XILINX®

XAPP774 (v1.2) February 23, 2006

Connecting Xilinx FPGAs to Texas Instruments ADS527x Series ADCs

Author: Marc Defossez

Summary

This application note describes how to connect a high-speed Texas Instruments (TI) ADS5273 analog-to-digital converter (ADC) with serialized LVDS output to a Virtex™-II or Virtex-II Pro FPGA. Lower speed ADC devices from this family can be connected to Spartan™-3 FPGAs.

Introduction

Texas Instruments has an 8-channel, 12-bit ADC family with synchronous LVDS outputs. The performance rates of these ADCs range from 40 MSPS to 70 MSPS. This family fits nicely with the LVDS I/Os of the Virtex-II, Virtex-II Pro, and Spartan-3 devices.

To highlight the performance of both the ADC and the FPGA, the reference design described in this application note uses the ADS5273, which is the highest speed sampling ADC. The ADS5273 interfaces to an XC2V250-6FG256 device (to fit the Texas Instruments demo board) and to an XC2VP20-6FF896 device (to fit Xilinx demo boards).

ADS527x ADCs

The ADS527x components are 8-channel, 12-bit analog-to-digital converters with serialized LVDS interfaces and sample rates from 40 MSPS to 70 MSPS. This section summarizes their features and applications. For complete information, refer to the TI website at:

http://focus.ti.com/analog/docs/analoghomepage.jsp?templateId=1&familyId=2&navigationId=9628

Features

The key features of the ADS527x family are:

- Maximum sample rate of 40 MSPS (ADS5279), 50 MSPS (ADS5271), 65 MSPS (ADS5272), and 70 MSPS (ADS5273)
- 12-bit resolution
- No missing codes
- Power dissipation of 1.1W
- CMOS technology
- Simultaneous sample and hold
- 70.5 dB signal-to-noise ratio (SNR) on a 10 MHz interface
- Serialized LVDS outputs meet or exceed the requirements of the ANSI TIA/EIA-644-A standard
- Internal and external references
- 3.3V digital/analog supply
- TQFP-80 PowerPAD package

© 2004–2006 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and further disclaimers are as listed at http://www.xilinx.com/legal.htm. All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

NOTICE OF DISCLAIMER: Xilinx is providing this design, code, or information "as is." By providing the design, code, or information as one possible implementation of this feature, application, or standard, Xilinx makes no representation that this implementation is free from any claims of infringement. You are responsible for obtaining any rights you may require for your implementation. Xilinx expressly disclaims any warranty whatsoever with respect to the adequacy of the implementation, including but not limited to any warranties or representations that this implementation is free from claims of infringement and any implied warranties of merchantability or fitness for a particular purpose.

Applications

The ADS527x ADC family is intended to be used in the following applications:

- Medical equipment
- Ultrasound systems
- Tape drives
- Test equipment
- Communications
- Optical networking

Implementation Description

Eight differential inputs are sampled at a 70 MHz clock rate. The eight ADC channels can use either an internal or an external reference. Use the internal reference for simplified design and best results.

The ADC has eight LVDS outputs, each one providing a serial 12-bit data bitstream with either MSB or LSB first. An LVDS high-speed clock output (LCLK) and an LVDS delayed repowered sampling clock (ADCLK) are also provided.

The high-speed bit clock, LCLK, is six times the ADCLK sampling clock.

Figure 1 shows the relationship between the different clocks and the output data.

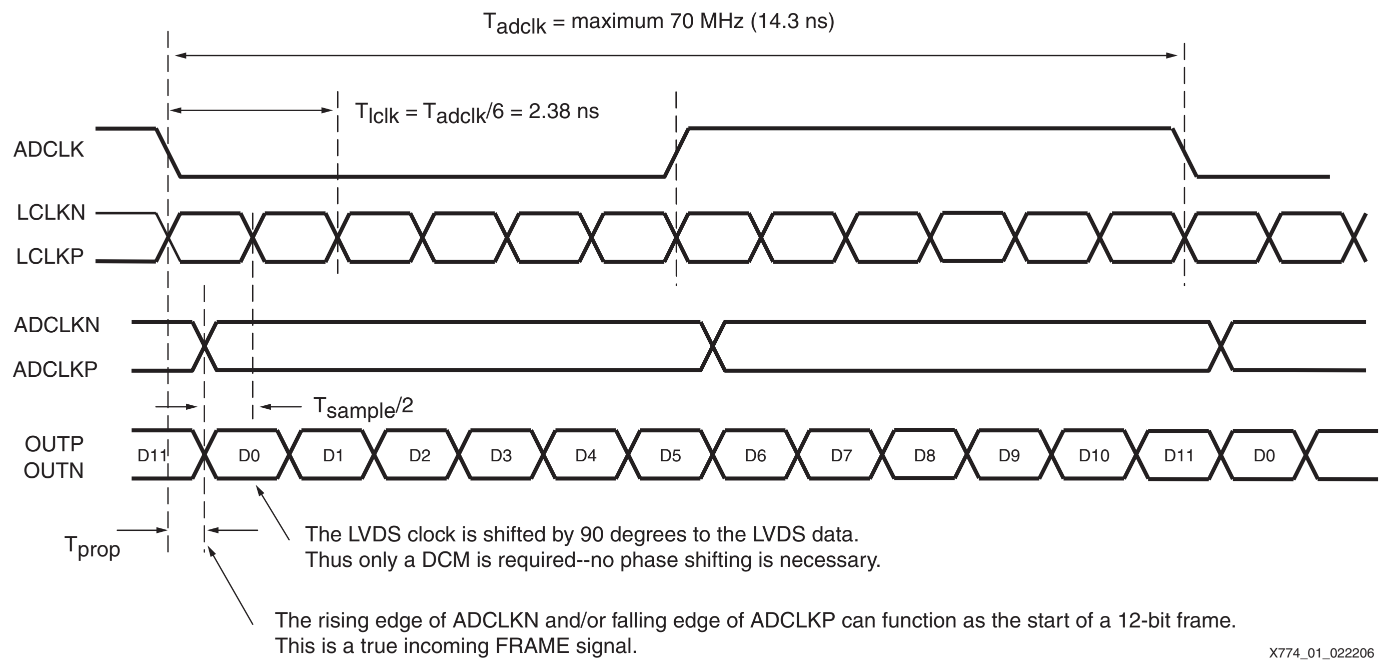
T_{adclk} = maximum 70 MHz (14.3 ns)
$T_{lclk} = T_{adclk}/6$ = 2.38 ns
ADCLK
LCLKN
LCLKP
ADCLKN
ADCLKP
$T_{sample}/2$
OUTP
OUTN
D11
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D0
T_{prop}
The LVDS clock is shifted by 90 degrees to the LVDS data.
Thus only a DCM is required--no phase shifting is necessary.
The rising edge of ADCLKN and/or falling edge of ADCLKP can function as the start of a 12-bit frame.
This is a true incoming FRAME signal.
X774_01_022206

Figure 1: **LVDS Timing Diagram (One Channel)**

A set of ADC registers can be read or written through an SPI/Microwire (I^2C look-alike) protocol. Figure 2 shows the timing of the SPI/Microwire interface. Table 1 defines the parameters listed in the figure.

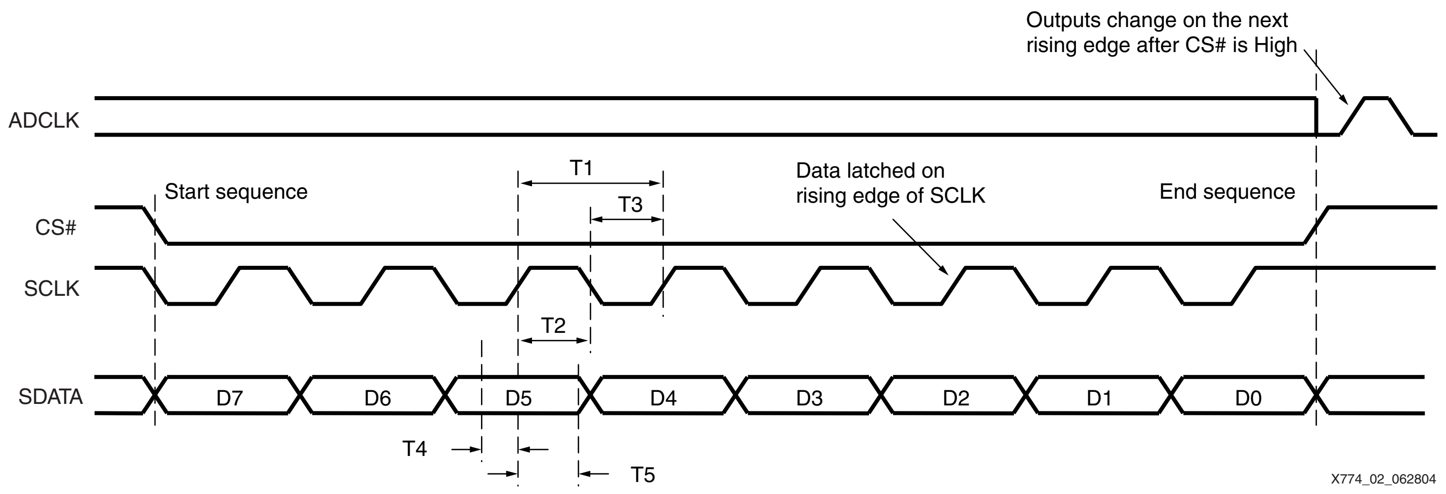
Outputs change on the next rising edge after CS# is High
ADCLK
T1
T3
Start sequence
Data latched on rising edge of SCLK
End sequence
CS#
SCLK
T2
SDATA
D7
D6
D5
D4
D3
D2
D1
D0
T4
T5
X774_02_062804

Figure 2: **SPI/Microwire Interface Timing Waveforms**

Table 1: **Microwire Interface Timing Parameters**

Parameter	Description	Minimum	Units
T1	SCLK Period	50	ns
T2	SCLK High Time	13	ns
T3	SCLK Low Time	13	ns
T4	SDATA Setup Time	5	ns
T5	SDATA Hold Time	5	ns

The SPI/Microwire protocol and its FPGA implementation are not discussed in this application note. Free, adaptable designs can be located on the Internet. For example, search for "SPI core" at http://www.opencores.org to find sample designs.

Single-Channel Interface

For a single-channel design, the ADC specifications are:

- 70 MSPS (equal to 70 MHz)
  - ADCLK = 70 MHz (14.29 ns)
  - LCLK = 420 MHz (2.38 ns)
- 12-bit serial LVDS interface
- LVDS used in dual data rate (DDR) mode

  DDR mode means that the bits are clocked at the FPGA at 840 Mb/s.

The FPGA can receive LVDS data streams at that specified speed without problems when the interface design is carefully constructed. This reference design uses an interface design similar to that in telecom data communication LVDS applications.

The ADS527x transmits edge-aligned data and sync signals with a 90-degree shifted clock. Taking into account the PCB layout, data, sync, and clock arrive at the FPGA pins shifted by 90 degrees.

To register the received data into the FPGA with the receive clock, a digital clock manager (DCM) can be used in fixed phase shift mode.

If doubt exists on the data-to-clock alignment, a DCM can be used in dynamic phase shift mode. DCM phase shifting is discussed in "Auto Phase Shift DCM."

Note: When the "Auto Phase Shift" design approach is used, a fixed shift parameter must be applied to the shift operation state machine to keep the 90-degree phase shift between data and clock.

Figure 3 shows a one-channel receiver module. This module takes in the serial differential data of one channel and outputs it, internal to the FPGA, as 12-bit parallel data. This module is used eight times for an 8-channel ADC device.

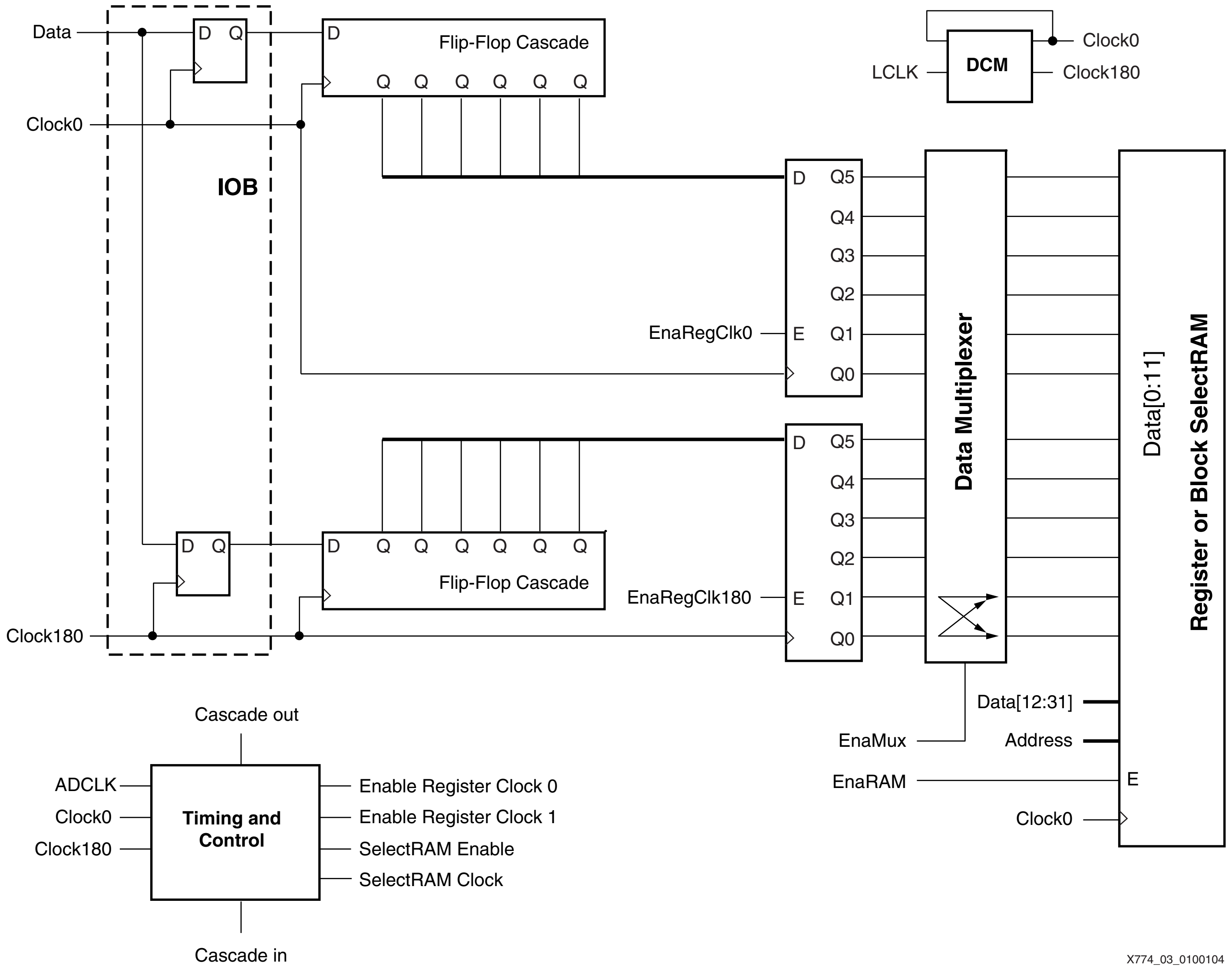

Figure 3: **12-bit Single-Channel Receiver**

For the single-channel design, the received clock is input to a DCM. This DCM aligns the internal clock to the external applied clock or phase shift the internal clock to the received clock. At the FPGA, the presented data is registered at the right moment, in the middle of the valid data range.

Data is presented in DDR format, meaning that it must be clocked into the receiver registers on the rising and falling edges of the receiver clock. Therefore, the clock and 180-degree outputs of the DCM are used. Now the rising edge of both clocks can be used at the flip-flops.

The 12 incoming serial data bits of one channel are split into two sections. The even bits are clocked on the DCM clock, and the odd bits are clocked on the 180-degree shifted clock. At a strobe pulse, ADCLK, these serial registered bits are stored in a parallel register. The result is a 12-bit parallel word with a jumbled data bit order. For ADCLK and strobe pulse generation, refer to "Single-Channel Interface Timing."

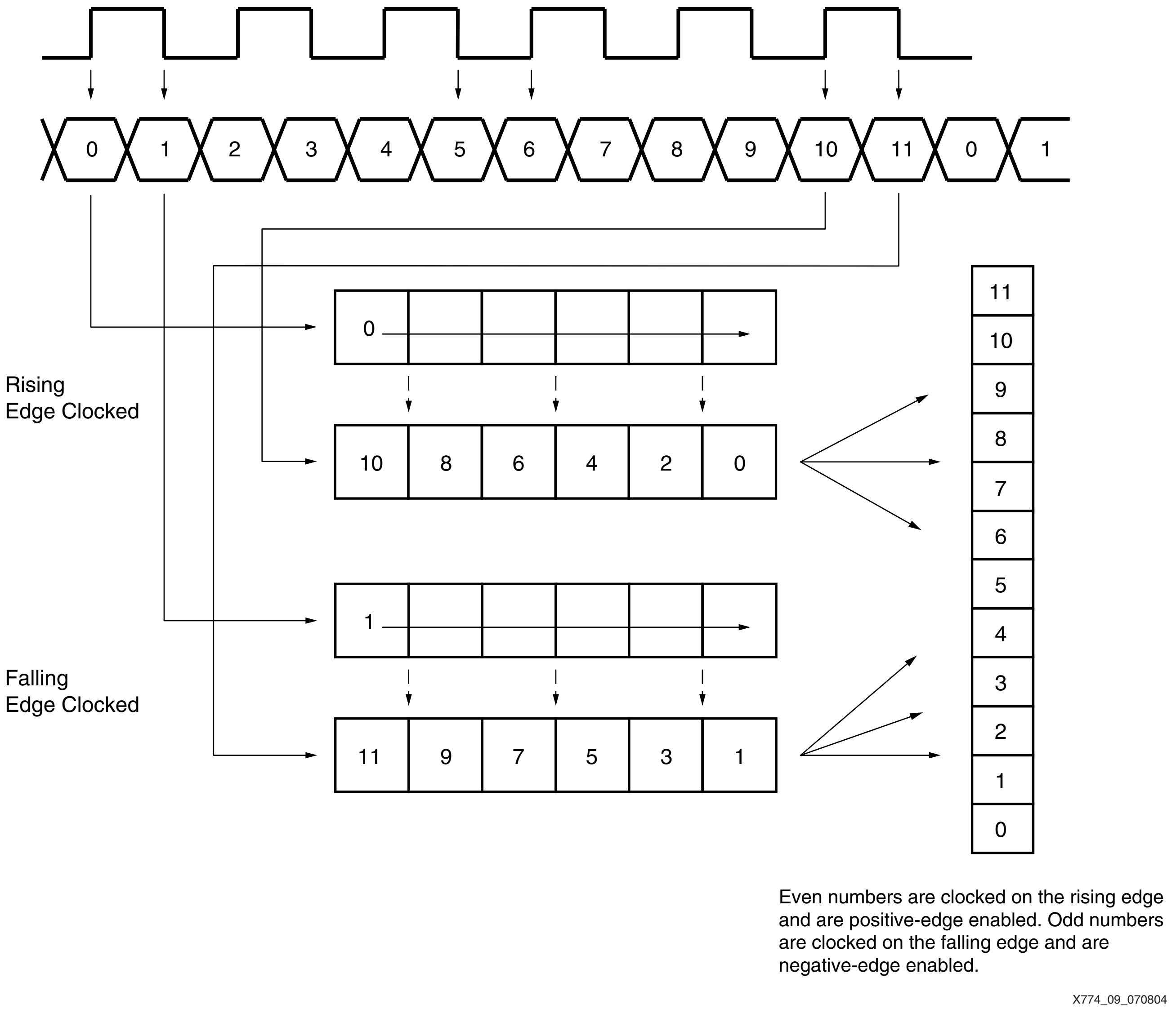
0
1
2
3
4
5
6
7
8
9
10
11
0
1
Rising
Edge Clocked
0
10
8
6
4
2
0
Falling
Edge Clocked
1
11
9
7
5
3
1
11
10
9
8
7
6
5
4
3
2
1
0
Even numbers are clocked on the rising edge and are positive-edge enabled. Odd numbers are clocked on the falling edge and are negative-edge enabled.
X774_09_070804

Figure 4: **Serial Receiver Bit Ordering**

After the bit ordering is resolved in the FPGA routing, this 12-bit word can be used as is by a backend application or can be fed to a set of registers, a Block SelectRAM™ FIFO, or a Block SelectRAM memory used as a register. When the 12-bit word is used as is, the backend application must run at a clock in phase with the received ADCLK clock, which is not simple to do because this clock is used as a sync (frame) input of the LVDS interface. An easier solution is to let the backend FPGA design run at a clock rate similar, but not referenced, to the sampling clock, and use a FIFO to cross the clock domains between the ADC receiver interface and the application logic. Example designs of both options are included in the reference design.

The receiver module, consisting of two data receive channels and one timing block, is built as a macro containing relationally placed macro (RPM) directives. The reference design provides placement of attributes in the design source code, a user constraint file (UCF), and additional information for placement in different I/O banks of the FPGA.

Note: A timing block takes the sync (frame) signal or a previous RPM timing output and generates the necessary signals for enabling the receiver parallel register and the backend design stage.

Figure 5 shows the first instance of a dual-channel block in the FPGA.

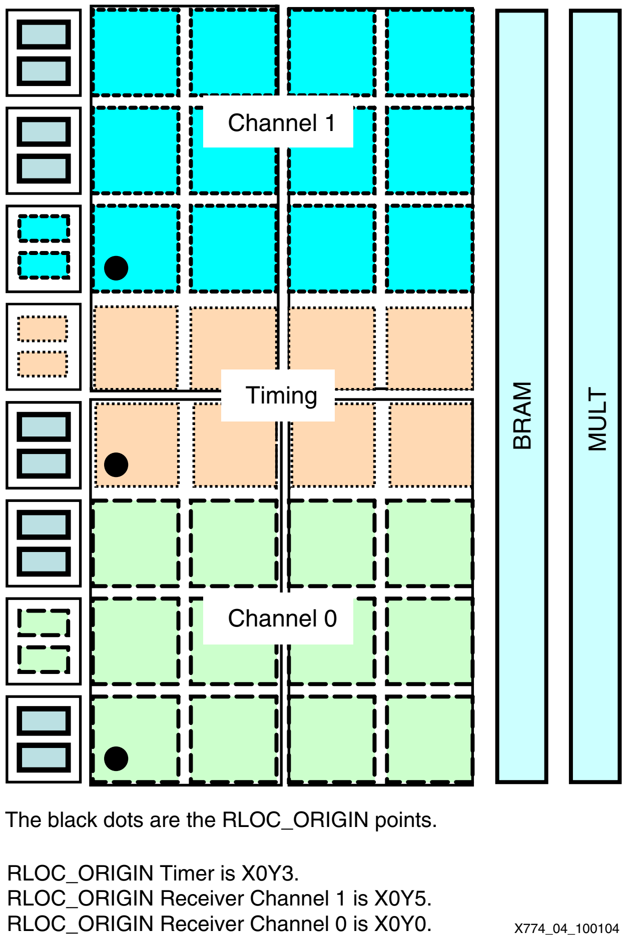
Channel 1
Timing
Channel 0
BRAM
MULT
The black dots are the RLOC_ORIGIN points.
RLOC_ORIGIN Timer is X0Y3.
RLOC_ORIGIN Receiver Channel 1 is X0Y5.
RLOC_ORIGIN Receiver Channel 0 is X0Y0.
X774_04_100104

Figure 5: **RPM Dual Channel and Timing Block**

All clocked elements of the data receiving and timing modules are RLOC'ed in the module. The three modules are then RLOC'ed with respect to each other. These operations are done, with a set of attributes, in the HDL source code. Rearranging the data receiving and timing modules is now easy. Each module can be rearranged while keeping its shape, making it easy to use the design in single-channel applications.

Single-Channel Interface Timing

The high-speed LCLK is used as an interface clock and is therefore fed into a DCM. This DCM generates two phase-aligned clocks, CLK0 and CLK180.

The data-aligned LVDS version of the ADCLK sample clock is used as a strobe signal to align the captured 12-bit words. The strobe signal is sampled on the positive and negative edges of the high-speed LCLK. Edge detection and phase detection are performed. When a rising edge of ADCLK is detected, two signals are generated to enable the parallel storage registers for the positive and negative clocked data.

Normally, even bits (0, 2, ..., 10) are clocked on the rising edge of CLK0, while odd bits (1, 3, …, 11) are clocked on the rising edge of CLK180. However, it is possible for the CLK180 edge to arrive first and clock the even data in the odd data shift register. The same then happens for CLK0. The result is a mangled data word stored in the parallel register. Figure 6 and Figure 7 show the even bits clocked with CLK0 and CLK180, respectively.

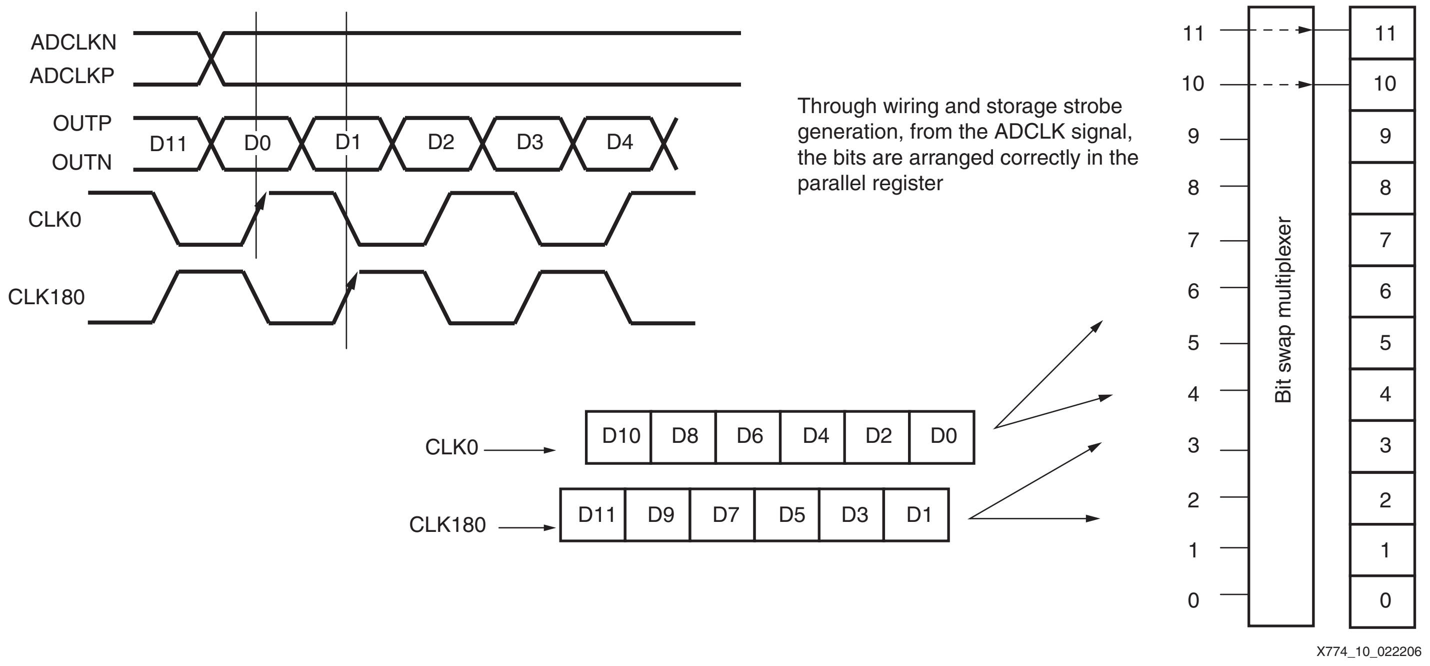

Figure 6: **Even Bits Clocked with CLK0**

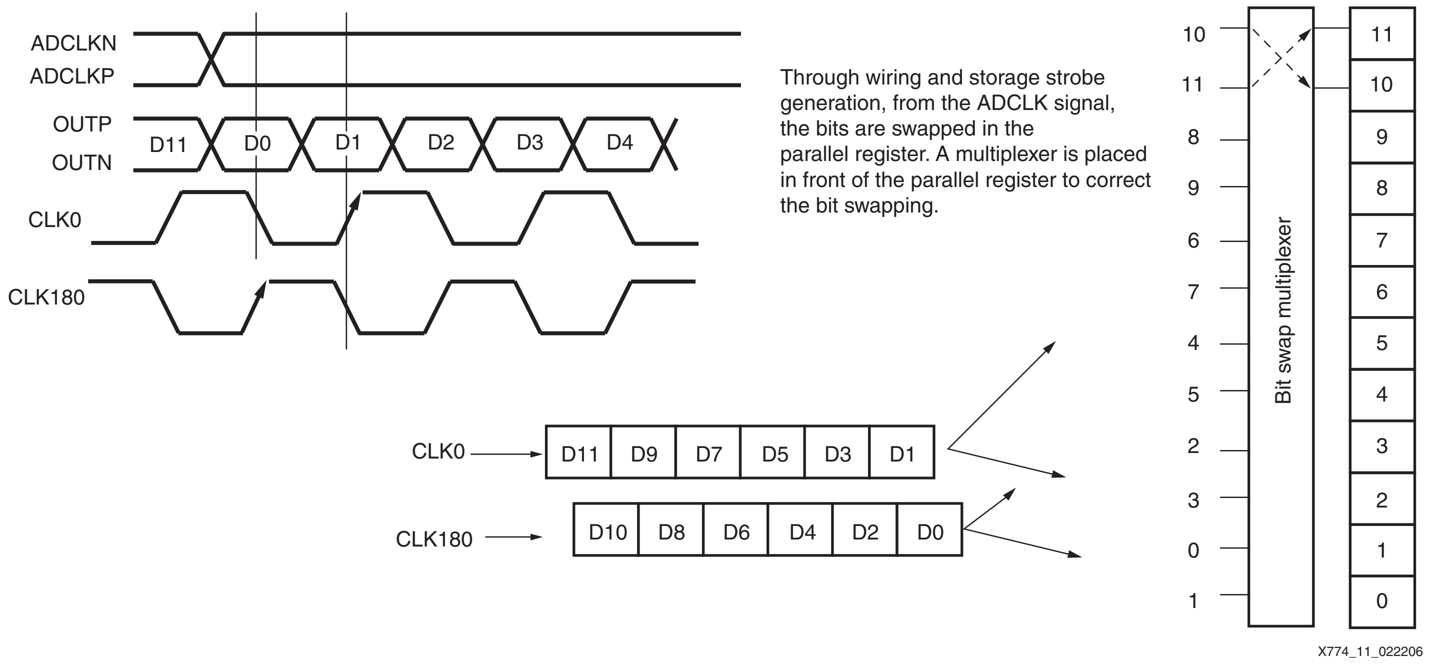
ADCLKN
ADCLKP
OUTP
OUTN
D11
D0
D1
D2
D3
D4
CLK0
CLK180
Through wiring and storage strobe generation, from the ADCLK signal, the bits are swapped in the parallel register. A multiplexer is placed in front of the parallel register to correct the bit swapping.
CLK0
D11 D9 D7 D5 D3 D1
CLK180
D10 D8 D6 D4 D2 D0
10 11 8 9 6 7 4 5 2 3 0 1
Bit swap multiplexer
11 10 9 8 7 6 5 4 3 2 1 0
X774_11_022206

Figure 7: **Even Bits Clocked with CLK180**

To automatically prevent this bit swapping, a small multiplexer is placed between the parallel latch register and the storage register or Block SelectRAM (FIFO). When the negative frame edge of CLK0 comes before CLK180, negative frame edge data is passed as is to the parallel register. When the negative frame edge is first detected with CLK180, data is normally bit-swapped when loaded in the storage register or FIFO. However, data is now arranged correctly through the multiplexer.

Auto Phase Shift DCM

The data, sync, and clock signals are shifted 90 degrees (see Figure 1) when the ADC device transmits them to the FPGA interface. Although data and clock traces in the PCB layout are recommended to have the same length to preserve the original phase shift up to the FPGA, it is possible that this phase relationship is distorted.

The phase shift capability of the DCM can be used to position the received LVDS clock exactly in the middle of the received data bit width. The principle of this design is described in XAPP268 (see Reference Item 2, page 16).

In the Virtex-II, Virtex-II Pro, and Spartan-3 device families, all Input/Output blocks (IOBs) are the same except those used as clock inputs. Each of these IOBs has an extra direct input to the DCM clock input. When these IOBs are used as clock inputs, the normal IOB logic is omitted but can be used. This functionality is used by the reference design.

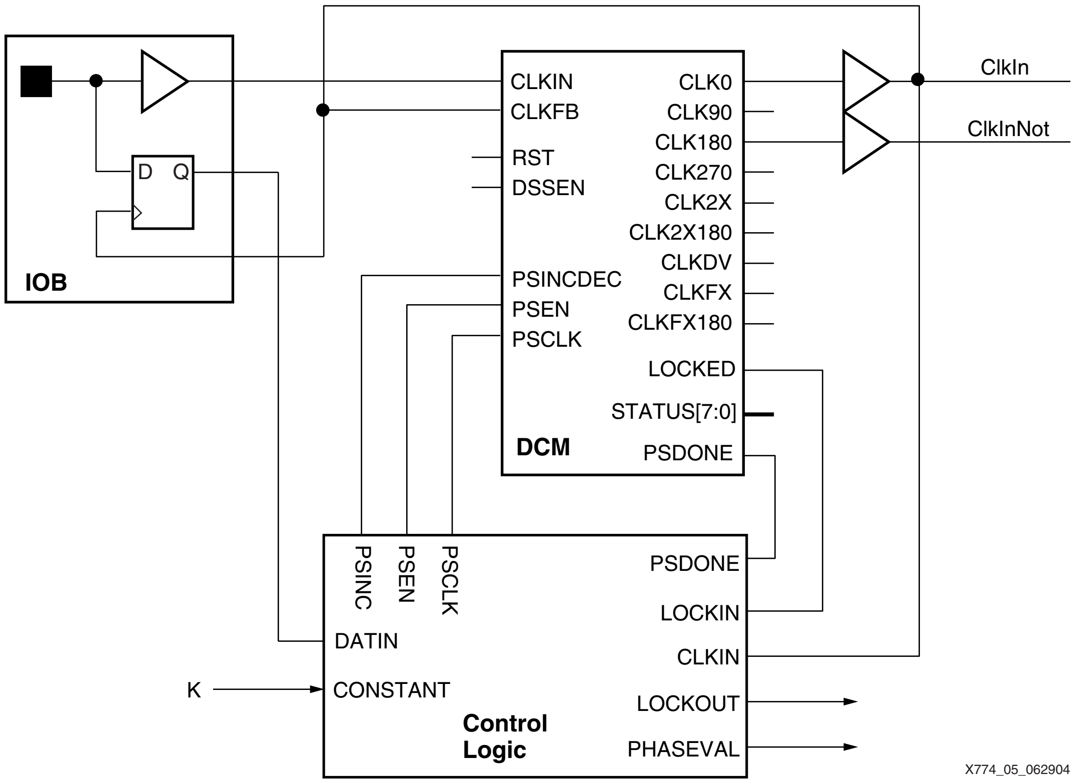

Figure 8: **Auto Phase Shift DCM Circuit**

In the circuit shown in Figure 8, the incoming clock is used as the clock input and the data input for the IOB flip-flop. The input clock registers itself at the IOB flip-flop with the DCM-adjusted clock (CLK0). The DCM is set in variable mode with an initial phase shift of –255. The equivalent counter in the control logic is set to zero as is the internal state machine. When taken out of reset, the DCM adjusts its output clocks and indicates this process with the LOCKED output (see Figure 9).

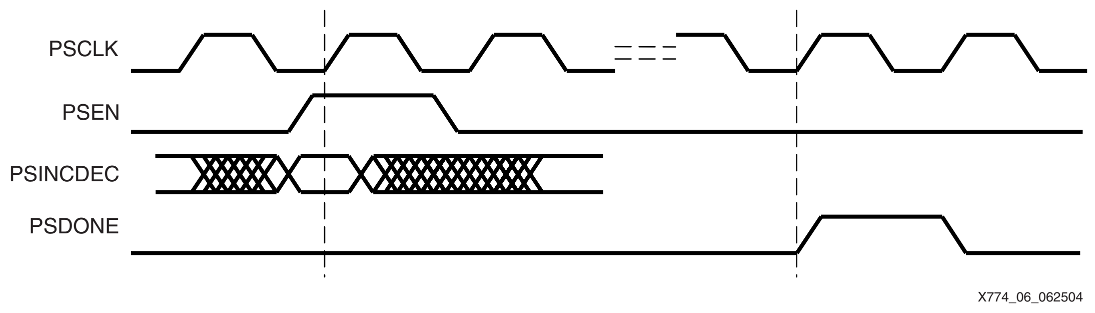

Figure 9: **Phase-Shift DCM Interface Timing**

Next the DCM is incremented by one with the control logic. The control logic counter is incremented by the PSDONE signal from the DCM. When this operation happens, the output of the IOB flip-flop is checked. When the output is "0," the DCM is again incremented. Then the current counter value is stored (ps0), and the state machine jumps to state one.

Incrementing continues until the input flip-flop changes back to "0." The counter value at this point is also stored (ps1). Next the value ps3 (equal to ps1 – ps0) is calculated. The DCM is now decremented until its value is ps3.

Setup is finished. The LOCKOUT output of the control logic is driven High, which performs the same function as the LOCKED output of the DCM, indicating the startup of the DCM is finished.

When using auto-phase shift correction without precautions, the initial 90-degree phase shift might be lost. The DCM moves the clock edge (after adjustment by the control logic) in the middle of a half clock period, reducing the phase shift between clock and data to zero. The fixed value K must be applied to the state machine to be sure a phase shift of 90 degrees is established. The value K is a constant applied in the HDL code.

FIFO and Data Storage Extension

When the basic reference design is used as provided, after the parallelized 12-bit values are clocked in a single register, they must be taken immediately by the backend application design. This design must function at a clock synchronous to ADCLK.

Another approach is to store received parallel data in a FIFO and read it out with the backend application design. In this case, it is possible to run the backend design at an unrelated but frequency-similar clock. The FIFO is used to bridge the two clock domains.

As in telecom applications, ADC converters produce a continuous stream of data, resulting in a continuous stream of parallel 12-bit words. The FIFO that best fits this application is called a self-addressing FIFO.

FIFOs are constructed from memory, read counters (pop), write counters (push), and flagging logic. With a self-addressing FIFO, the counters are replaced by the memory itself, and clock skew is no longer an issue.

Figure 10 shows a standard self-addressing FIFO as described in XAPP291 (see Reference Item 3, page 16). Part of the memory data output is fed back as memory address.

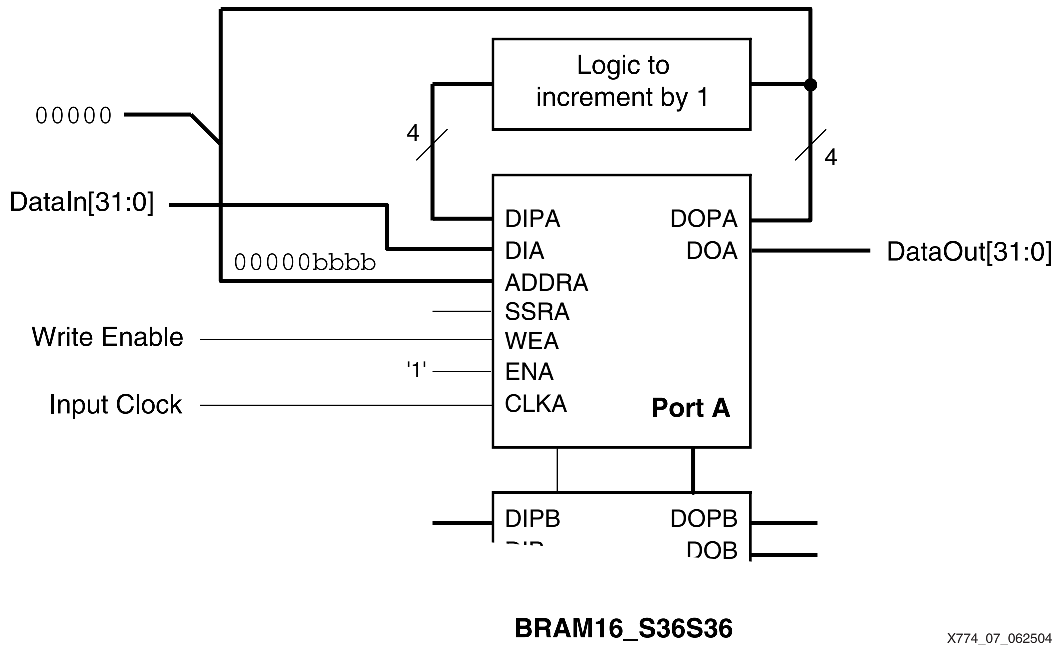

Figure 10: **Standard Self-Addressing FIFO**

At the start of operation, the memory output is assumed to be all zeros. The RAM is addressed with this value from the feedback part of the output. At the same time, the data is incremented (in the LUT adder) by one, and the result is presented at the memory data input together with the incoming data.

The next clock edge addresses memory space “zero” and writes the data value “zero+1 & input data”. Due to the construction of the memory blocks, this value appears at the output after a Clock-to-output (Tbcko) time. Now memory space “one” is addressed, and a data value of two (output one + added one) is presented at the input.

The address increment logic is built from a LUT and does not require a clock. Addition of a clock loses the advantage of direct data out.

Block SelectRAM memories have an ideal design for this mechanism. When a 32-bit wide FIFO is needed, the extra four bits (parity bits) can be used for the address storage-incrementer functionality.

A 16-deep FIFO can be constructed using four bits (the parity RAM bits). If a deeper FIFO is needed, the following solutions are possible:

- Use normal data bits as adder / counter bits

  This solution limits the width of the input data. One block RAM is used per two ADC channels (2*12 bits), using 24 bits and leaving 8 bits as counter / adder storage bits.

- Use an extra adder

  This solution extends the address with the extra adder. When the address reaches the maximum value available in memory, the result is incremented by one. The extra adder does not need to run at full FIFO speed, but the increment to the block RAM timing must be controlled.

Figure 11 shows an address-extended self-addressing FIFO.

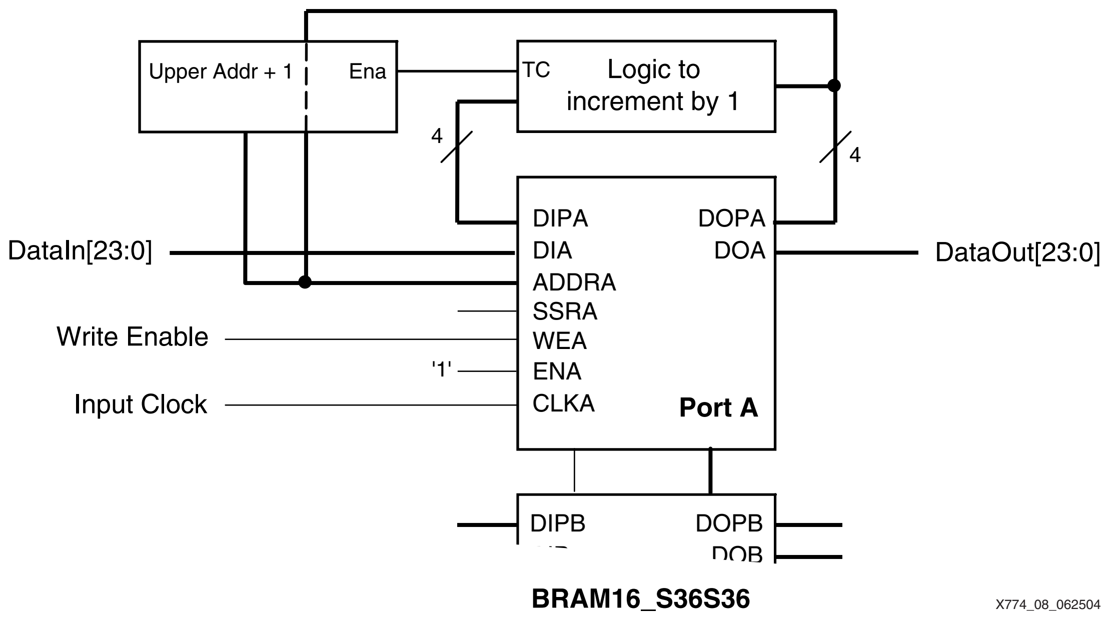

Figure 11: **Address-Extended Self-Addressing FIFO**

PCB Guidelines

This section alerts the PCB designer to several PCB layout issues. Remember that PCB design is not just putting traces on a piece of hard material in order to make electronic components do something meaningful.

Component Placement

Try to place the different circuit components as close as possible to each other and line them up with regard to the I/O pinning. Position the components so that PCB traces do not take a lot of turns, corners, and pass through PCB vias when going from one component to another.

The FPGA is flexible to pin-locking with regard to the internal FPGA design.

A straight, short connection improves all possible parameters of a PCB layout:

- Signal integrity
- Transmission line effects
- Capacitance and inductance
- Operating frequency

When distances between components are long, transmission line effects matter. Make sure that all transmission lines are terminated properly to control reflections.

Before starting a PCB layout, determine the number of layers to be used. Decide also the destination of the layers (signal, ground, power, and so forth).

It is best to take an extra day with the layout software to shuffle the components on your PCB and find a good placement, rather than risk ending up with a bad PCB design.

Component Placement Examples

The data/frame receiver in the FPGA has been RLOC'ed between the IOB and the first Block SelectRAM column in the FPGA. One module contains two data inputs and one framing/timing input (Figure 12).

This makes it possible to lock I/O pins in a range of six possibilities (four I/Os in front of the receiver block, one I/O up, and one I/O down).

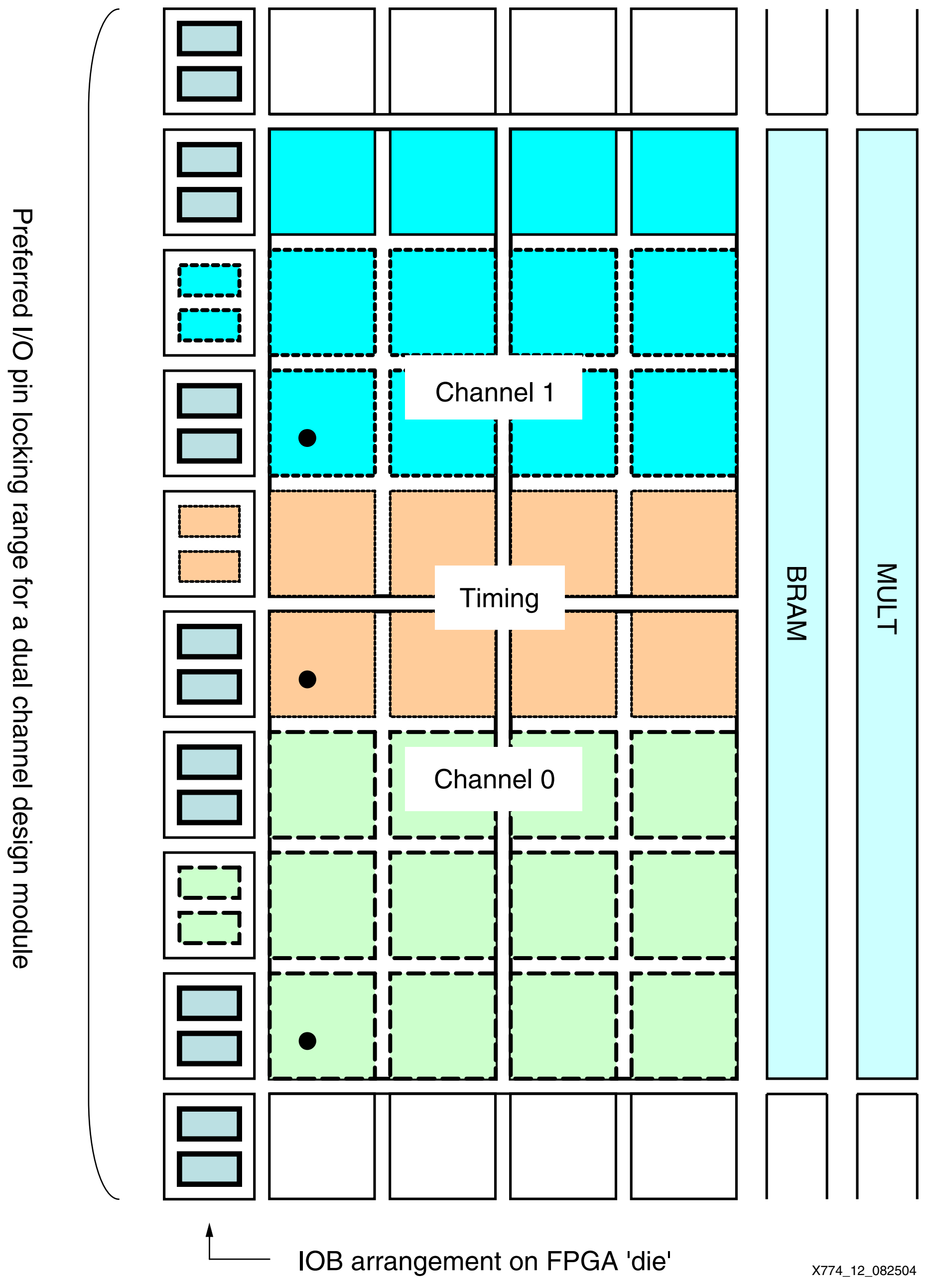
Preferred I/O pin locking range for a dual channel design module
Channel 1
Timing
Channel 0
BRAM
MULT
IOB arrangement on FPGA 'die'
X774_12_082504

Figure 12: **FPGA Placement Example**

Figure 13 shows a possible connection between the ADS527x device and a Virtex-II or Virtex-II Pro FPGA.

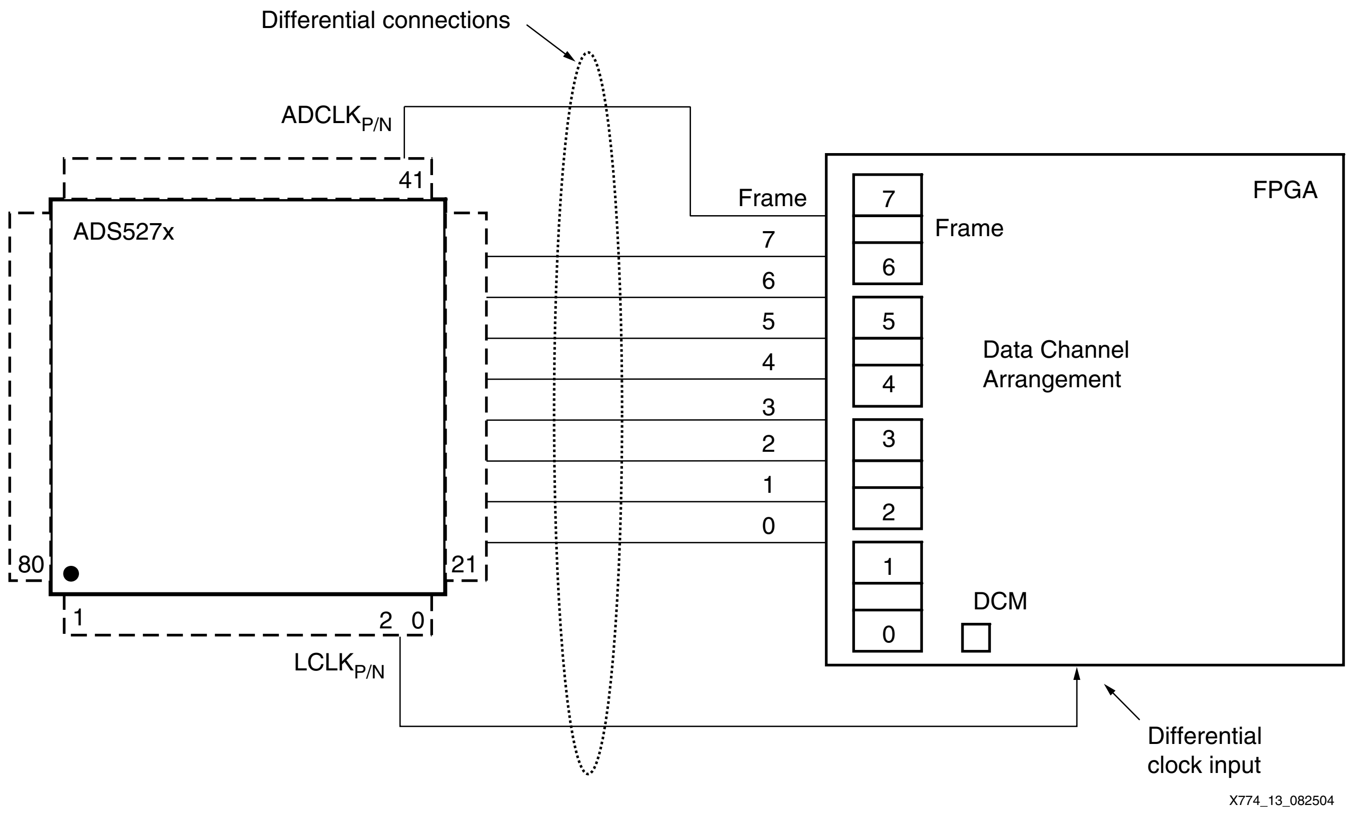

Figure 13: **Connection between an ADS527x and FPGA**

Make sure all Low Voltage Differential Signaling (LVDS) transmission lines have the same length! It is recommended that the clock and data signals have the same length. The DCM in the Virtex-II or Virtex-II Pro FPGA can provide help positioning the clock to the data when used in Phase Shift mode.

Guidelines and Recommendations

The following bulleted list summarizes key guidelines for PCB designs:

- Spend sufficient time when placing components for layout. With today’s layout tools, this is like a shuffling block game.
- Keep the trace lengths as short as possible.
- Spend time determining the number of PCB layers and how the layer stack-up is realized.
- If possible during PCB layout, keep the length of a track shorter than the travel and reflection time of the signal on the trace. If not possible, take the transmission line theory into account.
- Match the length of all differential traces (data and clock).
- When making turns with these differential traces, turn as many times to the left as to the right. When making a turn with a differential trace, the inner trace becomes shorter than the outer trace of the pair. When using more turns in one direction, one trace of the differential pair is longer than the other (without direct correction possibilities).
- Route traces on the PCB as far from each other as possible to avoid or minimize crosstalk effects. Spread traces after routing over the available space of the PCB.
- Do not route traces into 90-degree turns, and avoid routing traces into 180-degree turns (except when you know what you are doing). 90-degree turns increase the effective width of the trace, contributing to parasitic capacitance. At very fast edge rates, these discontinuities can cause significant signal integrity problems.

- Use round, circular turns. If this is not possible, use 45-degree corners.
- Take the guidelines of the signal return paths into account.
- Guard traces.
- Remember the importance of ground planes.
- Specific to the ADS52xx TQFP package: Use the thermal pad connection, called PowerPad, to improve the operating stability of the device.

Reference Design

The reference design uses techniques described in XAPP291, XAPP265, and XAPP268 (see "Reference Material"). The design is set up as a modular block design.

Design Files

A complete functional design is included in the ZIP file with this application note. The reference design files can be downloaded from:

http://www.xilinx.com/bvdocs/appnotes/xapp774.zip

The files refer to the Xilinx demo boards using XC2VP20-6FF896 devices or Texas Instruments demo boards using XC2V250-6FG256 devices.

Note: The BIT file for download of the design is for JTAG use only! The `xapp774.zip` file contains `readme.txt` files in order to make customization of this reference design easy.

Additional design files are provided so that the design can easily be used in other applications and systems. These files include:

- Small and large receive FIFOs
- Auto phase shift DCM
- A UCF file with the parallel data inputs and outputs placed on the same side of the FPGA (in the same I/O bank)
- Simulation files for the reference design and subdesigns

Design Directory Setup

This section provides the directory setup of the reference design files:

```
/AdcTi
  Ads5273IntV2Simple_readme.txt   -- This readme file.
  /Documents
    ads5273.pdf                   -- Texas Instruments ADC datasheet.
    ADS5273 Simple Interface.ppt  -- PowerPoint presentation about this design.
    XAPP774.pdf                   -- Xilinx Application note. Look on the Xilinx
                                  -- web pages for the latest version.
  /Extra
    Now empty but can contain PCB layout, schematics, etc.
  /Ise
    /FreqGenXlvdsPro
    /Virtex2V2506Fg256            -- Directory for Xilinx tools, simple design.
    /Virtex2VP206FF896            -- Directory for Xilinx tools, full design.
  /SimScripts
    SimReadMe.txt
    RandomNumberGenerator.xls
    /ADS527xEVM_ADSDeSer-50EVM
    /XlvdsPro                     -- Various simulation scripts for Models use.
                                  -- All files are .do files and can be invoked
                                  -- from within Models.
      Tools --> Execute Macro --> browse to the correct .do file.
  /Simulation
    /work                         -- Work directory for the simulator.
  /Synthesis
    SynthReadMe.txt
    /FreqGenXlvdsPro
    /Synplicity
      /Virtex2V2506FG256          -- Synthesis directory of TI eval. design.
      /Virtex2VP206FF896          -- Synthesis directory for XlvdsPro design.
    /Precision
      /Virtex2V2506FG256          -- Synthesis directory of TI eval. design.
      /Virtex2VP206FF896          -- Synthesis directory for XlvdsPro design.
  /Ucf
    UcfReadMe.txt
    Different User Constraint Files for use with ISE tools.
  /Verilog
    Verilog source code
  /Vhdl
    VhdlReadMe.txt
    /ADS527xEVM_ADSDeSer-50EVM
      /AdcReceiver
        AdcReceiver.vhd
        AdcReceiver_TestBench.vhd
        Ads5273_tester.vhd
        Receiver.vhd
        RxTiming.vhd
      /AdsV2SmplNoFifo              -- design for Virtex-II FPGA, no FIFO.
        Ads5273_Tester.vhd
        GenStuff_Tester.vhd
        Toplevel_V2_Smpl_Nff.vhd
        Toplevel_V2_Smpl_Nff_TestBench.vhd
      /DcmPhaseCtrl
        Bcd35FullRange.vhd
        PhaseControlFullRange.vhd
    /XlvdsPro
      /AdcReceiver
        AdcReceiver.vhd
        AdcReceiver_TestBench.vhd
        Ads5273_tester.vhd
        Receiver.vhd
        RxTiming.vhd
      /AdsV2SmplNoFifo              -- design for Virtex-II FPGA, no FIFO.
        Ads5273_Tester.vhd
        GenStuff_Tester.vhd
```

```
          Toplevel_V2_Smpl_Nff.vhd
          Toplevel_V2_Smpl_Nff_TestBench.vhd
        /DcmPhaseCtrl
          Bcd35FullRange.vhd
          PhaseControlFullRange.vhd
        /DcmLifeIndicator
          DcmLifeIndicator.vhd
      /ExtraSourceCode
        /LargeSelfAddrFifo
          GrayCnt4b.vhd
          LargeSelfAddrFifo.vhd
          LargeSelfAddrFifo_TestBench.vhd
          LargeSelfAddrFifo_Tester.vhd
        /SmallSelfAddrFifo
          SmallSelfAddrFifo.vhd
          SmallSelfAddrFifo_TestBench.vhd
          SmallSelfAddrFifo_Tester.vhd
```

Reference Material

The following documents provide supplementary material useful with this application note:

1. Xilinx XAPP265: "High-Speed Data Serialization and Deserialization (840 Mb/s LVDS)"
2. Xilinx XAPP268: "Active Phase Alignment"
3. Xilinx XAPP291: "Self-Addressing FIFO"
4. Xilinx XAPP623: "Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors"
5. Xilinx XAPP659: "Using 3.3V I/O Guidelines in a Virtex-II Pro Design"
6. *RF Circuit Design: Theory and Applications*: Reinhold Ludwig/Pavel Bretchko (Prentice Hall ISBN 0-13-095323-7).
7. *RFI/EMI/EMC: A Designer's Handbook*: Gary A. Breed
8. *Microwave Circuit Analysis and Amplifier Design*: Samuel Y. Liao (Prentice Hall ISBN 0-13-586736-3)
9. *Reference Data for Engineers*: *Radio, Electronics, Computer, & Communications*, Mac E. Van Valkenburg (Newnes)
10. *Transmission Lines for Digital and Communications Networks*: Richard E. Matick (McGraw Hill, 1969)
11. IPC publications: IPC-2141 and IPC-D-317A, along with corrections and adjustments to these publications from IPC and others
12. *Radio Handbook*, 23rd edition: William I. Orr (Newnes, 1997. ISBN 0-7506-9947-7)
13. *High-Speed Digital Design: A Handbook of Black Magic*, Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-395724-1)
14. *High-Speed Signal Propagation: Advanced Black Magic*, Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-084408-x)

Conclusion

A Xilinx Virtex-II, Virtex-II Pro, or Spartan-3 (lower speed) FPGA easily can be used as an interface to high-speed Texas Instruments ADS527x analog-to-digital converters. The FPGA can be used as a direct data processing unit in high-speed data conversion designs and perform a front end or backend application for a DSP, such as the TMS320C64xxx devices. A possible design setup can be: ADC — LVDS FPGA interface — FPGA processing unit — FPGA EMIF interface — DSP.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
07/26/04	1.0	Initial Xilinx release.
11/04/04	1.1	Replaced Figure 3. Added paragraph about RLOC'ing under Figure 5. Rewrote "Single-Channel Interface Timing" and included two new figures (Figure 6 and Figure 7). Added "PCB Guidelines" section. Revised the directory structure in "Design Directory Setup". Added additional documents to "Reference Material".
02/23/06	1.2	Updated Figure 6 and Figure 7.