XPS Delta-Sigma Digital to Analog Converter (DAC) (v1.01a)

DS588 December 2, 2009 **Product Specification**

Introduction

The digital to analog converter (DAC) converts a binary number into a voltage directly proportional to the value of the binary number. A variety of applications use DAC including waveform generators and programmable voltage sources. The only external circuitry required is a low pass filter comprised of one resistor and one capacitor.

Features

- PLB interface is based on the PLB v4.6 specification
- Selectable DAC width
- Selectable full scale output
- Programmable interrupt generation
- 16 entry deep data FIFO

LogiCORE™ IP Facts	
Core Specifics	
Supported Device Family	Spartan®-3A/3A DSP, Spartan-3, Spartan-3E, Automotive Spartan 3/3E/3A/3A DSP, Spartan-6, Virtex®-4 /4Q/4QV, Virtex-5/5FX, Virtex-6/6CX
Resources Used	
See Table 10, Table 11, Table 12, Table 13, and Table 14.	
Provided with Core	
Documentation	Product Specification
Design File Formats	VHDL
Constraints File	EDK TCL Generated
Verification	N/A
Instantiation Template	EDK
Design Tool Requirements	
Xilinx Implementation Tools	ISE® 11.4 or later
Verification	ModelSim PE/SE 6.4b or later
Simulation	ModelSim PE/SE 6.4b or later
Synthesis	XST
Support	
Provided by Xilinx, Inc.	

Functional Description

A Delta-Sigma DAC uses digital techniques to convert a digital number into an analog voltage. Consequently, it is impervious to temperature change, and may be implemented in programmable logic. Delta-Sigma DACs are actually high-speed single-bit DACs. Using digital feedback, a string of pulses is generated. The average duty cycle of the pulse string is proportional to the value of the binary input. The analog signal is created by passing the pulse string through an analog low-pass filter. While an in-depth discussion of Delta-Sigma conversion is beyond the scope of this document, the basic architecture, implementation, and trade-offs are covered.

Delta-Sigma Architecture

Figure 1 is a top-level block diagram of a typical Delta-Sigma DAC implementation. As shown in this diagram, the inputs include reset and clock signals, in addition to the binary number bus.

DACoutDrvr drives an external low-pass filter. V_{OUT} can be set from 0 V to V_{CCO}, where V_{CCO} is the supply voltage applied to the FPGA I/O bank driving the resistor-capacitor filter.

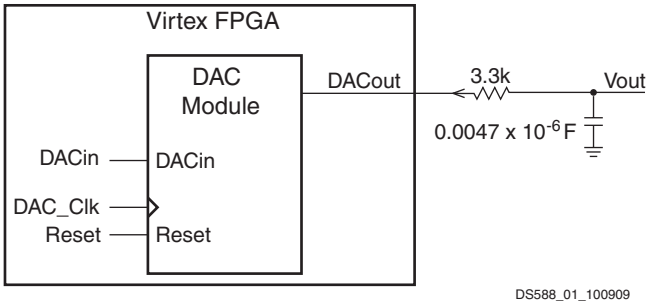

Figure 1: **DAC Module**

The classic current summing digital to analog converter uses matched resistors to convert a binary number to a corresponding voltage level. This technique works well for high-speed DACs when the binary number is up to ten bits wide. However, it is difficult to maintain accuracy over a range of temperatures as the number of bits increases.

Delta-Sigma DACs are used extensively in audio applications. They are suited for low frequency applications that require relatively high accuracy.

As is standard practice, the DAC binary input in this implementation is an unsigned number with zero representing the lowest voltage level. The analog voltage output is also positive only. A zero on the input produces zero volts at the output. All ones on the input cause the output to nearly reach V_{CCO}. For AC signals, the positive bias on the analog signal can be removed with capacitive coupling to the load. Figure 2 is the detailed block diagram of the XPS Delta-Sigma DAC. The width of the binary input in the implementation described below is configurable. For simplicity, the block diagram depicts a DAC with an 8-bit binary input.

The term, *Delta-Sigma*, refers to the arithmetic difference and sum, respectively. In this implementation, binary adders are used to create both the difference and the sum. Although the inputs to the Delta adder are unsigned, the outputs of both adders are considered signed numbers.

The Delta Adder calculates the difference between the DAC input and the current DAC output, represented as a binary number. Since the DAC output is a single bit, it is “all or nothing”; i.e., either all

zeroes or all ones. As shown in Figure 2, the difference will result when adding the input to a value created by concatenating two copies of the most significant bit of the Sigma Latch with all zeros.

This also compensates for the fact that DACin is unsigned. The Sigma Adder sums its previous output, held in Sigma Latch, with the current output of the Delta Adder. In most cases, the Delta adder is optimized out when the high level design is synthesized.

This is because all bits on either the A or B inputs are zero, so A and B are simply merged, rather than added. As noted below, the DAC input can be widened by one bit to allow the full analog range of 0V to VCCO. In this case, the Delta adder is needed.

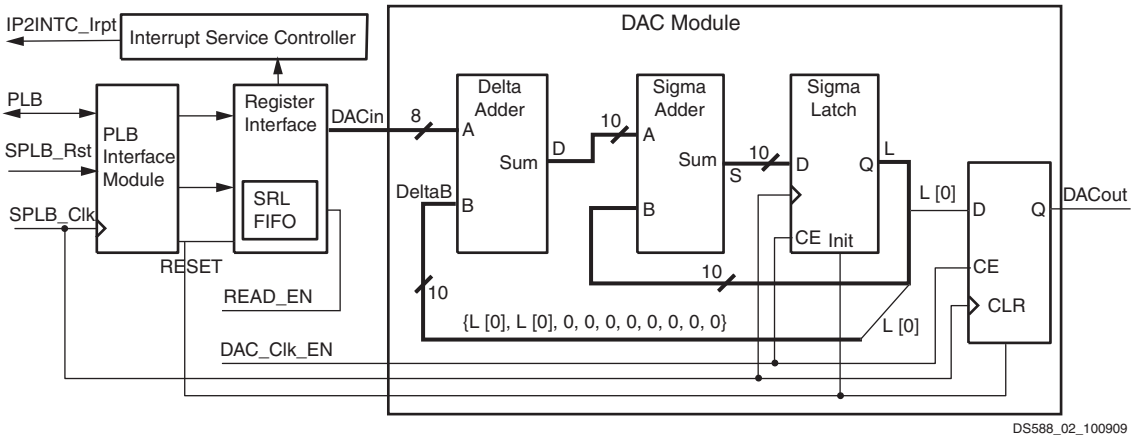

Figure 2: **XPS Delta-Sigma DAC Internal Block Diagram**

For the implementation in Figure 1, the output voltage (VOUT) as a function of the DAC input may be expressed as follows:

$$V_{OUT} = (DACin/(2^{(C\_NUM\_DAC\_BITS)})) \times V_{CCO} \text{ Volts}$$

For example, for an 8-bit DAC (C_NUM_DAC_BITS = 8) the lowest V_{OUT} is 0 V when DACin is 0x00. The highest V_{OUT} is 255/256*V_{CCO} volts when DACin is 0xFF.

For some applications, it may be important for V_{OUT} to swing through the entire voltage range: 0 V to V_{CCO} (rail-to-rail). This is accomplished by using the C_FULL_RANGE generic which increases the DACin bus width by one bit and leaving all other bus widths the same. For an 8-bit DAC with an input value of 256, $V_{OUT} = V_{CCO}$. Note for C_NUM_DAC_BITS = 8 that all DAC in values greater than 256 are illegal and should not be used. The valid range of digital inputs for given values of C_NUM_DAC_BITS and C_FULL_RANGE is given as 0x0 to $0x(2^{(C\_NUM\_DAC\_BITS)} - 1 + C\_FULL\_RANGE)$, where $2^{(C\_NUM\_DAC\_BITS)}$ is always greater than or equal to $(2^{(C\_NUM\_DAC\_BITS)} - 1 + C\_FULL\_RANGE)$.

As outlined in this specification, it is often advantageous to use a high-frequency clock. The desired clock may be faster than that which can be practically sourced externally.

Low-Pass Filter

The resistor/capacitor low-pass filter shown in Figure 1 is adequate for most applications. A 24 mA LVTTL output buffer is used to provide maximum current drive.

There are three primary considerations in choosing values for the resistor and capacitor:

- Output Source and Sink Current: Unlike normal digital applications, it is important that signal DACout always switch the entire voltage range from 0 V to V_{CCO} (rail-to-rail). If the value of R is too low and signal DACout can not switch rail-to-rail, the analog output is non-linear; i.e., the absolute output voltage change resulting from incrementing or decrementing DACin is not constant. The worst-case output impedance of the 24 mA LVTTL buffer is about 25 W. R must be 2.5 KW or greater to ensure rail-to-rail switching, with an error of 1% or less.
- Load Impedance: Keep the value of R low relative to the impedance of the load so that the current change through the capacitor due to loading becomes negligible.
- Time Constant: The filter time constant (τ = RC) must be high enough to greatly attenuate the individual pulses in the pulse string. On the other hand, a high time constant may also attenuate the desired low-frequency output signal. These potentially conflicting requirements are analyzed separately.

Pulse String Filtering

In the midrange voltages, signal DACout is switching rapidly, making it relatively easy to filter. When the DAC input is at 1 or the highest possible value, the signal DACoutDrvr is at the same level for all but one CLK cycle for each sample period. These are the most difficult output strings to filter; i.e., they are the *worst case*.

Although the filter noise may be calculated as an absolute peak-to-peak voltage, it is more useful to consider it as a fraction of the step voltage. The step voltage (V_S) is defined as the absolute change in V_{OUT} when the DAC input is incremented or decremented. For an 8-bit DAC, V_S equals (1/256) x V_{CCO}.

The worst-case peak-to-peak filter noise for an 8-bit DAC can be expressed as follows:

$$PPN_{FS} = (1-e^{-(1/f\tau)}) \times ((1-e^{-(255/f\tau)})/(1-e^{-(256/f\tau)})) \times 256$$

where:

PPN_{FS} is peak-to-peak noise expressed as a fraction of step voltage

f is the DAC clock frequency

τ is the filter time constant, RC.

For simplicity, we did not generalize this equation to handle any width DAC. For other widths, the equation for worst-case peak-to-peak noise is,

$$PPN_{FS} = (1-e^{-(1/f\tau)}) \times ((1-e^{-(y/f\tau)})/(1-e^{-(z/f\tau)})) \times z$$

where:

$y = 2^{C\_NUM\_DAC\_BITS} - 1$

$z = 2^{C\_NUM\_DAC\_BITS}$

This equation was used to create Figure 3, Figure 4 and Figure 5. These charts may be used to determine the value of RC for the desired worst-case noise voltage and operating frequency. For example, for an 8-bit DAC with a clock frequency of 80 MHz, the user might choose an RC value of 13.0×10^{-6}, corresponding to a peak-to-peak noise voltage of about 0.25 V_S. This leaves 0.75 V_S noise margin between steps. Part of this noise margin is needed to handle other noise sources such as noise on V_{CCO}.

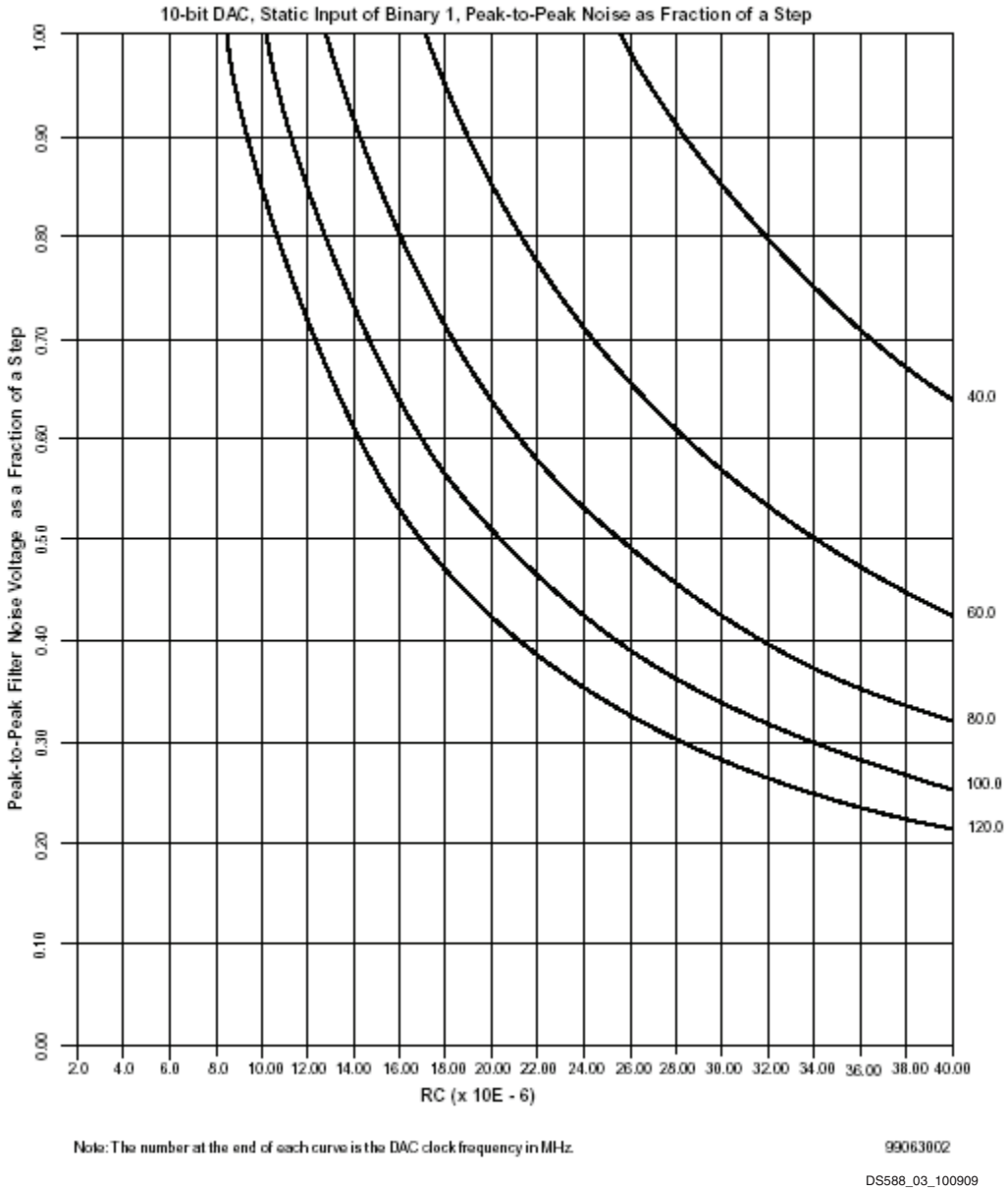

Figure 3: **Peak-to-Peak Noise as a Function of RC and Frequency (10-bit DAC)**

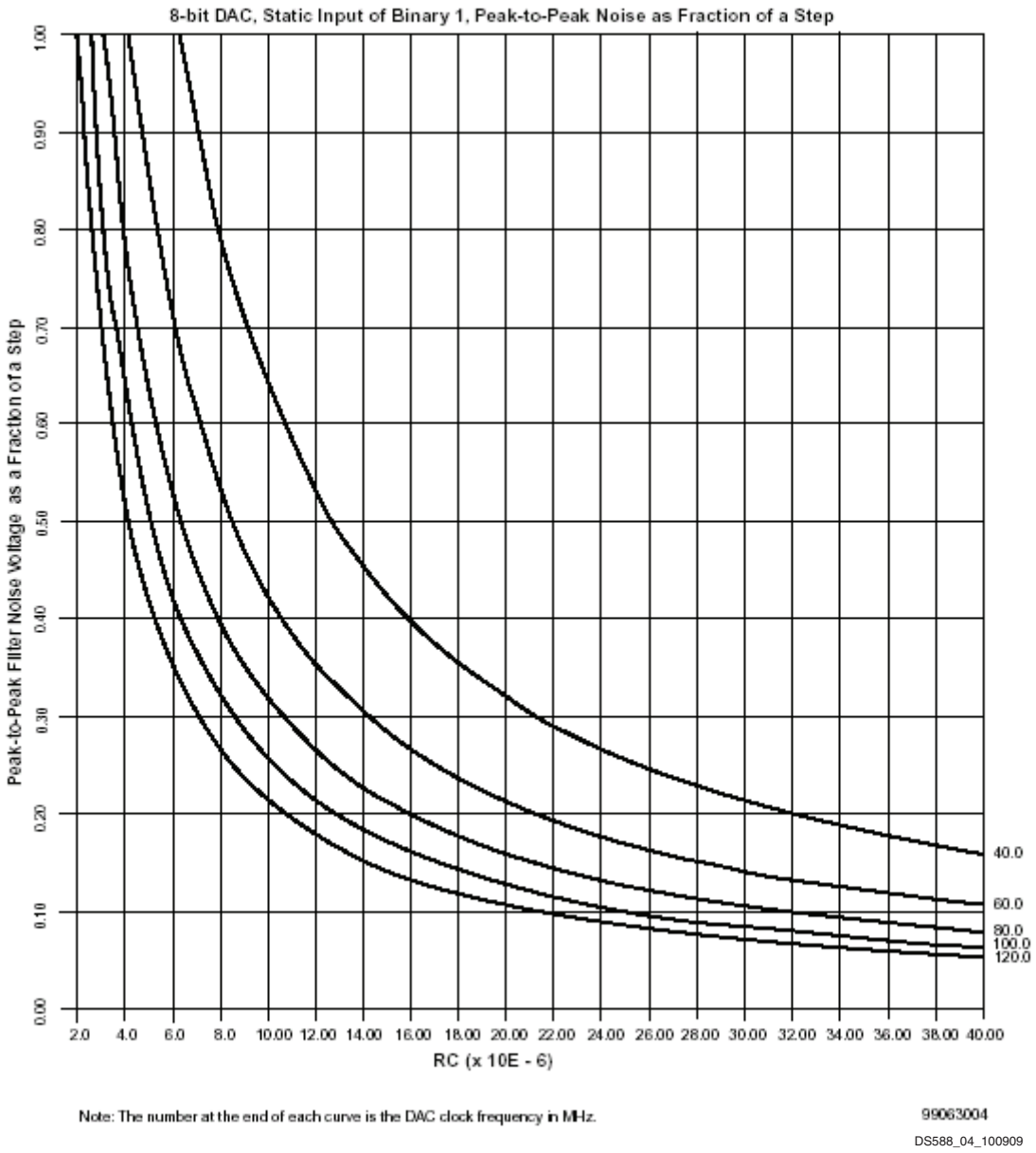

Figure 4: **Peak-to Peak Filter Noise as a Function of RC and Frequency (8-bit DAC)**

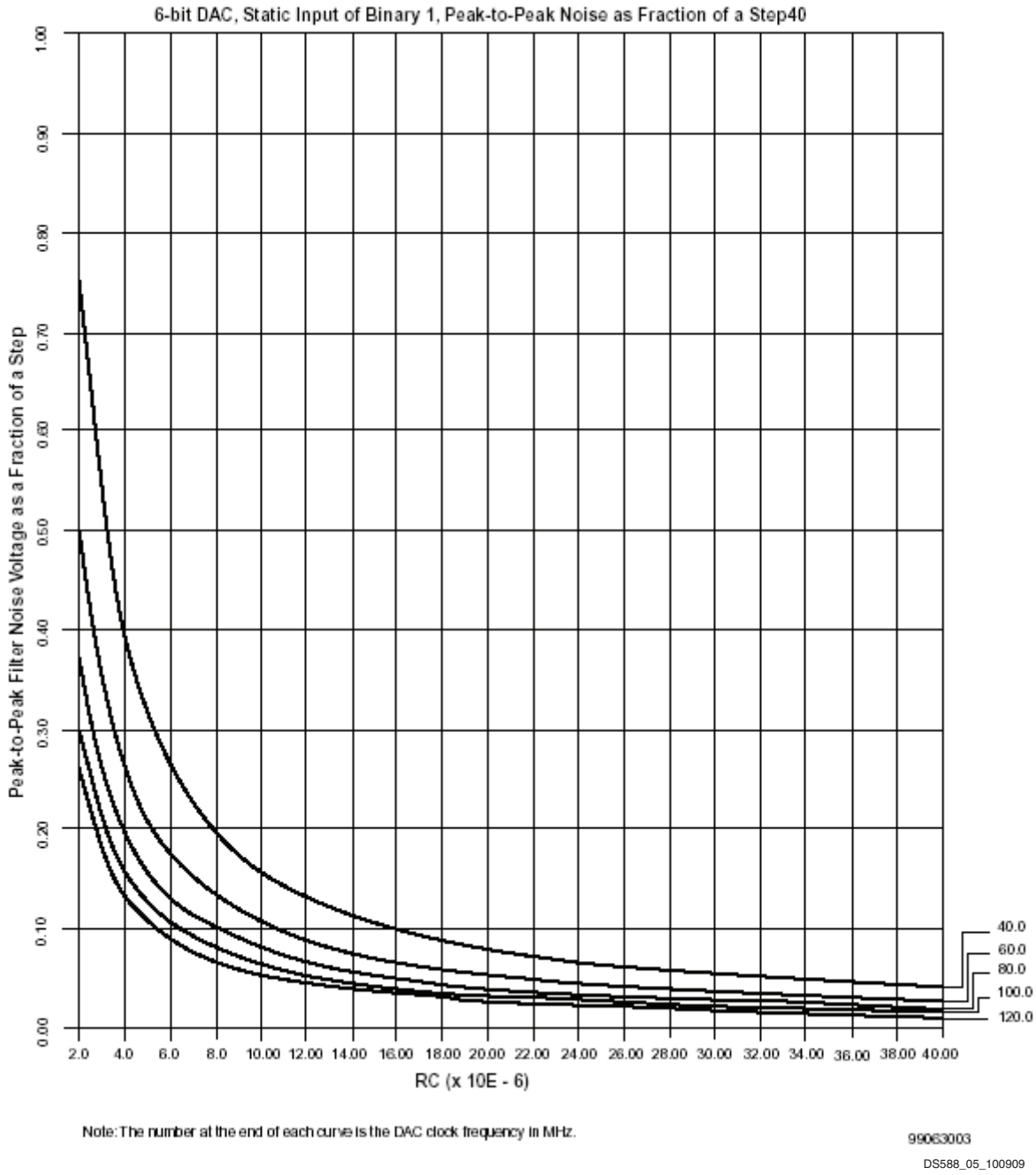

Figure 5: **Peak-to Peak Filter Noise as a Function of RC and Frequency (6-bit DAC)**

Output Attenuation

By convention, the cutoff frequency of a low pass filter is defined as the half-power point. The cutoff frequency of the simple RC filter may be expressed as:

$$f_C = 1/(2\pi\tau)$$

where:

f_C is the filter cutoff frequency

τ is the filter time constant, RC.

The above equation was used to create Figure 6.

Figure 6 may be used in conjunction with Figure 3, Figure 4, or Figure 6 to choose the RC time constant that is optimum for a particular application. All figures cover the same RC range.

Figure 4 shows an RC value of 13.0×10^{-6} results in a peak-to-peak noise voltage of 0.25 V when a DAC clock frequency of 80 MHz is used on a 8-bit DAC. From Figure 6, it can be determined that the filter cutoff frequency for this RC value is about 12 KHz. If the expected output is essentially a DC level, e.g., a programmable voltage generator, then RC may be increased to reduce the clock noise. On the other hand, if the fundamental frequency of the analog output is high, or it has sharp edges, then a lower RC may be needed. When determining the actual component values, remember that R should be at least 2500 Ω.

The user may implement a more sophisticated filter if the simple RC filter has inadequate cutoff or drive characteristics for the application.

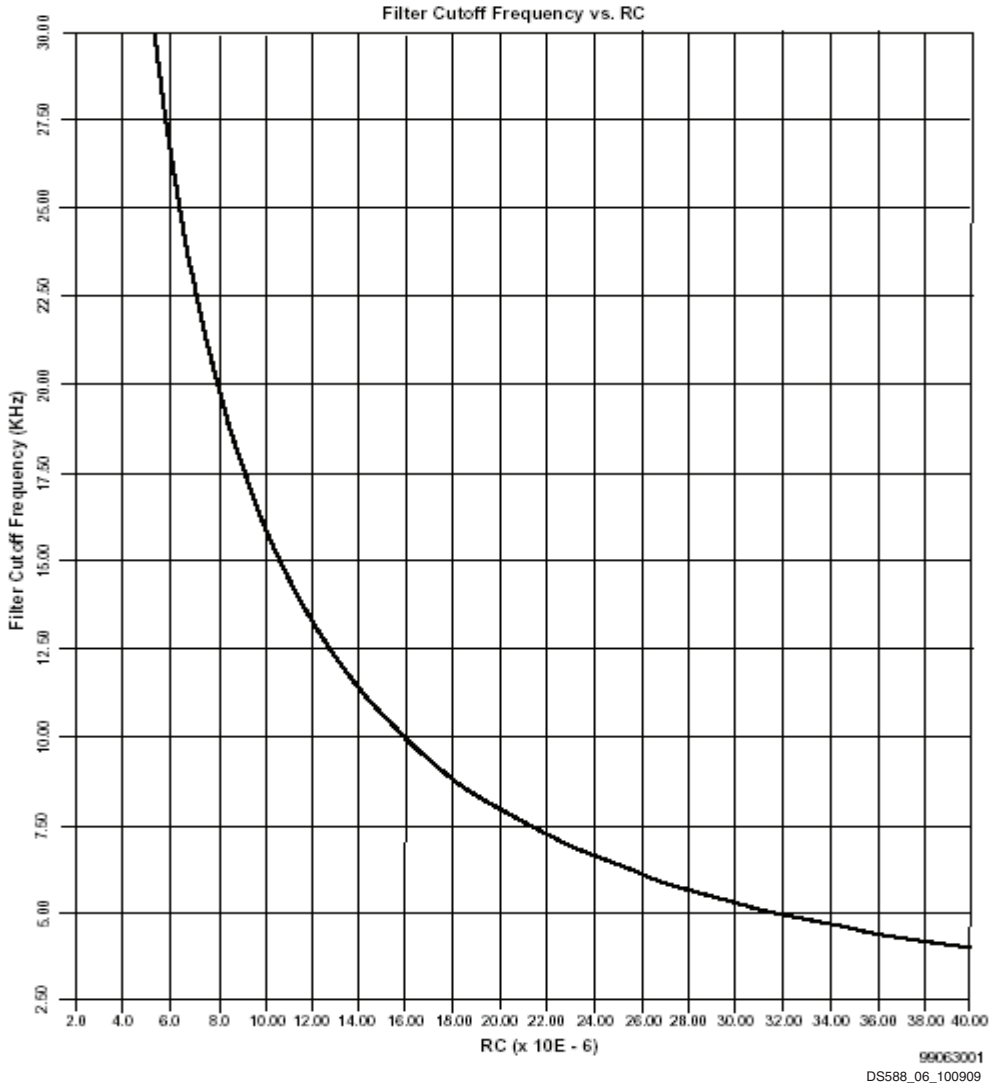

Figure 6: **Filter Cutoff Frequency as a Function of RC**

To resolve each DACin sample to the full precision of a Delta-Sigma DAC, the sample rate, i.e., the rate that DACin changes, must be less than or equal to $1/(2^{(C\_NUM\_DAC\_BITS)})$ of the CLK frequency. In some applications, such as a programmable voltage source, this is not an issue.

As DAC width and the desired sample frequency increases, it may not be possible to meet the above criterion. In practice, the sample rate sometimes exceeds $1/(2^{(C\_NUM\_DAC\_BITS)})$ of the CLK frequency. Though this compromises precision at higher frequencies, it is often possible to get satisfactory results. For example, the16-bit audio DACs in a CD system would require a clock frequency of 2.9 GHz for full resolution of the highest frequencies. In practice, a much lower clock frequency is used. One reason this is acceptable is because the sensitivity of the human ear to noise becomes lower as the frequency increases.

PLB Interface Module

PLB Interface Module provides bidirectional interface between Delta-Sigma DAC IP core and the PLB. The base element of the PLB Interface Module is slave attachment, which provides the basic functionality of PLB slave operation.

Interrupt Service Controller

The Interrupt Service Controller is a continuation of the Xilinx family of IBM CoreConnect™ compatible LogiCORE products. It provides interrupt capture support for the connected IP function. Interrrupt Service Controller provides the following functions:

- Parameterized number of interrupts needed by the IP.
- Provides both Interrupt Status Register (ISR) and Interrupt Enable Register (IER) functions for the user IP.

Flow Description

The following is a brief discussion on setting the DAC registers to initiate a D/A conversion.

To generate a analog value:

1. Write the data to be converted into the Data FIFO.
2. Enable the DAC by writing a "1" to the control register.
3. Drive Read_en high for one SPLB_Clk, the value written into the Data FIFO will start being converted.
4. If a new value is needed, write the new value into the Data FIFO.

Applications of DAC

This section lists some of the applications for the XPS Delta-Sigma DAC.

- **Programmable Voltage Generator**: A variable voltage between 0 V and V_{CCO} can be generated with a granularity determined by the bus width of DACin. In these applications, the voltage typically does not change quickly, so RC may be large to minimize noise.
- **VREF Generator**: This is a specific application of a Programmable Voltage Generator. For some SelectIO receivers standards, a reference voltage is required for each bank of receivers. If a DAC is used to generate this voltage, V_{REF} can be dynamically changed to verify operating margins when conducting system tests. See the Xilinx application note XAPP133 (Using the Virtex SelectI/O Resource) for more information on selectIO.
- **Waveform Generator**: Various analog waveforms, such as sine, sawtooth, triangle, etc., can be created by sequentially feeding the proper values to DACin. The values are normally pre-stored in SRAM. Block SelectRAM+ is ideal for this purpose. See Xilinx application note XAPP130 (Using the Virtex Block SelectRAM+ Features) for more information on Block SelectRAM+.
- **Sound Generator:** Delta-Sigma DACs are widely used in sound reproduction, speech synthesis, etc. Since the analog output is changing rapidly, RC must be chosen with an acceptable trade-off between noise and frequency response.
- **RGB Color Generator**: Although Delta-Sigma DACs are too slow to directly generate Red-Green-Blue signals for a raster display, they are applicable in some color generation systems that do not operate in real time.

- **Analog to Digital Conversion:** This DAC may be used as a voltage reference in an ADC. See XILINX application note XAPP155 (Analog to Digital Converter) for a complete discussion of this application.

XPS Delta-Sigma DAC I/O Signals

The XPS Delta-Sigma DAC I/O signals are listed and described in Table 1.

Table 1: **XPS Delta-Sigma DAC I/O Signals**

Port	Signal Name	Interface	I/O	Initial State	Description
System Signals					
P1	IP2INTC_Irpt	System	O	0	System interrupt
P2	SPLB_Clk	PLB	I	-	PLB clock
P3	SPLB_Rst	PLB	I	-	PLB reset
PLB Master Interface Signals					
P4	PLB_ABus[0 : C_SPLB_AWIDTH]	PLB	I	-	PLB address bus
P5	PLB_PAValid	PLB	I	-	PLB primary address valid indicator
P6	PLB_masterID[0 : C_SPLB_MID_WIDTH - 1]	PLB	I	-	PLB current master identifier
P7	PLB_RNW	PLB	I	-	PLB read not write
P8	PLB_BE[0 : C_SPLB_DWIDTH/8 - 1]	PLB	I	-	PLB byte enables
P9	PLB_size[0 : 3]	PLB	I	-	PLB transfer size
P10	PLB_type[0 : 2]	PLB	I	-	PLB transfer type
P11	PLB_wrDBus[0 : C_SPLB_DWIDTH - 1]	PLB	I	-	PLB write data bus
Unused PLB Master Interface Signals					
P12	PLB_UABus[0 : 31]	PLB	I	-	PLB upper address bits
P13	PLB_SAValid	PLB	I	-	PLB secondary address valid
P14	PLB_rdPrim	PLB	I	-	PLB secondary to primary read request indicator
P15	PLB_wrPrim	PLB	I	-	PLB secondary to primary write request indicator
P16	PLB_abort	PLB	I	-	PLB abort bus request
P17	PLB_busLock	PLB	I	-	PLB bus lock
P18	PLB_MSize[0 : 1]	PLB	I	-	PLB data bus width indicator
P19	PLB_TAttribute[0 : 15]	PLB	I	-	PLB transfer attribute
P20	PLB_lockerr	PLB	I	-	PLB lock error
P21	PLB_wrBurst	PLB	I	-	PLB burst write transfer
P22	PLB_rdBurst	PLB	I	-	PLB burst read transfer
P23	PLB_wrPendReq	PLB	I	-	PLB pending bus write request
P24	PLB_rdPendReq	PLB	I	-	PLB pending bus read request

Table 1: **XPS Delta-Sigma DAC I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P25	PLB_rdPendPri[0 : 1]	PLB	I	-	PLB pending read request priority
P26	PLB_wrPendPri[0 : 1]	PLB	I	-	PLB pending write request priority
P27	PLB_reqPri[0 : 1]	PLB	I	-	PLB current request priority
PLB Slave Interface Signals					
P28	Sl_addrAck	PLB	O	0	Slave address acknowledge
P29	Sl_SSize[0 : 1]	PLB	O	0	Slave data bus size
P30	Sl_wait	PLB	O	0	Slave wait indicator
P31	Sl_rearbitrate	PLB	O	0	Slave rearbitrate bus indicator
P32	Sl_wrDack	PLB	O	0	Slave write data acknowledge
P33	Sl_wrComp	PLB	O	0	Slave write transfer complete indicator
P34	Sl_rdBus[0 : C_SPLB_DWIDTH - 1]	PLB	O	0	Slave read data bus
P35	Sl_rdDAck	PLB	O	0	Slave read data acknowledge
P36	Sl_rdComp	PLB	O	0	Slave read transfer complete indicator
P37	Sl_MBusy[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave busy indicator
P38	Sl_MWrErr[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave write error indicator
P39	Sl_MRdErr[0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave read error indicator
Unused PLB Slave Interface Signals					
P40	Sl_wrBTerm	PLB	O	0	Slave terminate write burst transfer
P41	Sl_rdWdAddr[0 : 3]	PLB	O	0	Slave read word address
P42	Sl_rdBTerm	PLB	O	0	Slave terminate read burst transfer
P43	Sl_MIRQ[0 : C_SPLB_NUM_MASTERS -1]	PLB	O	0	Master interrupt request
DAC Signals					
P44	Dac_clk_en	System	I	-	DAC clock enable. This allows the SPLB_Clk to clock the sigma latch and the D flip-flop.
P45	Read_en	System	I	-	Read enable. A new value is read from the FIFO when this input is high.
P46	Dac_Out	System	O	0	DAC output. This is the pulse string that drives the external low pass filter.

Delta-Sigma DAC Design Parameters

To allow the user to create a XPS Delta-Sigma DAC that is uniquely tailored for the user's system, certain features are parameterizable in the XPS Delta-Sigma DAC design. This allows the user to have

a design that utilizes only the resources required by the system and runs at the best possible performance. Table 2 lists the features that are parameterizable in the XPS Delta-Sigma DAC core.

Table 2: XPS Delta-Sigma DAC Design Parameters

Generic	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameters					
G1	Target FPGA family	C_FAMILY	spartan3, spartan3e, spartan3a, spartan3adsp, aspartan3, aspartan3e, aspartan3a, aspartan3adsp,spartan 6, virtex4, qvirtex4, qvvirtex4, virtex5, virtex5fx, virtex6, virtex6cx	virtex5	string
PLB Parameters					
G2	High Address	C_HIGHADDR	Address range must be 2^n - 1 and greater than or equal to 0x1FF (3)	None (1) (2)	std_logic_ vector
G3	Base Address	C_BASEADDR	Valid Address Range (3)	None (1) (2)	std_logic_ vector
G4	PLB Data Bus Width	C_SPLB_DWIDTH	32, 64, 128	32	integer
G5	PLB Address Bus Width	C_SPLB_AWIDTH	32	32	integer
G6	PLB Point-to-Point or shared topology	C_SPLB_P2P	0, 1	0	integer
G7	PLB master ID bus width	C_SPLB_MID_WIDTH	$\log_2$(C_SPLB_NUM_M ASTERS) with a minimum value of 1	3	integer
G8	Number of PLB masters	C_SPLB_NUM_MASTER S	1 to 16	8	integer
G9	Width of slave data bus	C_SPLB_NATIVE_DWID TH	32	32	integer
G10	Burst support	C_SPLB_SUPPORTS_B URST	0	0	integer
Delta-Sigma DAC					
G11	Number of DAC bits	C_NUM_DAC_BITS	2 to 16	8	integer
G12	Allow the DAC output to go to full scale	C_FULL_RANGE	1 = The DAC output will go to full scale 0 = The DAC output will be 1 LSB less than full scale (4)	0	integer

1. No default value will be specified to insure that the actual value is set, i.e. if the value is not set, a compiler error will be generated.
2. For example, C_BASEADDR = 0xE0000000, C_HIGHADDR = 0xE00001FF
3. Address range specified by C_BASEADDR and C_HIGHADDR must be at least 0X200 and must be a power of 2.
4. With C_FULL_RANGE set to 0, the DAC output will be $(2^n - 1)/2^n$. Where n is the number of DAC bits, C_NUM_DAC_BITS.

XPS Delta-Sigma DAC Port Dependencies

The dependencies between the XPS Delta-Sigma DAC core design parameters and I/O signals are described in Table 3. In addition, when certain features are parameterized out of the design, the related logic will no longer be a part of the design. The unused input signals and related output signals are set to a specified value.

Table 3: **XPS Delta-Sigma DAC Parameter Port Dependencies**

Generic	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_SPLB_DWIDTH	P8, P11, P34	-	Width of the PLB Data Bus and PLB Slave Data Bus
G5	C_SPLB_AWIDTH	P4	-	Width of the PLB Address Bus
G7	C_SPLB_MID_WIDTH	P6	G8	$\log_2$(C_SPLB_NUM_MASTERS) with a minimum value of 1
I/O Signals				
P4	PLB_ABus[0 : C_SPLB_AWIDTH - 1]	-	G5	Width of the PLB Address Bus varies according to C_SPLB_AWIDTH
P6	PLB_masterID[0 : C_SPLB_MID_WIDTH - 1]	-	G7	Width of the PLB_mastedID varies according to C_SPLB_MID_WIDTH
P8	PLB_BE[0 : [C_SPLB_DWIDTH/8] - 1]	-	G4	Width of the PLB Byte Enable varies according to C_SPLB_DWIDTH
P11	PLB_wrDBus[0 : C_SPLB_DWIDTH - 1]	-	G	Width of the PLB WriteData Bus varies according to C_SPLB_DWIDTH
P34	Sl_rdBus[0 : C_SPLB_DWIDTH - 1]	-	G4	Width of the Slave Read Data Bus varies according to C_SPLB_DWIDTH
P37	Sl_MBusy[0 : C_SPLB_NUM_MASTERS - 1]	-	G8	Width of the Sl_MBusy varies according to C_SPLB_NUM_MASTERS
P38	Sl_MWrErr[0 : C_SPLB_NUM_MASTERS - 1]	-	G8	Width of the Sl_MWrErr varies according to C_SPLB_NUM_MASTERS
P39	Sl_MRdErr[0 : C_SPLB_NUM_MASTERS - 1]	-	G8	Width of the Sl_MRdErr varies according to C_SPLB_NUM_MASTERS

XPS Delta-Sigma DAC Register Descriptions

The internal registers of the XPS Delta-Sigma DAC are offset from the base address C_BASEADDR. The XPS Delta-Sigma DAC internal register set is described in Table 4.

Table 4: **XPS Delta-Sigma DAC Registers**

Register Name	Address	Access
Device Global Interrupt Enable Register (GIE)	C_BASEADDR + 0x01C	Read/Write
IP Interrupt Status Register (IPISR)	C_BASEADDR + 0x020	Read/Write
IP Interrupt Enable Register (IPIER)	C_BASEADDR + 0x028	Read/Write
Control Register (CR)	C_BASEADDR + 0x100	Read/Write
Data FIFO (FIFO)	C_BASEADDR + 0x104	Read/Write

Table 4: **XPS Delta-Sigma DAC Registers *(Cont'd)***

Register Name	Address	Access
Data FIFO Occupancy (OCCY)	C_BASEADDR + 0x108	Read
Data FIFO programmable depth interrupt Register (PIRQ)	C_BASEADDR + 0x10C	Read/Write

Control Register (CR)

The bit definitions for the register are shown inTable 5. The enable bit (EN) when set to "0"will prevent the XPS Delta-Sigma DAC from creating a pulse string, and the Dac_out will be zero.

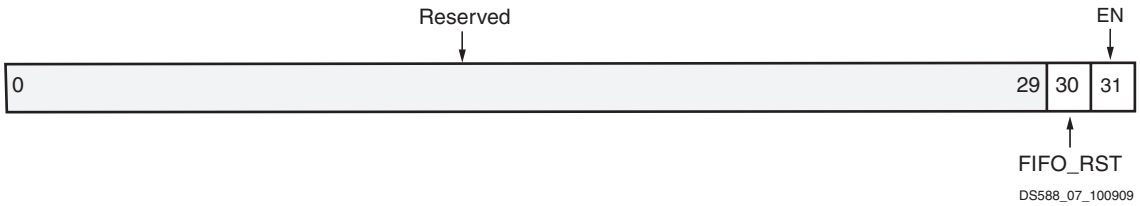

Figure 7: **XPS Delta-Sigma DAC Control Register**

Table 5: **XPS Delta-Sigma DAC Control Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0- 29	Reserved	-	NA	**Reserved.**
30	FIFO_RST	Read/Write	'0'	FIFO Reset '1' = Resets the Data FIFO. '0' = Data FIFO normal operation.
31	EN	Read/Write	'0'	**DAC Enable** '1' = Enables the XPS Delta-Sigma DAC. '0' = Resets and disables the XPS Delta-Sigma DAC. The DAC output will be zero.

Data FIFO

This 16 entry deep SRL FIFO contains data to be output by the XPS Delta-Sigma DAC. The Data FIFO is shown in Table 6. Reading of this location will result in reading the current word being output from the FIFO. Attempting to write to a full FIFO is not recommended and results in that data byte being lost.

When the Data FIFO is empty and the DAC is enabled, the DAC output will be zero.

Figure 8 shows the location for data on the PLB when C_NUM_DAC_BITS is set to 8 and C_FULL_RANGE is set to 0. If C_FULL_RANGE is set to 1 and C_NUM_DAC_BITS is set to 8 then data bits 23 through 31 will be used, but any value greater than 0X100 will result in an undefined DAC output.

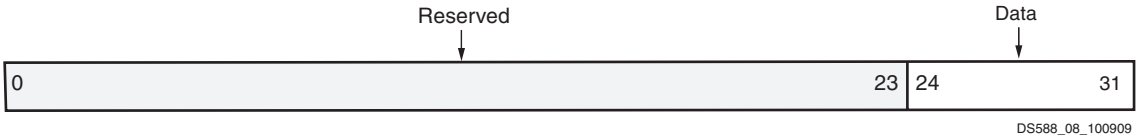

Figure 8: **Data FIFO**

Table 6: **XPS Delta-Sigma DAC Data FIFO Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
0 to [31 - C_NUM_DAC_BITS - C_FULL_RANGE]	Reserved	-	-	Reserved
[32 - C_NUM_DAC_BITS - C_FULL_RANGE] to 31	Data	Read/Write	Indeterminate	DAC Output Data

Data FIFO Occupancy Register (OCCY)

This field contains the occupancy value for the Data FIFO. Reading this register can be used to determine if the FIFO is empty, also the Data FIFO Empty Interrupt conveys that information. The value read is the binary count value, therefore reading all zeros implies that no location is filled and reading 10000 implies that all sixteen locations are filled. The bit definitions for the register are shown in Table 7.

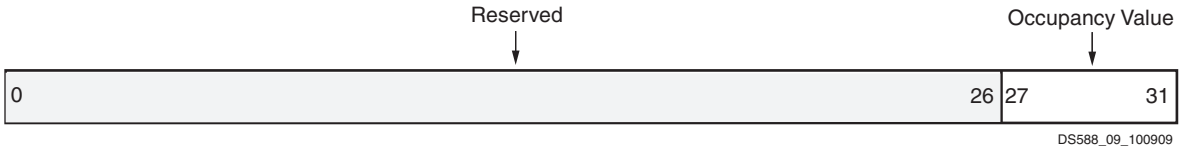

Figure 9: **Data FIFO Occupancy Register (OCCY)**

Table 7: **Data FIFO Occupancy Register Bit Definitions**

Bit(s)	Name	Access	Reset Value	Description
0 - 26	Reserved	NA	-	Reserved
27 -31	Occupancy Value	Read	0x0	Bit 27 is the MSB. A value of "10000" implies that all 16 locations in the FIFO are full.

Data FIFO Programmable Depth Interrupt Register (PIRQ)

This field contains the value which will cause the PIRQ Interrupt to be set. When this value is greater than or equal to the OCCY value, the PIRQ interrupt will be set and remain set until the equality or greater than is no longer true. The bit definitions are shown in Table 8.

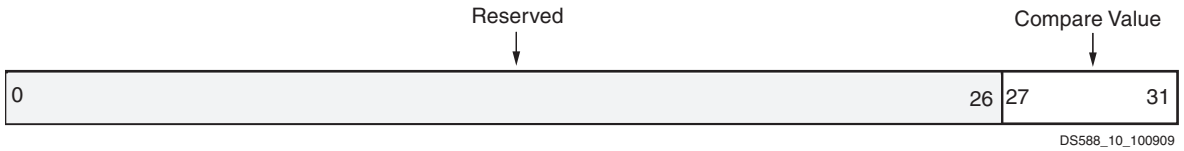

Figure 10: **Data FIFO Programmable Depth Interrupt Register**

Table 8: **Data FIFO Programmable Depth Interrupt Register Bit Definition**

Bit(s)	Name	Access	Reset Value	Description
0 - 26	Reserved	-	-	Reserved
27 - 31	Compare Value	Read/Write	0x0	Bit 27 is the MSB. A value of 00101 implies when either five or less than five locations in the Data FIFO are filled, FIFO PIRQ interrupt will be set.

XPS Delta-Sigma DAC Interrupt Descriptions

Interrupts

The interrupt signals generated by the XPS Delta-Sigma DAC are managed by the Interrupt Service Controller (ISC). This unit provides many of the features commonly provided for interrupt handling. The IPIER and IPISR contain the bit mapping as shown in Figure 11. Please refer to the *Processor IP Reference Guide* under Part 1 for a complete description of the GIE, IPISR and IPIER. The XPS Delta-Sigma DAC has two unique interrupts that are sent to the CPU. The number in the parenthesis is the interrupt bit number.

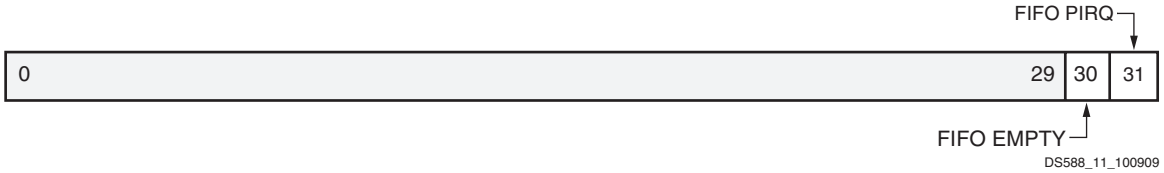

Figure 11: **Interrupt Mapping**

Table 9: **Data FIFO Interrupt Register Bit Definitions**

Bit(s)	Name	Access	Reset Value	Description
0 - 29	Reserved	NA	-	Reserved
30	FIFO EMPTY	Read/Write	0	This interrupt will be set and remain set as long the Data FIFO is empty.
31	FIFO PIRQ	Read/Write	0	This interrupt will be set and remain set as long the PIRQ is equal to, or greater than the OCCY. Clearing this interrupt requires that the Data FIFO be filled to a value greater than PIRQ.

Flow Description

The subsequent steps are required to set the DAC registers to initiate a conversion.

1. Initialize the interrupt registers GIE and IPIER as required, if interrupts are to be used. See the Processor IP Reference Guide for a complete description of the interrupt registers.
2. Write the data to be converted into the Data FIFO.
3. Set the PIRQ to generate an interrupt before the FIFO is empty.
4. Enable the DAC by writing a '1' to the control register.
5. Drive Read_en high for one SPLB_Clk. The first value written into the Data FIFO will start being converted two Dac_Clk_en later.
6. After the appropriate number of DAC_Clk_en have occurred drive the Read_en high for one clock and the next value will start being converted two Dac_Clk_en later.
7. If the Read_en signal is high and the Data FIFO is empty the DACout will be driven to '0'. DACout will remain zero until data is written to the Data FIFO and a Read_en occurs.

Timing Diagram

Following diagram shows the generation of the control signals for the conversion.

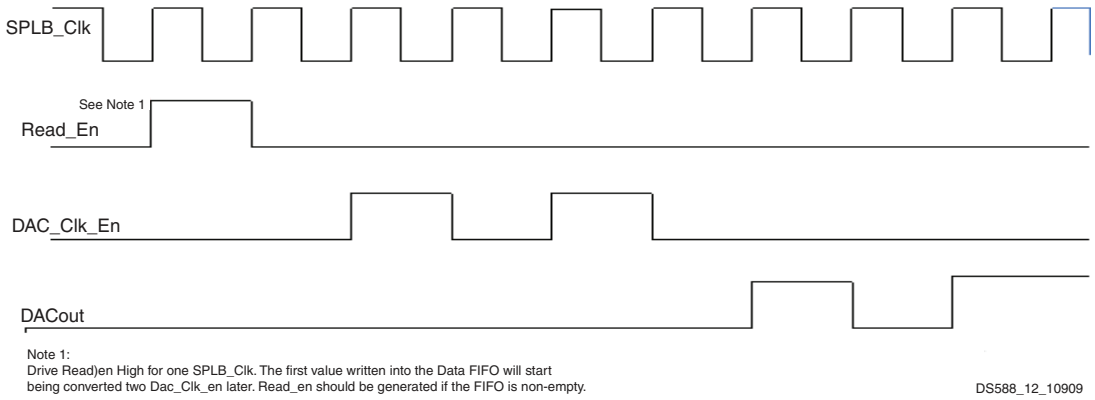

Figure 12: **Control Signal Generation for DAC**

Design Implementation

Device Utilization and Performance Benchmarks

Core Performance

Since the XPS Delta-Sigma DAC core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When the XPS Delta-Sigma DAC core is combined with other designs in the system, the utilization of FPGA resources and timing of the XPS Delta-Sigma DAC design will vary from the results reported here.

The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Virtex-4 FPGA as the target device are detailed in Table 10

Table 10: **Performance and Resource Utilization Benchmarks on the Virtex-4 FPGA (xc4vlx25-ff668-11)**

Parameter Values						Device Resources			Performance
C_NUM_DAC_BITS	C_FULL_RANGE	C_SPLB_NUM_MASTERS	C_SPLB_P2P	C_SPLB_DWIDTH	C_SPLB_MID_WIDTH	Slices	Slice Flip-Flops	LUTs	f_{MAX} (MHz)
2	0	1	1	32	1	165	143	175	198
4	0	4	0	64	2	240	237	170	198
8	0	8	0	128	3	287	295	200	186

Table 10: **Performance and Resource Utilization Benchmarks on the Virtex-4 FPGA (xc4vlx25-ff668-11) *(Cont'd)***

8	1	8	0	128	3	271	302	203	167
16	0	8	0	64	3	323	346	222	166
16	1	16	0	128	4	401	411	241	203

The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Virtex-5 FPGA as the target device are detailed in Table 11.

Table 11: **Performance and Resource Utilization Benchmarks on the Virtex-5 FPGA (xc5vlx30-ff676-2)**

Parameter Values						Device Resources		Performance
C_NUM_DAC_BITS	C_FULL_RANGE	C_SPLB_NUM_MASTERS	C_SPLB_P2P	C_SPLB_DWIDTH	C_SPLB_MID_WIDTH	Slice Flip-Flops	LUTs	f_{MAX} (MHz)
2	0	1	1	32	1	143	102	229
4	0	4	0	64	2	237	129	225
8	0	8	0	128	3	295	149	214
8	1	8	0	128	3	301	151	225
16	0	8	0	64	3	343	173	236
16	1	16	0	128	4	408	199	204

The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Spartan-3A DSP FPGA as the target device are detailed in Table 12.

Table 12: **Performance and Resource Utilization Benchmarks on the Spartan-3A DSP FPGA (xc3sd3400a-fg676-4)**

Parameter Values						Device Resources			Performance
C_NUM_DAC_BITS	C_FULL_RANGE	C_SPLB_NUM_MASTERS	C_SPLB_P2P	C_SPLB_DWIDTH	C_SPLB_MID_WIDTH	Slices	Slice Flip-Flops	LUTs	f_{MAX} (MHz)

Table 12: **Performance and Resource Utilization Benchmarks on the Spartan-3A DSP FPGA (xc3sd3400a-fg676-4) *(Cont'd)***

2	0	1	1	32	1	125	143	120	118
4	0	4	0	64	2	269	237	143	115
8	0	8	0	128	3	319	295	165	103
8	1	8	0	128	3	339	301	168	102
16	0	8	0	64	3	368	346	185	112
16	1	16	0	128	4	457	411	221	102

The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Virtex-6 FPGA as the target device are detailed in Table 13.

Table 13: **Performance and Resource Utilization Benchmarks on the Virtex-6 FPGA (xc6vlx195t-1-ff1156)**

Parameter Values						Device Resources			Performance
C_NUM_DAC_BITS	C_FULL_RANGE	C_SPLB_NUM_MASTERS	C_SPLB_P2P	C_SPLB_DWIDTH	C_SPLB_MID_WIDTH	Slices	Slice Flip-Flops	LUTs	f_{MAX} (MHz)
2	0	1	1	32	1	61	88	123	168
4	0	4	0	64	2	75	161	176	206
8	0	8	0	128	3	89	192	217	219
8	1	8	0	128	3	95	196	220	168
16	0	8	0	64	3	98	232	253	197
16	1	16	0	128	4	108	262	288	204

The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Spartan-6 FPGA as the target device are detailed in Table 14.

Table 14: **Performance and Resource Utilization Benchmarks on the Spartan-6 FPGA (xc6slxt100-2-fgg676)**

Parameter Values						Device Resources			Performance
C_NUM_DAC_BITS	C_FULL_RANGE	C_SPLB_NUM_MASTERS	C_SPLB_P2P	C_SPLB_DWIDTH	C_SPLB_MID_WIDTH	Slices	Slice Flip-Flops	LUTs	f_{MAX} (MHz)
2	0	1	1	32	1	53	88	110	100
4	0	4	0	64	2	82	161	155	110
8	0	8	0	128	3	95	192	185	110
8	1	8	0	128	3	95	196	186	100
16	0	8	0	64	3	99	232	206	100
16	1	16	0	128	4	100	262	218	100

System Performance

To measure the system performance (F_{MAX}) of this core, this core was added to a Virtex-4 FPGA system, a Virtex-5 FPGA system, and a Spartan-3A DSP FPGA system as the Device Under Test (DUT) as shown in Figure 13, Figure 14, and Figure 15.

Because the XPS Delta-Sigma DAC core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the core design will vary from the results reported here.

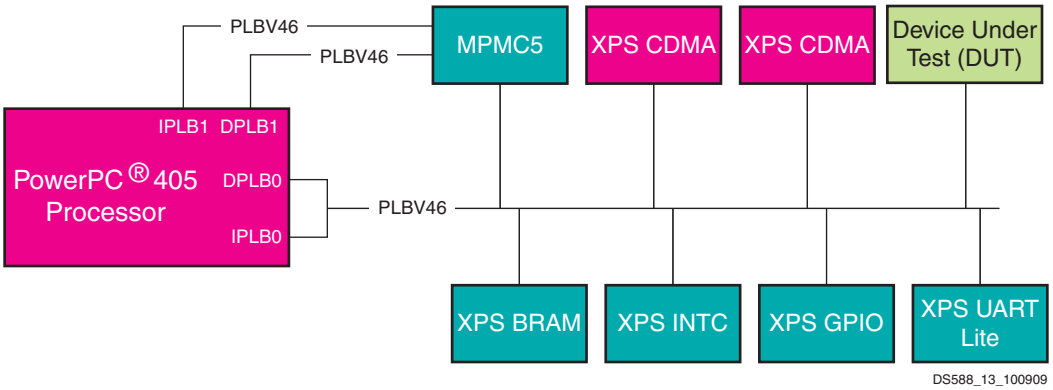

Figure 13: **Virtex-4 FX FPGA System**

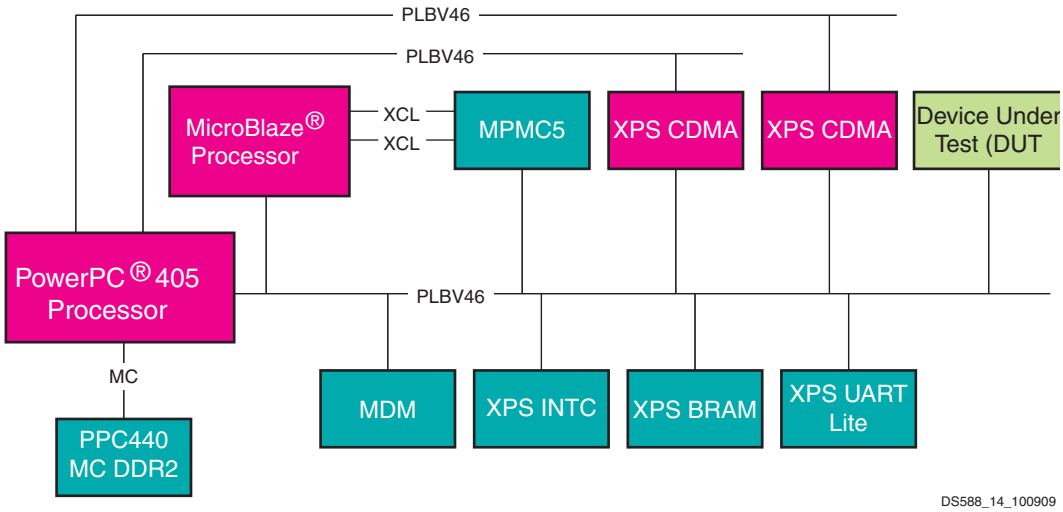

Figure 14: **Virtex-5 FX FPGA System**

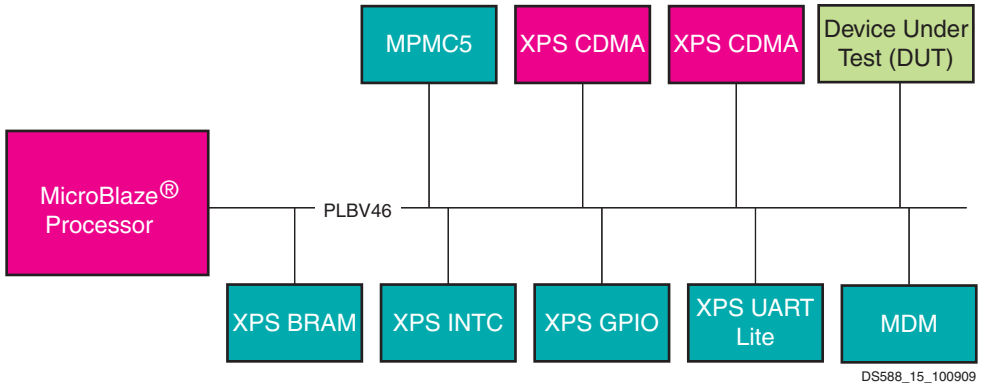

Figure 15: **Spartan-3A DSP FPGA System**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 15.

Table 15: **XPS Delta-Sigma DAC System Performance**

Target FPGA	Target F_{MAX} (MHz)
S3D3400 -4	100
V4LX25 -11	125
V5FXT70 -1	150

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Reference Documents

1. DS516 *Interrupt Control*
2. XAPP154 *Virtex Synthesizable Delta-Sigma DAC*
3. Analog Devices Data Converter Reference Manual, Volume I, 1992.
4. *High Performance Stereo Bit-Stream DAC with Digital Filter*, R. Finck, IEEE Transactions on Consumer Electronics, Vol. 35, No. 4,Nov. 1989

IBM CoreConnect 128-Bit Processor Local Bus: Architecture Specifications version 4.6

Support

Xilinx provides technical support for this LogiCORE product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Revision History

Date	Version	Description of Revisions
10/10/06	1.0	Initial Xilinx release
4/20/07	1.1	Added SP-3 support.
10/1/07	1.2	Added FMax Margin <RD Red>The XPS Delta-Sigma DAC resource utilization for various parameter combinations measured with the Virtex-6 FPGA as the target device are detailed in Table 13. section.
11/27/07	1.3	Added SP-3A DSP support.
1/14/08	1.3.1	Corrected spelling of virtex2p in Table 2.
4/15/08	1.4	Added Automotive Spartan-3E, Automotive Spartan-3, Automotive Spartan-3A, and Automotive Spartan-3 DSP support.
7/22/08	1.5	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant FPGA support.
9/25/08	1.6	Updated for version xps_deltasigma_dac_v1_01_a.
4/24/09	1.7	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
7/20/09	1.8	S6/V6 resource utilization tables added
12/2/09	1.9	Listed supported devices families in LogiCORE Table; converted to new DS template.

Notice of Disclaimer