each time there is a data transition, either in the rising-edge samples or the falling-edge samples, at least one bit in the edge-detect samples is set.

The state machine operates on a simple principle of always staying at least one sample away from the edge of the data valid window. During reset, the state machine positions the POS pointer, which selects the valid data sample, to the center of the tap delay line. During normal operation, jitter and PVT conditions might cause the data valid window to drift. The state machine continuously polls the edge-detect samples forwarded to it by the edge-detect mechanism and determines the position of the POS pointer relative to the edge of a data valid window. If the POS pointer strays too close to the edge, the state machine adjusts the POS pointer by moving one step to the left or right as appropriate to move away from the edge. The state machine then updates the POS pointer and feeds it back into the tap delay lines. Two 8:1 multiplexers use the POS signal to select one rising-edge sample and one falling-edge sample as the valid data output.

The actual state machine is implemented in block RAM. Block RAM can implement any state machine behavior with the same number of inputs and outputs using the same low number of logic resources without affecting the device utilization, routing, and timing performance.

In the DPA reference design, the state machine is encapsulated in the REGROM module. Table 4 summarizes the four output signals of the state machine.

Table 4: **State Machine Output Status Signals**

Signal	Width	Function
POS	3	Position pointer sent to the tap delay lines to select the valid data sample.
SKIP_LEFT	1	When asserted, this status signal indicates that the state machine has skipped the position pointer to the left. This signal may be safely ignored in a quasi-synchronous application, such as SPI 4.2.
SKIP_RIGHT	1	When asserted, this status signal indicates that the state machine has skipped the position pointer to the right. This signal may be safely ignored in a quasi-synchronous application, such as SPI 4.2.
ERROR	1	When asserted, this status signal indicates that the position pointer no longer rests in the data valid window. The state machine needs to be reset.

Word-Alignment Unit

The data recovery state machine can accurately recover data as long as the skew does not exceed the bit boundary. In a real system, board routing inconsistencies might cause channel-to-channel skew that exceeds the bit boundary. This situation is known as a *bitslip* condition. One way to remedy this problem is for the transmitting device to send a pre-arranged training pattern to the receiving device during the initialization phase. Based on the sequence of bits received from the transmitting device and comparing that sequence to the pre-arranged pattern, the receiving device can deskew the channels with respect to each other and eliminate the bitslip condition. In the SPI 4.2 protocol, there are two pre-arranged training patterns that the receiving device needs to detect: 10 zeros followed by 10 ones and vice versa. The word-alignment unit in this reference design (DESKEW_BIT) was designed to recognize both SPI 4.2 training patterns.

Figure 8 shows a detailed block diagram of the word-alignment unit.

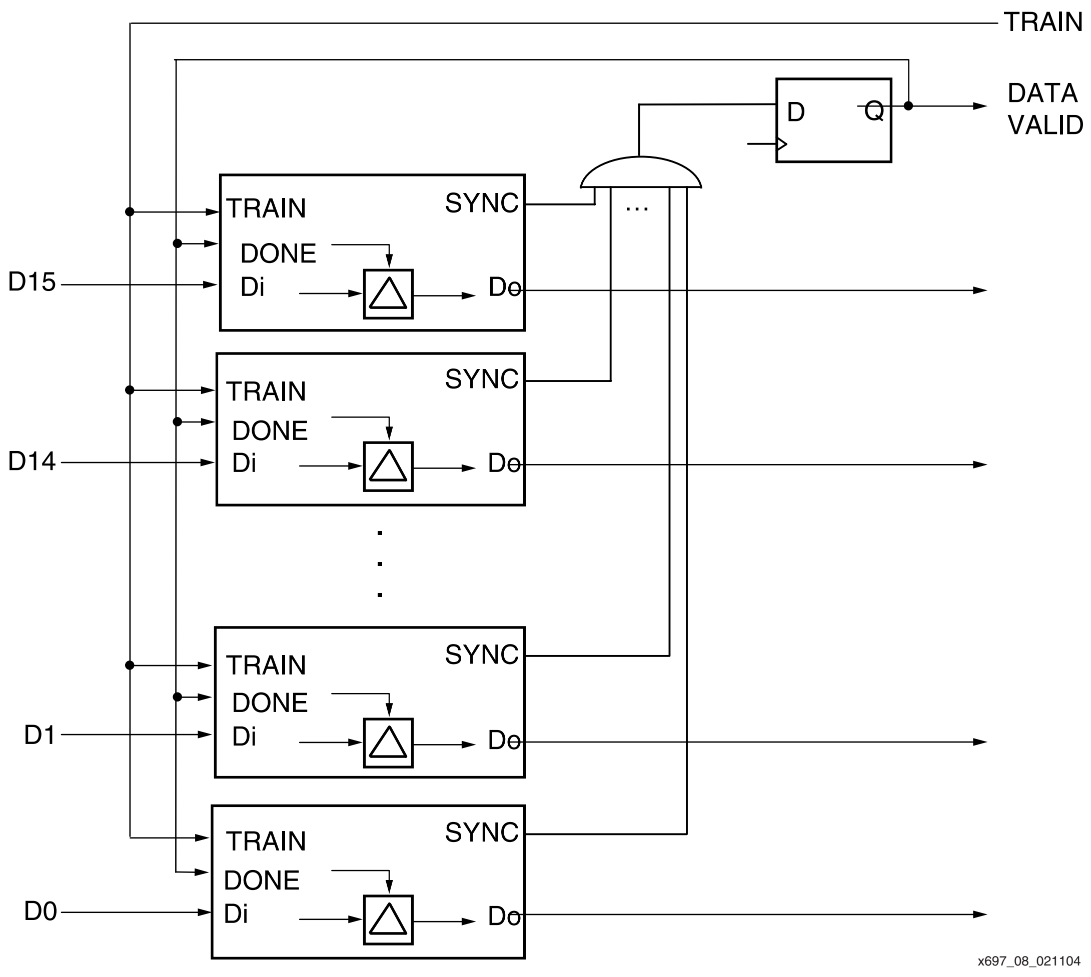


Figure 8: **Word-Alignment Unit**

During the initialization phase, the SPI 4.2 receiver core must instruct the transmitter to start sending the training pattern and assert the TRAIN signal for at least one clock cycle. This puts the DPA module in training mode and activates the word-alignment unit. Once activated, the word-alignment unit starts looking for the SPI 4.2 training pattern. The pattern is determined by the UP parameter. When UP = TRUE, the word-alignment unit looks for 10 zeros followed by 10 ones. When UP = FALSE, it looks for 10 ones followed by 10 zeros. In the SPI 4.2 protocol, channels 12 through 15 are configured with UP = FALSE and the remaining channels have UP = TRUE. When each channel has found the training pattern, the word-alignment unit pulls the SYNC signal High. A module in the top level (LUTDEL_TEST) detects all SYNC signals and pulls DONE High after all SYNC signals are pulled High. The word-alignment unit detects the assertion of the DONE signal and, using the time difference between SYNC and DONE going High, determines how to correct for misalignment. When the channels are completely deskewed, the DVAL signal in the wrapper file is asserted.

Modifying the Design

Using a Different Pinout on the Left / Right Side (Banks 2, 3, 6 and 7)

The DPA reference design is implemented in a Virtex-II XC2V1000-5 FF896 device. The design occupies the top left hand side of the device with input IOBs occupying the left banks (Bank 7). Each DPA channel is one configurable logic block (CLB) high and about 13 CLBs wide. Figure 9 shows the floorplanner view of the design.

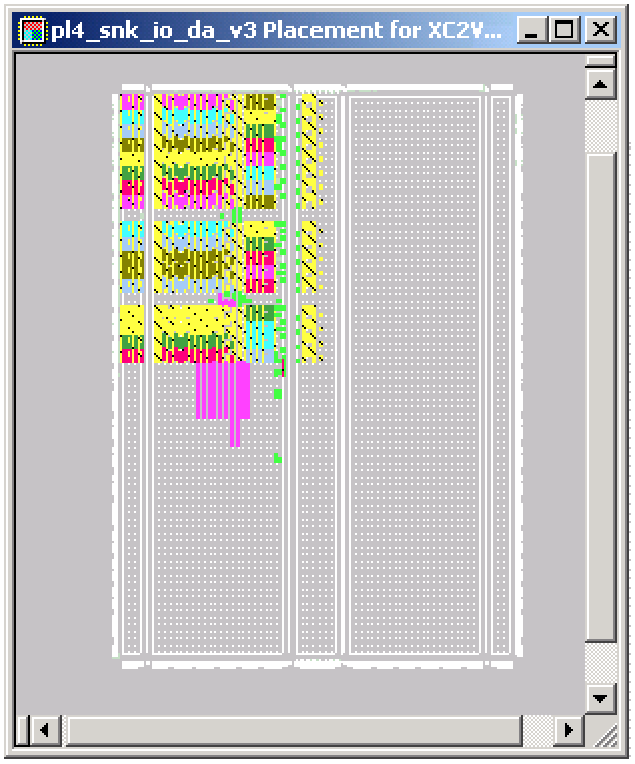


Figure 9: **Floorplanner View of DPA Reference Design**

Table 5 shows the parameters and functions that control the placement and routing of the channels. The parameters are passed down from the LUTDEL_TEST module. Some parameters are controlled by functions located in the package files (see “Package Files”).

Table 5: **Parameters / Functions that Control Placement and Routing of DPA Channels**

Parameter	Function	Values	Purpose
-	YLOC_TOP	Integer	Controls the placement of each channel with respect to the OFFSET parameter
-	OFFSET	Integer	Sets the absolute position of the channels
-	XOFFSET	Integer	Controls the x-coordinates of the design elements outside of the channels
-	PERMUTE	{16…0}	Determines the ordering of the channels
HIGH	HLOC	{0,1}	Determines whether the upper or lower pair of IOBs in the CLB row is used
SKIP_ROW	SKIP_ROW_LOC	{-1,0,1}	Used to indicate a “skip row” condition where the IOBs used are not in the same CLB row as the DPA channel. {0} indicates an IOB in the same CLB row. {-1} indicates an IOB that is one CLB row below the DPA channel. {1} indicates an IOB that is one CLB row above the DPA channel.
LEFT	-	{TRUE, FALSE}	Determines whether the design is placed on the left or right side of the device.

Whenever possible, position the pinouts on the left and right sides of the chip because the elements within the DPA channel are constrained with relative location constraints (RLOC) that remain valid no matter which CLB row it occupies. Also, since routing resources are identical for each CLB row with few exceptions (the row that contains the PowerPC™ multicontroller in Virtex-II Pro devices is a notable exception), almost all directed routing constraints in the DPA channel are valid as well. The user must create new directed routing constraints for the net that connects the IOB pads with the input of the tap delay line (this is the RDat data bus in the reference design). When the user manually routes the net, a Double interconnect must be used for connecting the rising-edge and falling-edge tap delay lines together to ensure consistent timing between both the tap delay lines. More importantly, for the interleaved tap delay line to work, the routing to the second half of the tap delay line must be approximately ½ t_{ILO} longer than the routing to the first half of the tap delay line.

Figure 10 shows the routing of the RDat data bus in the reference design.

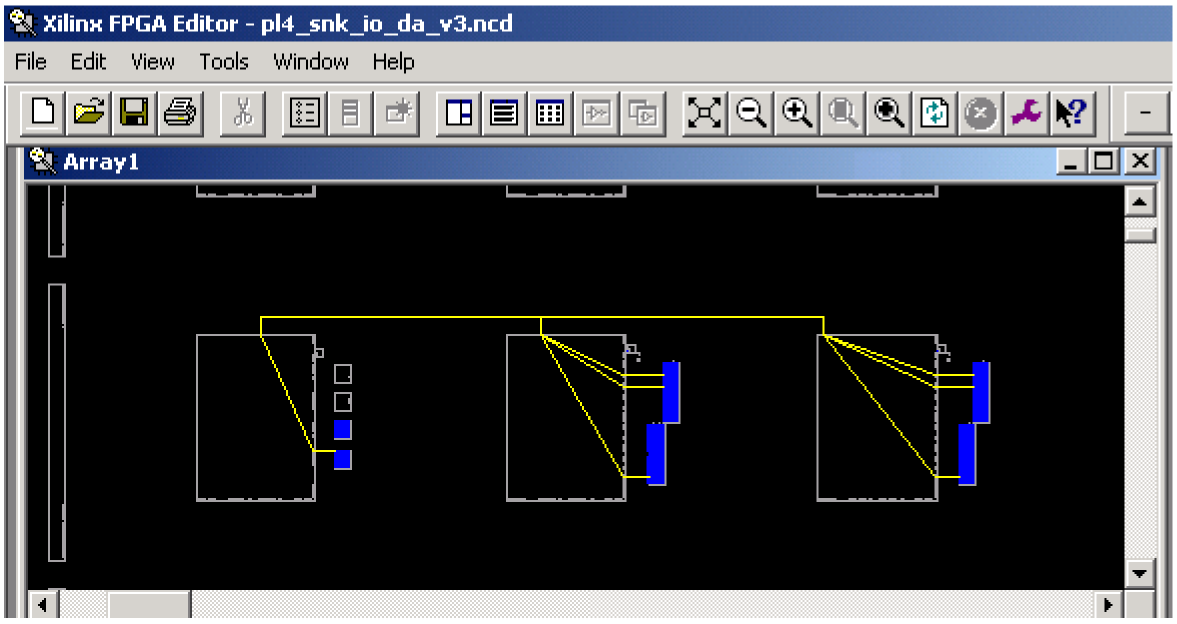

Xilinx FPGA Editor - pl4_snk_io_da_v3.ncd
File Edit View Tools Window Help
Array1
x697_10_021104

Figure 10: **FPGA Editor View of the Routing of the RDat Signal**

The delay from:

- the IOB to the first half of the tap delay line is 280 ps
- the IOB to the second half of the tap delay line is 444 ps

The difference between the two delays is 160 ps, or approximately ½ t_{ILO}.

Top / Bottom Banks (Banks 0, 1, 4, and 5)

Relocating the tap delay lines to the top and bottom banks is a much more difficult task than the left and right banks. To run at full speed (800 Mbps and above), new location and directed routing constraints must be found for almost every design element in the DPA channel. For slower speeds (644 Mbps and below), the strategy is to use location and directed routing constraints only for the asynchronous part of the design (the actual tap delay line). For the synchronous part of the design, the user can use area constraints to meet timing requirements.

Mixed (Left / Right and Top / Bottom) Implementation

This modification is not recommended. A workable implementation would waste a lot of CLB resources due to awkward placement of the DPA channels.

Slower Speeds

For slower speeds (644 Mbps and below), the strategy is to use location and directed routing constraints only for the asynchronous part of the design (the actual tap delay line). For the synchronous part of the design, the user can use area constraints to meet timing requirements.

Conclusion

In many high-speed parallel bus interfaces, data recovery techniques using asynchronous data capture are a necessity. Asynchronous data capture is an alternate method of performing data recovery without the use of a DCM. XAPP671 introduces the concept of using asynchronous data capture techniques for data recovery. This application note describes an implementation of a Dynamic Phase Alignment module (DPA) for a SPI4.2 interface. The reference design can run at 800 Mbps and faster in Virtex-II and Virtex-II Pro device families and can do channel-to-channel deskewing via a training protocol.

Appendix A: Timing Analysis

As described on page 4, the best-case and worst-case timing constraints determine the tap delays and length of the delay chain. Figure 4, page 5 illustrates the two constraints. This example has an input data rate of 800 Mbps and data-to-clock jitter of 0.50 UI. Based on the information given, the total bit time is 1250 ps, the data valid window is 625 ps, and jitter time is 625 ps. This example uses an interleaved LUT delay line with t_{ILO} = 396 ps and t_{NET} = 4 ps.

The t_{TAPMAX} and t_{TAPMIN} parameters are calculated as follows:

$$t_{TAPMAX} = (t_{ILO} + t_{NET})/2 = 200 \text{ ps}$$

$$t_{TAPMIN} = t_{TAPMAX} \times 40\% \text{ (using a 40\% derating factor)} = 80 \text{ ps}$$

Starting with the worst-case conditions, the parameters *n* and *k* are calculated based on t_{TAPMAX}. Because the edge-detect state machine requires at least two valid data samples, *k* = 2. Evaluating Equation 2 (page 4) yields:

$$t_{DATAMIN} > t_{TAPMAX} \times 2, \text{ or}$$

$$625 \text{ ps} > 400 \text{ ps}$$

The margin is 225 ps, which is at least two valid data samples. Therefore, the worst-case timing constraint is met.

The best-case timing constraint is evaluated next. Using an 8-tap delay line with Equation 1 (page 4) yields:

$$t_{JITMAX} < t_{TAPMIN} \times 5, \text{ or}$$

$$625 \text{ ps} \nless 400 \text{ ps}$$

Because the best-case condition is not met with an interleaved delay line, the noninterleaved mode must be used in this case, where t_{TAPMIN} is 160 ps:

$$t_{JITMAX} < t_{TAPMIN} \times 5, \text{ or}$$

$$625 \text{ ps} < 800 \text{ ps}$$

Switching between interleaved and noninterleaved modes can be done automatically using a ring oscillator to measure the actual tap delay. The best- and worst-case equations do not have to be met at the same time because the derating factor spread due just to voltage and temperature is much smaller than the total PVT variation range.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/26/04	1.0	Xilinx Confidential version.
03/24/04	1.1	Changed the best-case timing constraint example in Appendix A: Timing Analysis.
01/07/05	1.2	Initial Xilinx release.