XAPP268 (v1.2) December 9, 2002

Active Phase Alignment

Author: Nick Sawyer

Summary

The Digital Clock Manager (DCM) in the Virtex™-II series of FPGAs is an extremely powerful logic element. It allows fine phase adjustment of an incoming clock in increments of around 50 ps. This is typically necessary when clocking in an incoming data stream either single or double data rate at very high frequencies – up to 670 MHz SDR (420 MHz DDR) in a Virtex-II FPGA (-5 speed grade). Normally the DCM is set up to provide a constant phase shift that allows the incoming data to be correctly clocked in. This phase shift is corrected for both temperature and voltage, but can vary slightly across different devices and wafer lots, thus effectively reducing slightly the receiver window or "eye". One way of correcting for this is to set up the DCM phase shift dynamically via training either at device reset, or on a continuous basis. Systems using both single data rate (SDR) and double data rate (DDR) reception can use the concepts in this application note.

Circuit Description

Training is not a new concept, it is used widely in telecom systems. However, to train the DCM correctly, a known training sequence has to be present on the incoming data pins. This is traditionally all ones followed by all zeros. In many systems, there is inadequate bandwidth to allow for this training sequence to be transmitted. The design in this application note uses the incoming clock itself to provide the training information. A block diagram of a Virtex-II Input Output Block (IOB) input path showing the DDR registers is shown in Figure 1. For a clock input pin, the two input registers are normally ignored and the clock signal is passed directly to the DCM. However, it is also possible to feed the incoming clock signal to the D inputs of the two registers, as well as passing the signal through to the DCM. Only the signal Q_0 is used in this application.

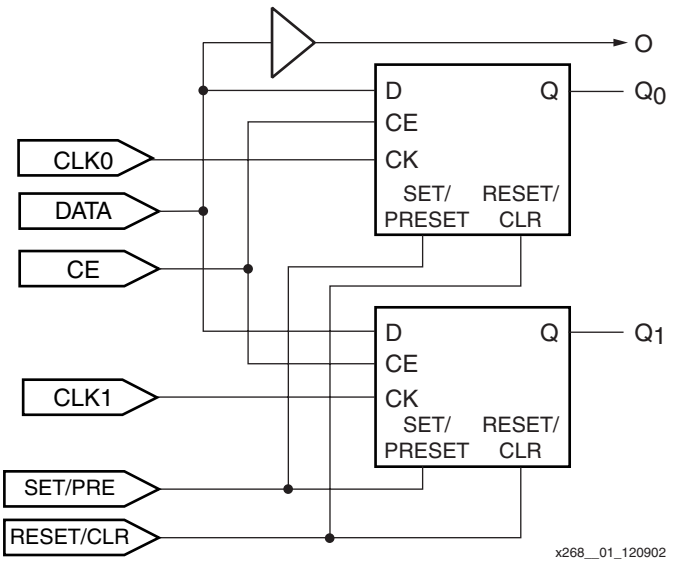

Figure 1: **Virtex-II IOB Input Path Showing the DDR Registers**

Using the corrected clock from the DCM to clock the upper register in Figure 1, the state of the input pin at corrected clock = 0° can be examined. The phase of this corrected clock can be varied by either ±360° (full-range operation) or by 0 to +360° (half-range operation) from the original incoming clock.

The operation of the full-range phase aligner now becomes fairly obvious, as shown in Figure 2, Figure 3, and Figure 4. By starting with a phase shift of –255 (–360°), output Q_0 (positive edge clocked) will be, for example, zero. As the phase shift is increased positively (clock retarding), this will change at some point to one. The tipping point effectively indicates the setup time to the rising edge of the input clock, via the flip-flop, with respect to the corrected clock. Assume this phase shift value is ps0. The phase continues to be increased positively, and at some point the data from the register will change back to zero. This point effectively corresponds to the setup time of the falling edge of the input clock, via the register, with respect to the corrected clock. This phase shift is called ps1. Operation for the half-range phase aligner is identical except the starting point is a DCM phase shift value of 0 (= 0°)

The DCM has now effectively measured the data window or "eye" of the input registers for this particular device at the frequency of board operation. The value of ps1 – ps0, should be a value of 128 (or a 180° phase shift). However, an incoming clock with an exact 50:50 duty cycle is extremely unlikely. The value in practice varies. Additionally, the values ps0 and ps1 directly indicate the duty cycle of the incoming clock, forming a very useful test mechanism. The duty cycle percentage is found by calculating:

$$ps2 = \text{Duty Cycle Percentage} = \frac{ps1 - ps0}{2.56}$$

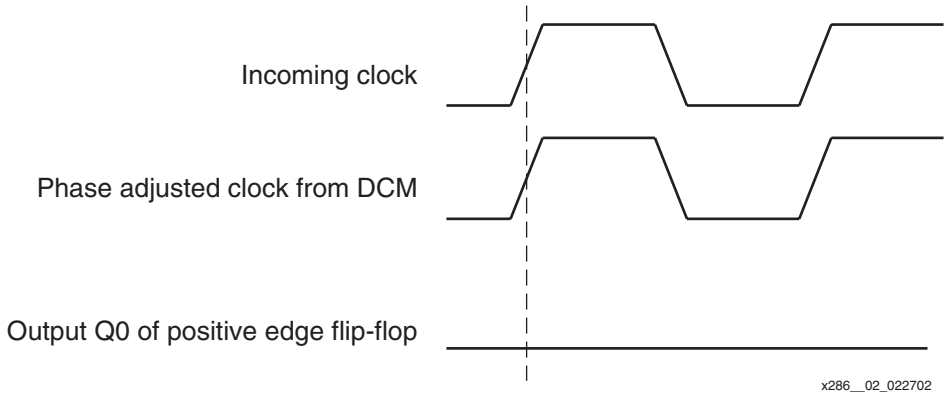

Figure 2: **Phase Value = –255**

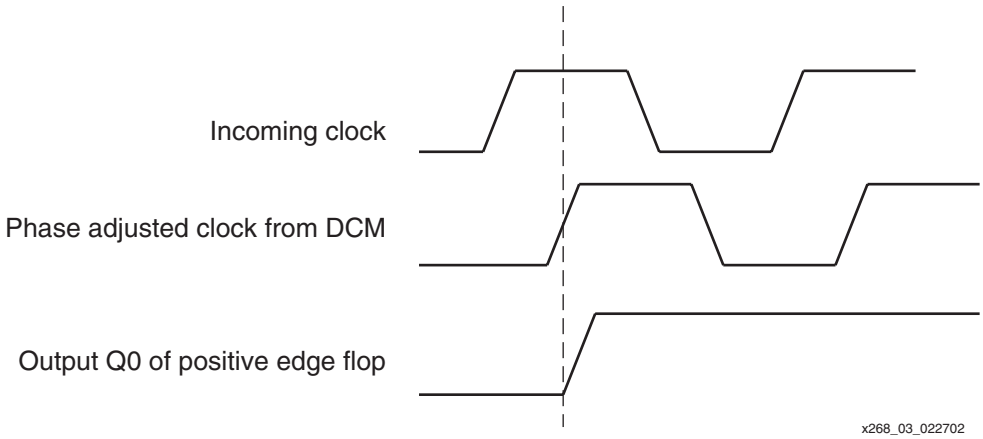

Figure 3: **Phase Value = ps0 (Setup for Rising Edge Found)**

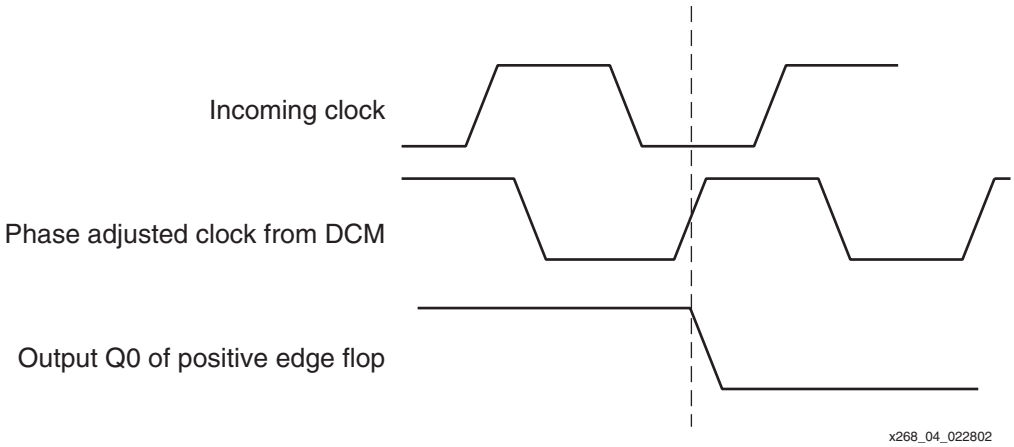

Figure 4: **Phase Value = ps1 (Setup for Falling Edge Found)**

By calculating ps1 – ps0, and using this phase shift value for the DCM, the sampling clock is placed exactly in the middle of the incoming data stream "eye". An assumption is made about the routing on the PCB for the received clock and the received data to arrive with very low skew. If this is not the case, then a constant is added to the equation. This is also the case where there is a known difference in output time for the clock and data in the transmitting device. The value of this constant is easily determined, and needs to be entered in the VHDL or Verilog file as necessary. If the clock and data arrive at the same time, then the value (K) will be zero. If, for instance, the clock arrives 90° phase shifted, then the value entered in the code should be 64. Once the DCM is set to the computed value (ps3) the circuit is ready for operation, i.e., reception of data, but the for obvious reasons phase value of the DCM should no longer be varied.

The clock actually controlling the logic needs to run at a slower speed than the received clock since the state machine has quite a few levels of logic. A good way to generate this clock is from the CLKDV output of the DCM. The input clock divided by four is a good choice. This is set with attributes applied to the DCM in the code. If there is a spare global buffer available for this clock, good design practices would use it. However, the logic can be clocked without a global buffer by using "secondary backbone" routing. If necessary, the place and route software will automatically use this feature.

If it is required to continuously monitor and or modify the input phase relationship (or indeed duty cycle), then a second DCM and control circuit will be required. This second DCM and associated control circuit will be measuring the optimum alignment value, while the first is still clocking in data using the original value calculated. If, for some reason, the values from DCM1 and DCM2 drift apart, then the value for ps3 from DCM2 can be used to modify DCM1 while it is still in operation without data loss.

The previous description assumes a DDR clock. If the incoming clock is SDR, then it needs to be divided by two at the input to the DCM. This is achieved by setting the attribute CLKIN_DIVIDE_BY_2 in the code. The detection mechanism is made slightly different. It will search for two rising edges rather than the DDR case of a rising edge followed by a falling edge.

Logic Description

The block diagram for the module is shown in Figure 5, and the timing requirements for the signals to control the DCM are shown in Figure 6. The pin PSCLK requires a free running clock, PSEN is asserted for one cycle of this clock only, with PSINCDEC either high or low, depending on the direction of shift required. Some time later, PSDONE is asserted for one cycle to indicate that the shift has occurred. If required, the DCM is then ready for the next phase change.

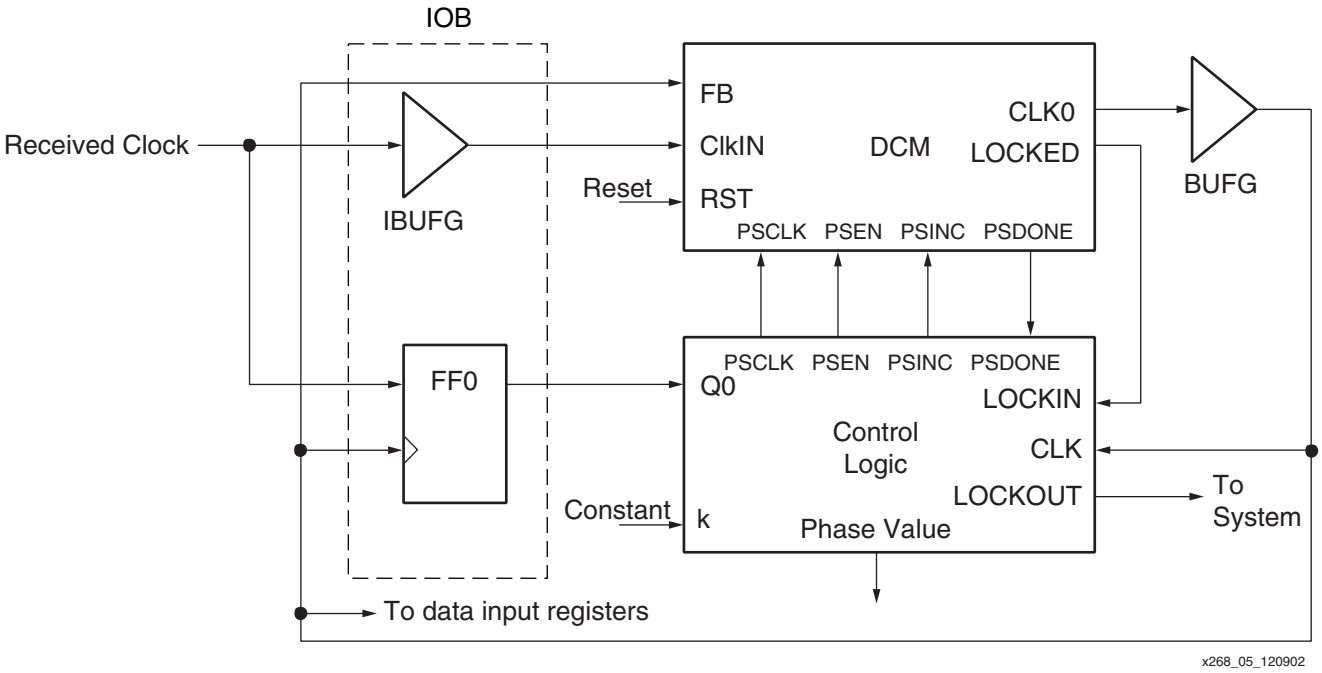

Figure 5: **Logic Block Diagram**

The incoming receiver clock is fed through an IBUFG for the appropriate input standard to a DCM. The DCM is programmed to be in variable mode with an initial phase shift of –255 for the full-range operation, or 0 for the half-range operation. The equivalent counter in the control logic is set to zero, as is the internal state machine. The circuit waits for the DCM to indicate locked. At this point the DCM is incremented by one, as shown in Figure 6. The control logic counter is incremented by the PSDONE signal returning from the DCM. This in turn enables a check to see if the input flip-flop is at "0" or had changed to "1". If it is still "0" the DCM is again incremented, and if it has changed, the current count value is stored (ps0) and the state machine takes state one. Incrementing continues until the input flip-flop changes back to "0". The count value at this point is also stored (ps1), the state becomes two, and the value ps3 (= ps1 – ps0) is calculated. The DCM is now decremented until it is at the value of ps3. Setup is then finished, and the LOCKOUT pin is driven High to indicate startup is finished to the rest of the logic. A 13-bit BCD coded output is available at all times to indicate the current value of the DCM phase shift. Bit 12 of this vector acts as a sign bit, allowing values from –255 to +255. Also included in the macro are inputs to allow the count to be forced up or down manually after the dynamic mechanism has finished its adjustment.

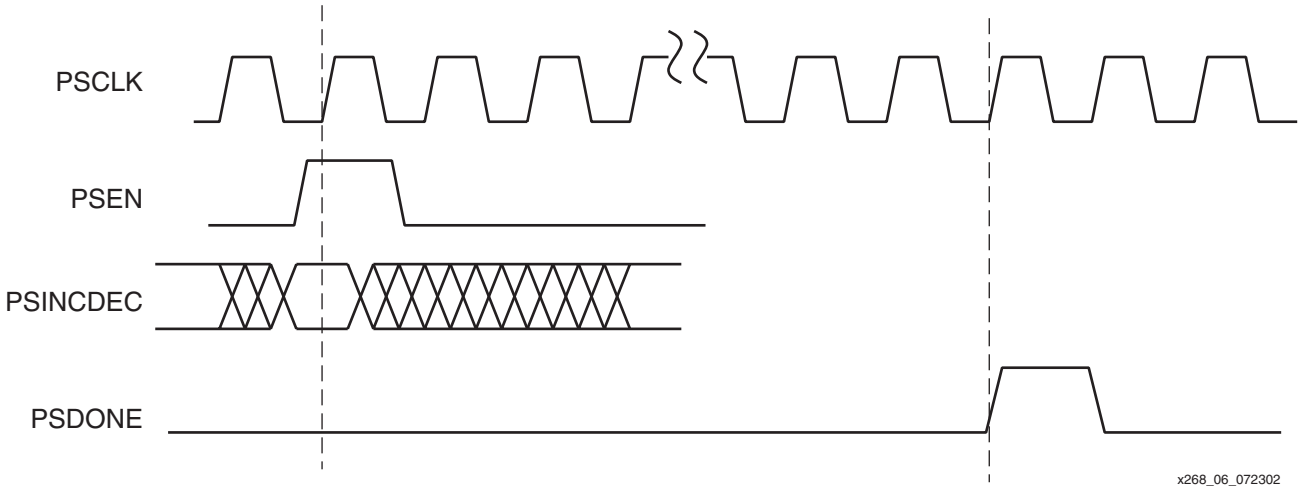

Figure 6: **DCM Control Signal Timing**

Design Files

The reference design files for this application note are written in both Verilog and VHDL. The Xilinx FTP site URL is **ftp://ftp.xilinx.com/pub/applications/xapp/xapp.268.zip**.

Conclusion

Dynamic phase shifting can be used to "tune" individual components to the operating environment. The adjustment are made once at system power-up, or on a continuous basis.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
03/08/02	1.0	Initial Xilinx release.
07/24/02	1.1	Revised Figure 6.
12/09/02	1.2	Added half range (0 –255) operation.