HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.co.uk
http://www.hunteng.co.uk
http://www.hunt-dsp.com

Digital Down Converter Demo/Framework for HERON modules with FPGA

Rev 1.2 T.Hollis 11/05/05

Introduction

The advent of larger and faster Xilinx FPGA's has opened up the field of digital signal processing. The large array of configurable logic blocks within the FPGA give great flexibility together with speed, once configured the FPGA is not as flexible as a processor but is much faster. For many DSP applications speed is important especially for the initial processing of the data, after which the data rate reduces and becomes more manageable. Digital Down Conversion is a technique that takes a band limited high sample rate digitised signal, mixes the signal to a lower frequency and reduces the sample rate while retaining all the information.
The main advantage of using an FPGA to implement the Digital Down Converter is the speed, but it also has advantages associated with any digital signal processing system in that once it is defined it is fixed relative to the sample frequency, and will not change with time or temperature. For example once the coefficients for a digital filter have been set the characteristics of that filter are defined, if an identical filter is required then implement the same length filter with the same coefficients. Identical filters are particularly useful in quadrature sampling systems.
The use of Field Programmable Gate Arrays means that while developing your system if, for example, the filter characteristics are not quite right all that is required is to reconfigure the FPGA with a different filter. Dedicated hardware would most likely require component changes.
HUNT ENGINEERING manufacture FPGA modules together with A/D and D/A converters on them with sample rates of 100MSamples per Second, which makes them suitable for digital radio applications that would require Digital Down Conversion.
This DDC demo/Framework builds a DDC example for the IO2V2 FPGA module developed by HUNT ENGINEERING to show how a RF signal can be down converted. The RF signal for this demonstration is composed of an amplitude modulated signal plus a frequency modulated signal both shifted up in frequency by 10.0MHz. All of the source files for the FPGA and the Host program used can be found on the HUNT ENGINEERING CD under "Getting Started→Getting Started with FPGA modules and tools→DDC framework". Click on the "Files" link for the HERON-IO2V.

What is a DDC

In many applications the signal of interest may not be at the at the optimum part of the spectrum for processing, for example in a communications system the the signal band may only be narrow, KHz wide, but the signal band could be centered at RF frequencies, at many MHz. If the signal is sampled according to the Nyquist criterea, ie twice the highest frequency, then the data rate for the RF signal will be very high. Processing the data at this high rate is both difficult and expensive in terms of the amount of hardware required. A DDC will select the frequency band of interest and shift it down in frequency, which allows the data rate required to retain all the information to be much lower, and consequently reduces the complexity of any further processing.

A DDC can be split into two main sections, first a digital mixer to frequency shift the spectrum of the signal, and then filters to remove unwanted specral components and allow reduction, or decimation, of the data rate. The processing in the example DDC is based on quadrature sampling, the amount of hardware is doubled, as there is now both Phase and Quadrature channels, but the data rate required to retain all the information is halved. Quadrature sampling has the advantage that the negative frequency components cancel.

The Phase and Quadrature outputs of the DDC could be linked to other CoreGen blocks such as a complex FFT to analyse the spectrum, or a CORDIC block to extract the magnitude and/or phase from the complex signal.

Demonstration Concept

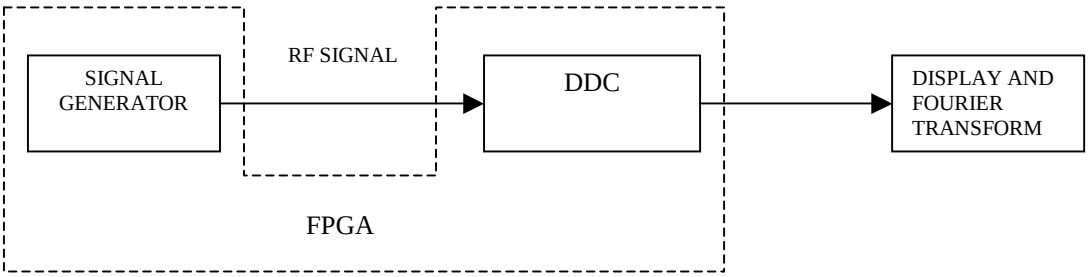


Figure 1 – Demonstration Concept

The demonstration shows how an RF signal, generated within the FPGA and output via one of the DAC's on the IO2V2, can be Digitally Down Converted to baseband. The output of the DDC is then Fourier transformed and compared with the same size Fourier transform of the RF signal.

This allows the demo to be run by simply connecting the outputs of the HERON-IO2V to its inputs – removing the need for test equipment to evaluate the DDC.

The Demonstration implementation can be split into three sections, first there is a signal generator in the FPGA that takes a waveform stored in RAM and frequency shifts if up by 10MHz this RF signal is then output onto DAC-A on the IO2V2. The second section is the DDC, from the HUNT ENGINEERING DDC example, in the FPGA which takes the signal from ADC-A on the IO2V2, sampled at 100Msamples per Second, and down converts it to base band. The sample values from the ADC-A and the DDC are then output on the HERON interface. The third section is a C++ program that generates the waveform values that are downloaded into the FPGA RAM for the signal generator and also grabs data from the HERON interface and displays it both as samples in time and as their Fourier Transform.

The Fourier Transform is 500 points so that the resolution on the RF signal will be 200kHz, while after the DDC the resolution is now 390Hz and the structure of the RF signal can now be seen.

Signal Generator

A block diagram of the signal generator is shown in figure 2. The object of the signal generator is to generate a composite signal centred on 10MHz.

There are two banks of RAM inside the FPGA that can be loaded over the HERON interface, with values describing the required signal waveforms. The two blocks of RAM are loaded in the demonstration with an Amplitude Modulated (AM) waveform and a Frequency Modulated (FM) waveform respectively. These RAMs are read at a rate of 1.5625 MHz with the address incrementing after each read, when the maximum address value is reached the next address starts again at zero. The adder combines the waveforms stored in the RAMs to give a composite signal with both AM and FM. In this demonstration the AM signal is on a 61.035kHz carrier with a 6.1035kHz modulation, and the FM signal is on a 33.569kHz carrier with a 3.051 kHz modulation frequency.

Before this signal can be shifted up to 10MHz the sample frequency must be increased, this is achieved using an interpolating Cascaded Integrator-Comb (CIC) filter to increase the sample frequency by a factor of 64 and remove the unwanted spectral components. The output is now the same composite signal but now sampled at 100Msamples per Second. To shift this signal up to 10MHz the composite signal is mixed, or multiplied, by a 10MHz carrier. The carrier is generated using a Direct Digital Synthesiser (DDS) with a fixed output frequency of 10.0MHz at a sample rate of 100MHz. The output of the mixer is connected to DAC-A on the IO2V2 to produce an analogue output signal. For this demonstration the DAC-A output should be connected to the ADC-A input on the IO2V2 as this is the input to the DDC.

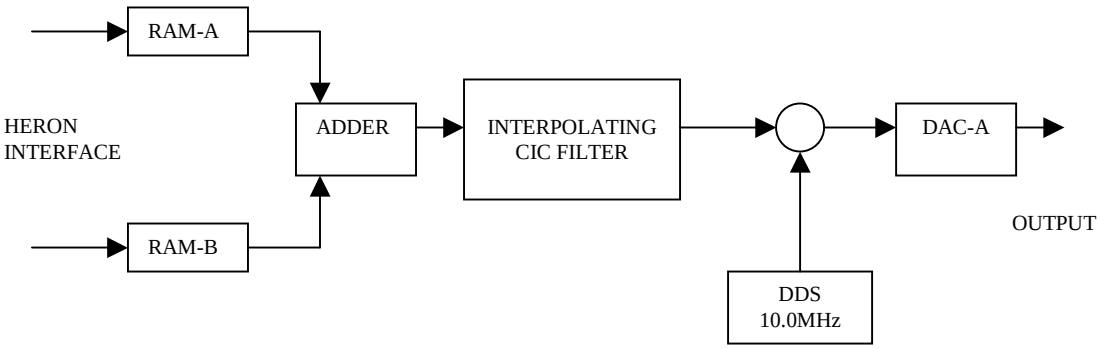


Figure 2 – Signal Generator

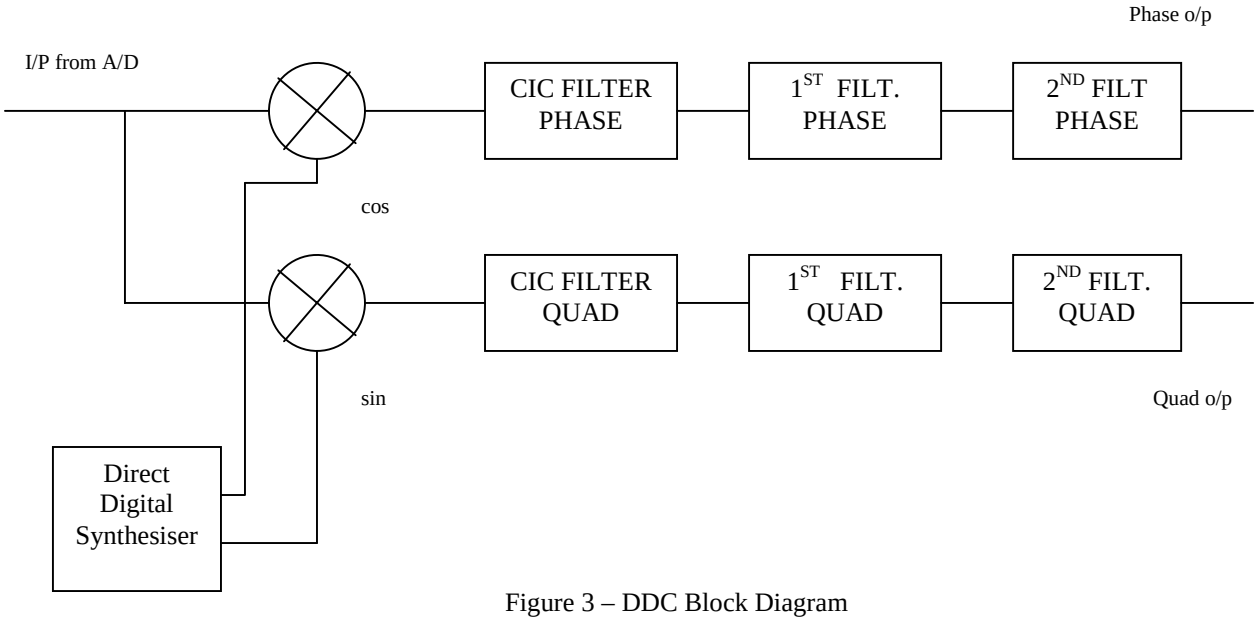


Figure 3 – DDC Block Diagram

DDC

The block diagram in figure 3 shows the basic structure of the DDC implemented in this example. The interface between each of the blocks in the example DDC has been made signed 14 bits, which makes it possible to use the DAC's on the IO2V2 to observe the signals. To this end the output from each stage goes not only to the next stage but also to a multiplexor that allows one of the stage outputs to be selected and linked to DAC's and to the Heron interface. The output selected by multiplexor is determined by a register that can be written to using the HSB. If the HSB is not available the multiplexor output defaults to the output of the second filter stage,the output of the DDC.

In this example the digital mixer is made up of two digital multipliers, one for the Phase signal and the other for the Quadrature signal, together with a Direct Digital Synthesiser that generates Phase(cosine) and Quadrature(sine) signals to Mix with the digitised input samples. The input data stream is from a single A/D converter, A/D converter 'A', and the mixers are used to generate the Phase and Quadrature signals. The default frequency of the DDS is 10MHz, but this is controlled by a value held in registers that can be written to by the HSB.

The DDC has three filter, and decimation stages. The first is a Cascaded Integrator Comb(CIC) filter, which will give the coarse filtering and high decimation, the remaining two filter stages allow for spectral refinement and further decimation.

The initial object of this example is to demonstrate the operation of a DDC in an Xilinx FPGA so the suggested signal levels and frequencies are to this end rather than for a particular system. Once the basic concepts are understood then the frequencies, or the DDC blocks can be modified and resulting effects observed.

The output of the DDC for this demonstration has been buffered in a FIFO so that when a block of results is requested by the software 500 samples of both the ADC-A output and the DDC Phase output can be transferred at the same time despite the sample rate difference between the ADC output and the DDC output . The output data transferred across the HERON interface has ADC data in the least significant half word and DDC output data in the most significant half word.

Host Software

We supply a pre-compiled Windows program for the PC, which can be used to communicate with the HERON-IO2, and to gather the data captured onto the PC where it is displayed.

The program is called 'hegraph.exe' and it is located in the 'host' sub-directory of this demonstration. The 'hegraph.exe' program is supplied as an exe file, but we have included the Microsoft Visual C/C++ 6 project of this program as well. This allows you to change it and re-compile it as you want.

The code that controls the waveform generation that is loaded into the RAM's is in the example2.c file. The amplitude modulation carrier has been made to have exactly 20 cycles in the 512 point sample window, while the frequency modulation carrier has exactly 11 cycles. The modulation frequency for the amplitude modulated signal has exactly 2 cycles in the 512 point sample window, while the modulation frequency for the frequency modulation has only one cycle. Having everything based on exact cycles in the sample window means that the output waveform when the RAM is read cyclically will not have any jumps or glitches in it.

Using the DDC Demonstration

The IO2V2 required for the demonstration should be fitted to a carrier board such as a HEPC9 or HEPC8. There should be a connection between the DAC-A output from the signal generator, and the ADC-A input to the DDC.

HEART carrier like HEPC9

If you have a HEART based module carrier like HEPC9, you can run the Host based demonstration code with your HERON-IO2 module fitted to any slot. Then you need to configure the HEART connections between the module and the Host using HeartConf. This is done from inside the hegraph program for you using the network file that is in the same directory as the host example software.

HEPC8 carrier

If you have an HEPC8, then you must fit your HERON-IO2 module to the first HERON slot as this is the only slot that has a FIFO connection to the Host machine. In this case the FIFO number the module uses is FIFO #1.

DDC Demonstration Bitstreams

The DDC demonstration for the HERON-IO2V2 is supplied on the HUNT ENGINEERING CD, along with bit streams that can be loaded directly onto the HERON-IO2V2. The bit streams can be found under 'fpga\io2v2\ddc_example' and the name of the bitstream file reflects the FPGA part number. I.e. 2v1000fg456 is for the HERON-IO2V2 when fitted to a HEPC9 carrier board. If you are using the HEPC8 carrier board you will need to use the bit stream that ends with '_pc8', so for this example, the correct bit stream will be 2v1000fg465_pc8.rbt.

The FPGA project files can be found in the ISE directory, and the SRC directory for the VHDL source and filter coefficient files. If you make changes to the project and re-build it you can change the functionality to be whatever you want.

Running the Host Software

The 'hegraph.exe' program is supplied as an exe file, but we have included the Microsoft Visual C/C++ 6 project of this program as well. This allows you to change it and re-compile it as you want.

After you have loaded the correct bit stream into your HERON-IO2 module, the "DONE" LED should be switched off showing that the configuration was successful. Also The USER LED4 should flash about once per second, showing that the clocks are properly running in the FPGA. If the DONE LED is off, but the LED is not flashing it may be because the Delay Locked Loop used in the clock circuit of the FPGA needs to be reset. You can do this using the utility under "programs → HUNT ENGINEERING → API board RESET" if you want, but such a reset will also be made when you start the hegraph program.

When you run the hegraph program you will see a window appear, that has some control menus available. Using these menus you will be able to 'start' the program (which will send HSB messages to correctly set the FIFO numbers that the FPGA should use, and also configure HEART if you have a HEART based module carrier). It will also load the RAMs in the FPGA with the AM and FM wave shapes before starting to capture and display the data coming from the IO2V2. The data is displayed both as a time sequences of samples and as their Fourier Transform. One channel shows the 'RF Signal' which is from the output of ADC-A, i.e. the input to the DDC, and the other channel shows the 'DDC Output'. A typical display is shown in figure 4.

Channel 1 of the output shows 500 samples of the RF Signal digitised by ADC-A at 100Msamples per Second

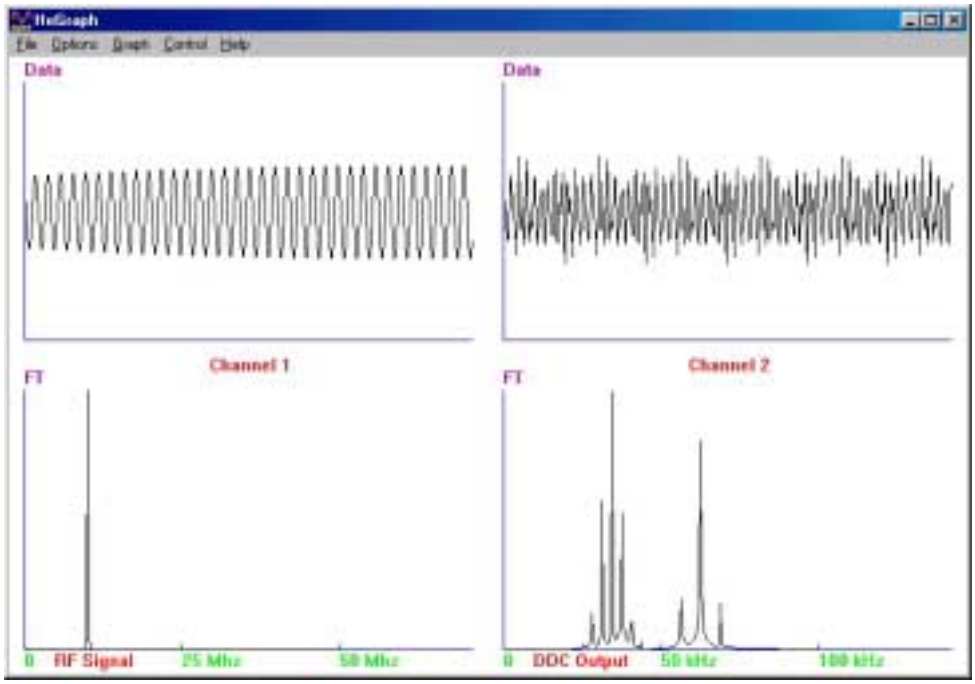


Figure 4 – Demonstration Output Display

(Fs), and below it is its Fourier transform. Because the Fourier transform has been performed on real data, as relative to complex data with both Phase and Quadrature components, the Fourier transform ouput is displayed only up to Fs/2 or 50MHz. The output for frequencies between Fs/2 and Fs are a mirror image of those below Fs/2.

With a sample frequency of 100MHz and a 500 point Fourier transform each resolution cell at the output of the transform is (100MHz / 500) = 200kHz wide, but both of the AM and FM signals are in a band of about 100kHzand so can not be individually resolved. To resolve down to a frequency of 1kHz with this sample frequency would require a 100000 point Fourier transform.

The other option to obtain the frequency structure of the signal is to use the Digital Down Converter to lower the sample frequency for the frequency band of interest. The Demonstration DDC moves a 100kHz band from up at 10MHz down to baseband and reduces the sample frequency to 195.31kHz. The same 500 point Fourier transform now has a resolution of (195.31kHz / 500) = 390.6Hz and now the structure of our composite signal is visable.

The AM signal with its single pair of side bands is shown at 61kHz, and the FM signal where approximately three pairs of significant side bands are visibleis shown at 33.5kHz. The modulation frequency of the AM signal is twice the modulation frequency of the FM signal and this can be seen in the relative spacing of the side bands.

The demonstration uses an Fourier transform on the output of the DDC to show the advantages of the lower sample frequency. Instead of a Fourier transform the output could have been filtered to separate the AM and FM signals prior to demodulation, this is much simpler at the lower sample rate.

DDC demonstration Specification

Signal Generator

RAM

RAM Size:	512x16bits
Ram read frequency:	1.5625MHz
Time to read all RAM :	(512 / 1.5625MHz) = 327.68 micro Seconds

Adder

Inputs bus width:	14 bits
Output:	registered 15 bits

CIC filter:

Input bus width :	14 bits
Number of stages:	5
Sample rate change:	64
Differential delay:	1

Multiplier:

Input bus width:	registered 12 bits
Output bus:	registered 24 bits
Multiplication rate:	100MHz

DDS:

Output:	12 bits, Sine only
Data width:	26 bits
Phase angle width:	10 bits
Phase increment(const):	0x666666
Accumulator latency:	1
Accumulator width:	26 bits

DAC

Output Update Rate:	100MHz
Digital Input:	14 bit binary
Analogue Output:	+/-1Volt max

Heron Interface:

Wave shape words written **to** the IO2V2:

MS half word	LS half word
0x ‘0001’(for Channel A RAM)	‘00’ & (14 bit value)
0x ‘0004’(for Channel B RAM)	‘00’ & (14 bit value)

Data **from** the IO2V2:

MS half word	LS half word
‘00’ & (14 bit of RF signal)	‘00’ & (14 bit of DDC Phase output)

AM / FM Signals:

AM carrier:	20 cycles in 327.68micro Seconds = 61.035kHz
FM carrier:	11 cycles in 327.68 microseconds = 33.569kHz
FM modulation:	1 cycle in 327.68 microseconds = 3.051kHz
FM modulation index:	1.0
AM modulation:	2 cycles in 327.68 microseconds = 6.103kHz
AM modulation index:	0.5

DDC

The DDC is implemented as described in the ‘Implementation of the DDC’ section of this document.

The Phase output from the DDC is buffered with a FIFO to align the ADC-A outputs, because of the different sample rates.

Oscillator Jitter

If the difference frequency is reduced, say to about 1kHz by reducing the RF signal generator frequency to 10.001MHz, then any jitter on the crystal oscillator, generating the sampling clock signal for the ADC, will become evident as noise on the DDC output signal.

FIGURE 10 – DDC Output with Poor Jitter

Figures 10 shows the output of the DDC with a difference frequency of 1kHz, using the standard 100MHz crystal oscillator to generate the sample frequency for the ADC. This oscillator has a jitter of 80 picoSeconds peak to peak.

Figure 11 shows the output of the DDC with the same difference frequency of 1kHz but with a low jitter 100MHz crystal oscillator to generate the sample frequency for the ADC. This oscillator has a jitter of 18 picoSeconds peak to peak.

The HERON-IO2V2 has ‘User Crystal Oscillator’ sites that can be used for oscillators to suit the application frequency and jitter requirements.

If the application requires the ADC’s to band pass sample the input signal then the effect of jitter will become worse.

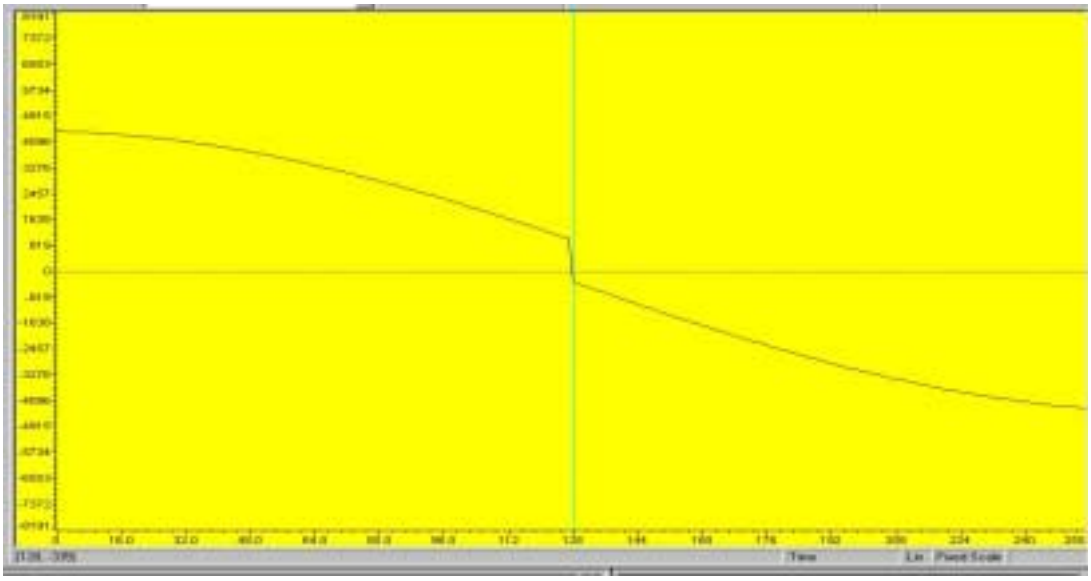


FIGURE 11 – DDC Output with Good Jitter

Implementation of the DDC

There is a DDC CoreGen block available which could be used instead of the individual CoreGen building blocks used in this example. The architecture of the DDC block is very similar to that of the DDC in the example and when implemented took up a similar amount of the FPGA.

Using individual blocks for this example has the advantage that the signals at every stage can be tapped off and displayed. In general the use of a single CoreGen DDC block, using several CoreGen blocks to make up a DDC or writing VHDL to implement the equivalent to the CoreGen block, but tailored to your application, depends to the level of control in the FPGA implementation your application requires.

DDC Specification

Input: The input to the DDC will be from the A/D converter 'A' on the IO2V2 module.

Sample Frequency: 100Msamples per Second

Sample Resolution: 12bit signed.

Analogue Input: +/- 1Volt Max

Direct Digital Synthesiser(DDS):

DDS Frequency(default): 10MHz

Frequency Resolution(delF): 1.5Hz

Spurious Free Dynamic Range: 60dB

Sine and Cosine output: 12 bit signed

Frequency Control : 26 bits default to give 10MHz, the value is held in registers at HSB **addresses 5,6,7 and 8**.

HSB address	Description
HSB address 5	LS 8 bits (D7 – D0)
HSB address 6	Next 8 bits (D15 – D8)
HSB address 7	Next 8 bits (D23 – D16)
HSB address 8	MS 2 bits (D25 – D24) MS--XXXXXX(D25)(D24)--LS

DDS FREQUENCY CONTROL REGISTERS

How to calculate the 26 bit value to be loaded into these registers is described in the Xilinx DDS data sheet.

Multipliers:

Input Resolution: 12 bit signed on both inputs.

Output Resolution: 24 bit signed.

One multiplication per sample clock cycle (100MHz)

The output of the multiplier is truncated to 14 bit signed with no gain.

Cascaded Integrator Comb(CIC) filter:

Number of Stages: 5

Decimation Rate: 16

Differential delay: 1

Output Resolution: 33 bit signed

The output of the multiplier is truncated to 14 bit signed with no gain.

First FIR Filter:

Cut off Frequency: 0.048Fs or 300kHz

Filter Length: 35 taps

Filter coefficients: symmetrical

Phase characteristic: linear

Decimation Rate: 8

Output Resolution: 32 bits

The output of the first filter is truncated to 14 bit signed with a gain of two.

Second FIR Filter:

Cut off Frequency: 0.128Fs or 100kHz.

Filter length: 35 taps

Filter Coefficients: non_symmetrical

Decimation Rate: 4

Output Resolution: 32 bits

The output of the second filter is truncated to 14 bit signed with a gain of 8.

Multiplexor:

The Multiplexor selects the signal that is linked to the DAC's and Heron Interface using the value in a register at **address 4** of the HSB. The default value in this register connects the second filter Phase and Quadrature outputs to the DAC's and Heron interface.

In the DDC demo the multiplexor stays in its default state and connects the second filter Phase output and the RF input signal to the DAC's and Heron interface.

REGISTER VALUE	CONNNECTION
XXXXX000	**ADC output** on DAC 'A' and LS half word of Heron interface output. **DDS Sine output** on DAC 'B' and MS half word of Heron interface output.
XXXXX001	**Phase Multiplier output** on DAC 'A' and LS half word of Heron interface output. **Quadrature Multiplier output** on DAC 'B' and MS half word of Heron interface output.
XXXXX010	**Phase CIC output** on DAC 'A' and LS half word of Heron interface output. **Quadrature CIC output** on DAC 'B' and MS half word of Heron interface output.

XXXXX011	**Phase First Filter output** on DAC 'A' and LS half word of Heron interface output. **Quadrature First Filter output** on DAC 'B' and MS half word of Heron interface output.
XXXXX100 XXXXX101 XXXXX110 XXXXX111	**Phase Second Filter output** on DAC 'A' and LS half word of Heron interface output. **Quadrature Second Filter output** on DAC 'B' and MS half word of Heron interface output. (Default output.)

MULTIPLEXOR CONTROL VALUES

The LEDs on the IO2V2 reflect the multiplexor control bits.

D/A Outputs:

Output Update Rate: 100MHz

Digital Input: 14 bit binary

Analogue Output: +/-1Volt max

Heron Interface:

The 32 bit data word to the Heron interface is split into two 16 bit half words :-

MS half word	LS half word
'00' & (14 bit of RF signal)	'00' & (14 bit of DDC Phase output)

DDC CORE GENERATOR BLOCKS

Direct Digital Synthesiser

The DDS is a Xilinx 'CoreGen' block which has a two page GUI to define the operation of the built core. If further information is required there is a data sheet available by selecting the button at the bottom of the GUI.

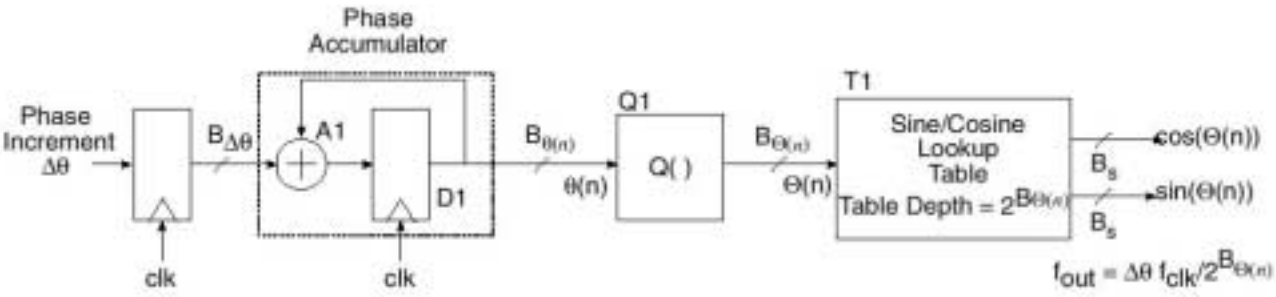


Figure 12 - Block Diagram of DDS

In this example the phase increment value is registered in and then its value is added into the accumulator on every cycle of the clock. The output value of the accumulator has a number of the least significant bits removed by the slicer Q1. The remaining bits address the sine/cosine look up table T1.

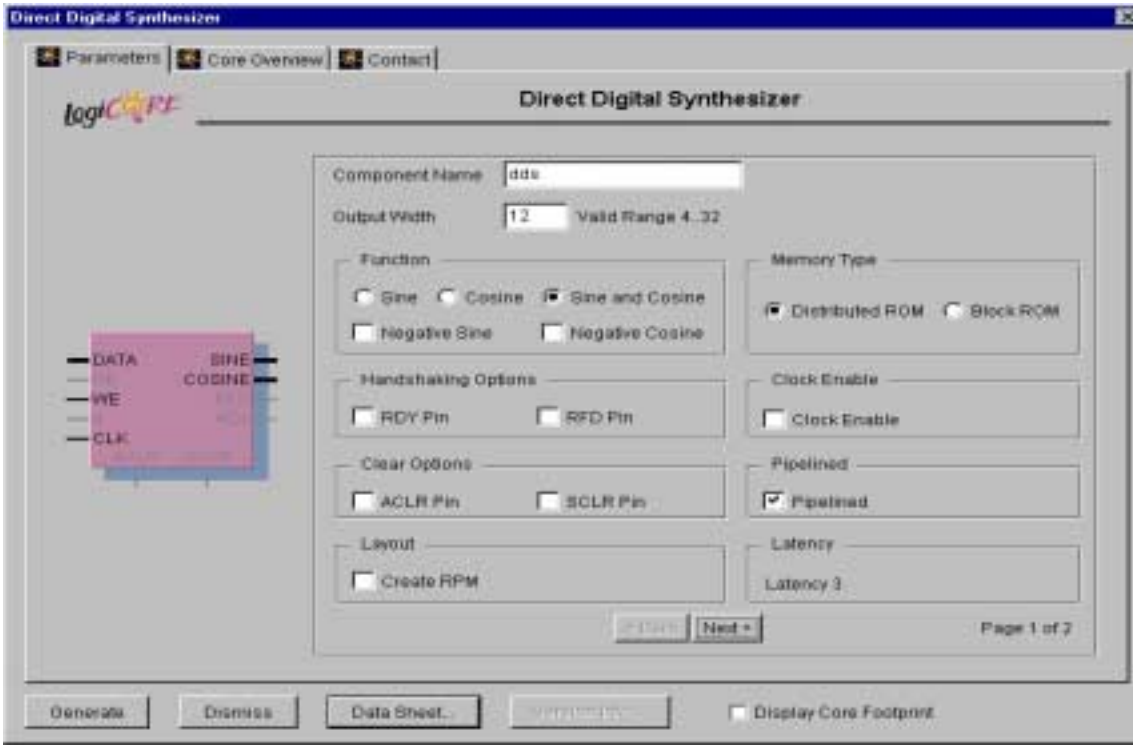


FIGURE 13 -- CoreGen DDS GUI Page 1

Output Width: 12Bits selected to match the A/D converter output width.

Function: 'Sine and Cosine' selected for quadrature multiplication.

Negative Sine or Cosine options have not been selected for this example as the output from the A/D converter is not in quadrature form.

Memory Type: Distributed ROM selected.

Handshake Options: No handshake options are required for this example as the outputs are updating on every clock cycle.

Clock Enable: The clock enable is not selected as the DDS is running at 100MHz Sample clock frequency.

Clear Options: Clear options are not selected for this example.

Pipelined: Pipelined option selected.

Layout: Create RPM option not selected.

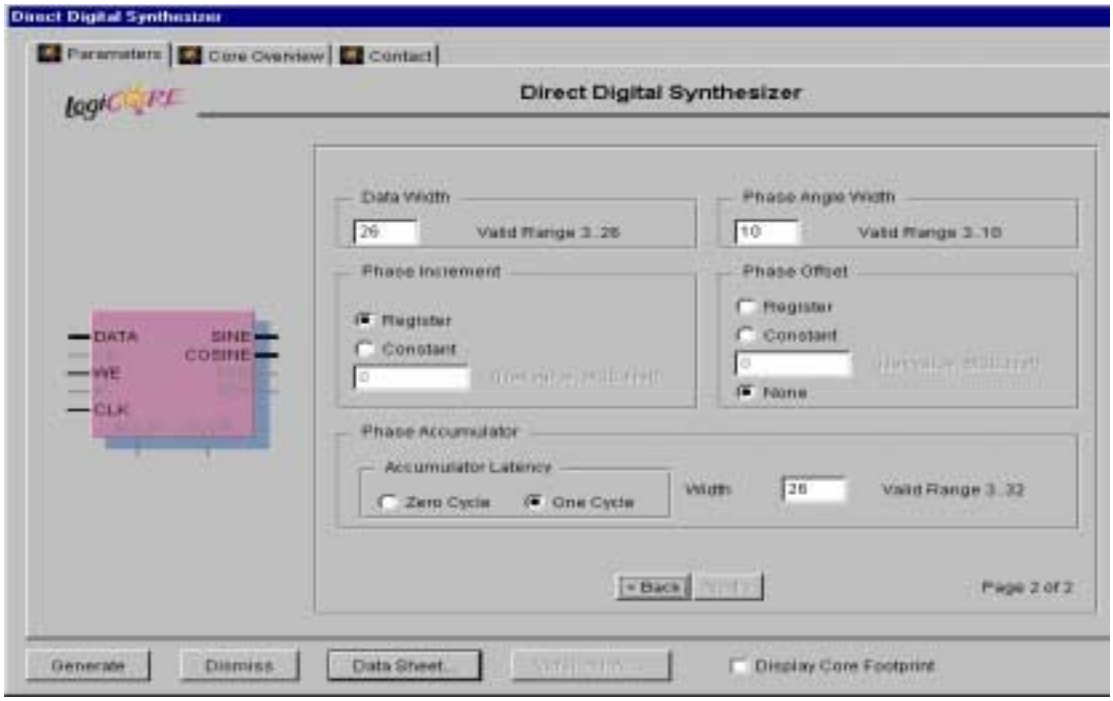


FIGURE 14 -- CoreGen DDS GUI Page 2

Data Width: This is the width of the input data bus used to define the phase increment, and has been set to the full accumulator width of 26 bits.

Phase Angle Width(Blut): This is the number of bits used to address the sine/cosine look up table and is directly related to the Spurious Free Dynamic Range by 6dB's per bit. So 10 bits gives the required 60dB Spurious free dynamic range.

Phase Increment: The register option has been selected so that the Phase Increment value is loaded via the 26 bit input data bus, using the Write Enable(WE). In this example it allows the frequency to be updated via the HSB. The Phase Increment is defined by the relationship:-

DelPhi = (Fout / Fclk) * 2^(Blut)

With Fout = 10MHz; Fclk = 100MHz; and Blut = 10bits

Then DelPhi = 102.4

This real number now needs to be converted into a binary number that con be loaded into the DDS via the data input. The value at the output of the accumulator is 26 bits of which only the most significant 10 bits are used to address the sine/cosine look up table and are the integer part of 'DelPhi', the remaining 16 bits are the fractional part. To generate the binary number to represent 102.4 :-

Multiply 102.4 by 2^(16) = 6710886.4

This has in effect moved the point by 16 binary places

Convert to binary =1100110 01100110 01100110

But this is only 23 bits as the most significant bits are zero, so the full 26 bit value is:-

0001100110 0110011001100110

Phase Offset: Not selected for this example.

Phase accumulator: 26 bit width with single cycle latency selected. The number of bits is determined by the frequency resolution relative to clock frequency:-

$$Bpa = \log(Fclk / delF) / \log(2)$$

With Fclk = 100MHz and delF = 1.5Hz

then Bpa = 26bits

Multipliers

There are two identical multipliers in the DDC, one for the Phase signal and the other for the Quadrature signal. The digitised signal from the A/D converter goes to both multipliers and is multiplied by the cosine output of the DDS for the Phase signal, and by the cosine output of the DDS for the Quadrature signal.

The multiplier is a CoreGen block called ‘ddc_mult’ with properties defined by the four GUI’s.

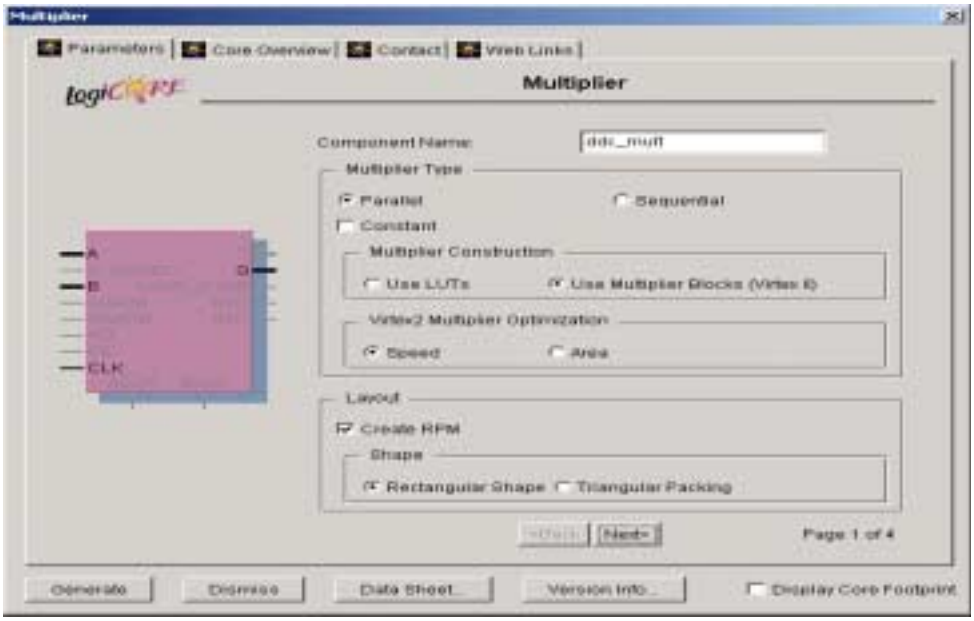


FIGURE 15 -- CoreGen Multiplier GUI Page 1

Multiplier Type: A parallel type has been selected as the multiplier will have to perform a multiplication on every sample clock cycle. Neither of the inputs are Constant , so this option has not been selected. The Multiplier Construction could be either using LUT’s or the Multiplier Blocks, as the example is for a VirtexII, with a high speed multiplier the best option is the Multiplier Block.

Optimisation: The multiplier is optimised for speed as it has to run at 100MHz.

Layout: The Create RPM option has been selected, with a rectangular shape.

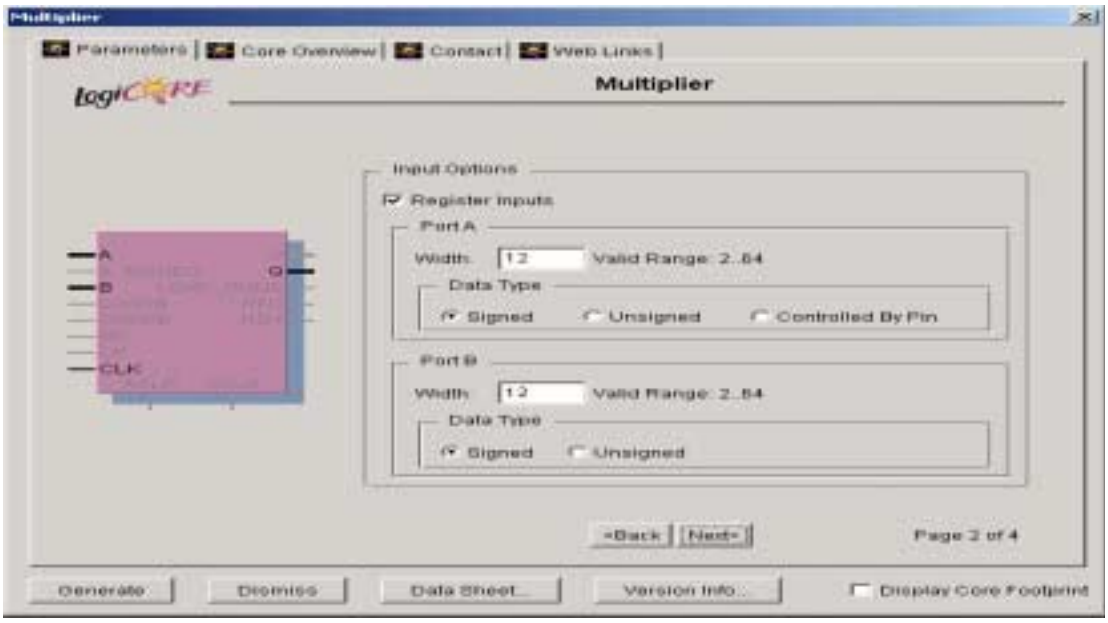


FIGURE 16 -- CoreGen Multiplier GUI Page 2

Input Options: The Register Inputs option has been selected as the safe option with the multiplications occurring on every sample clock cycle, the latency in this example is not important. Both inputs are 12 bit signed.

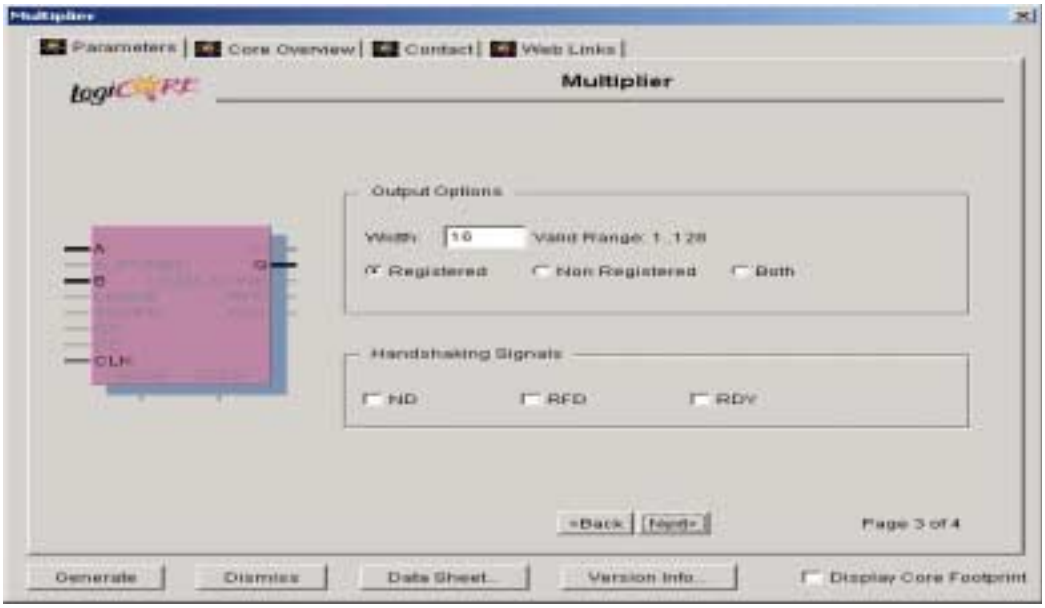


FIGURE 17 -- CoreGen Multiplier GUI Page 3

Output Options: The width of the two inputs to the multiplier are 12 bit signed so to give the full range of output 24 bits would be required, but in this case only 16 bits are selected. The least significant bits are truncated by the Core Generator. The safe option is to register the output.

Handshaking Options: No handshaking options have been selected as the multiplier is performing a multiplication on every cycle.

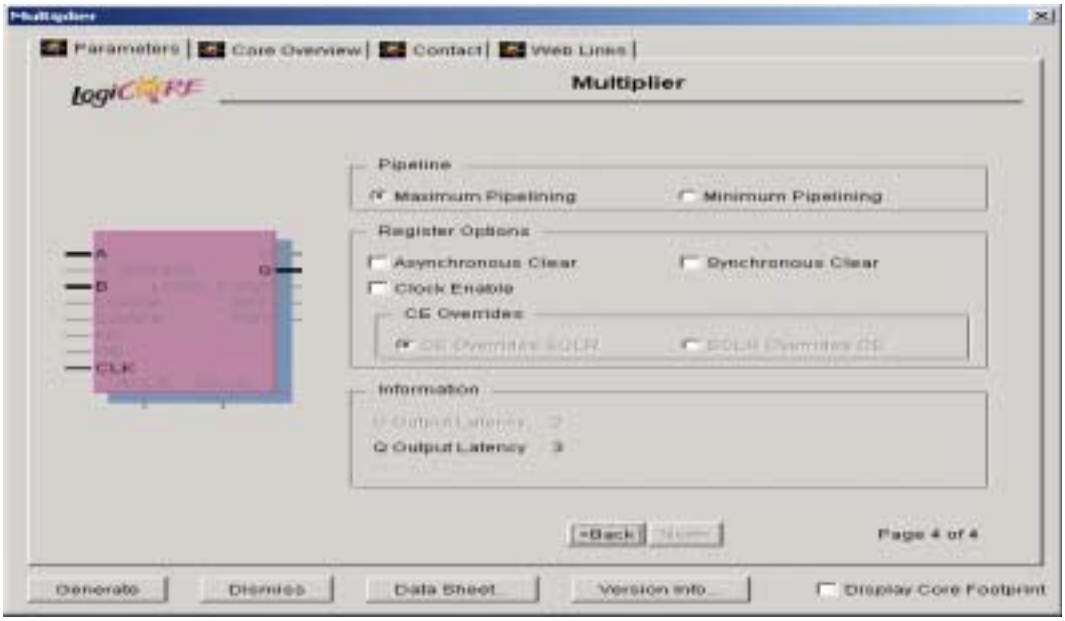


FIGURE 18 -- CoreGen Multiplier GUI Page 4

Pipeline: Maximum pipelining has been selected for high speed operation.

Register Options: For this high speed application none of the Register Options have been selected.

Information: With the inputs and output registered and maximum pipelining the latency is three clock cycles.

Cascaded Integrator Comb(CIC) Filter

CIC filters are multiplierless structures, consisting only of adders subtractors and registers. They are typically employed in narrowband applications that have large excess sample rates, to reduce the sample rate and filter off the unwanted frequency components. CIC filters have a linear phase response and constant group delay.

Figure 19 shows the frequency response of a CIC filter used in this example with 5 stages and a decimation rate of 16, the frequency scale is relative to the pre-decimated sample frequency. The notches in the response occur at the new sample frequency and its harmonics, as with all digital filters the frequency response is symmetrical about Fs/2 and is repeated at all the harmonics of the sample frequency.

At the lower sample frequency, after decimation, the filter response is aliased about the new sample frequency and its harmonics. The main lobe, or passband, as shown in figure 20 is twice as wide as all the other lobes, this means that all the aliased spectra will all have a notch at the centre of the passband. If the frequency band of interest is close to the peak of the passband then all the aliased components will be highly attenuated as they are in the response notches

The difference signal frequency of 50KHz will on our response be at (0.05/100)Fs or 0.0005Fs where the droop in the passband response is less than 0.01dB. If the band width of the signals was wider then the droop in the frequency response may have to be considered. The first aliased component onto the required 50kHz signal from the spectrum centred on Fs/16 will be over 200dB's down.

The sum frequency component at 20MHz is on our filter response at (20/100)Fs or 0.2Fs, which lies between the notch at (3/16)Fs or 0.1875Fs and the notch at (4/16)Fs or 0.25Fs at a level of –120dB's relative to the pass band.

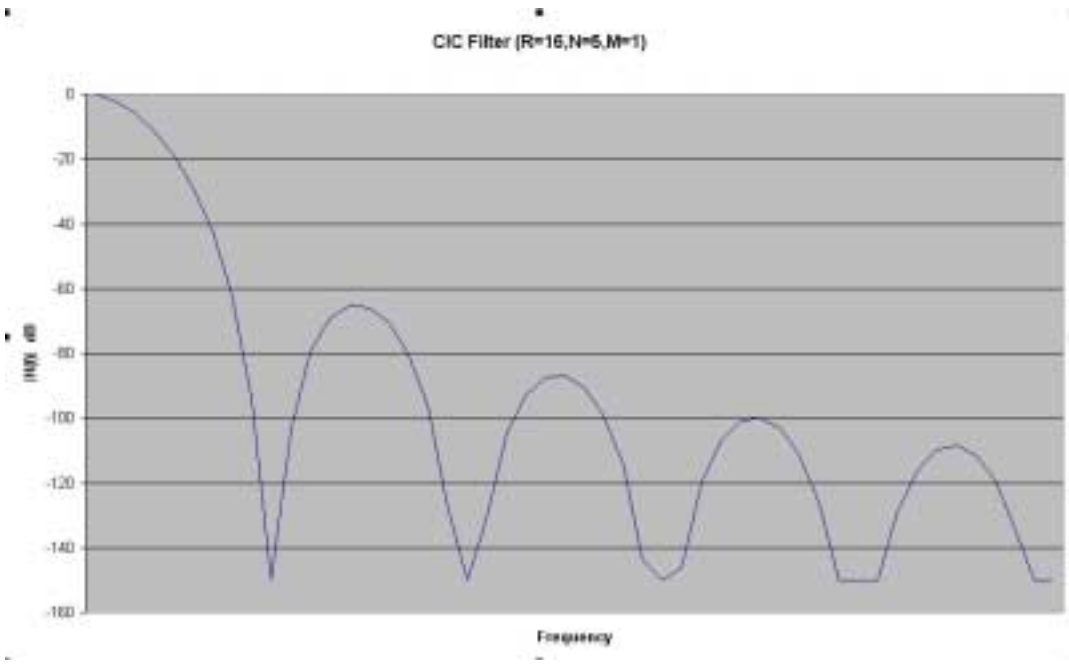
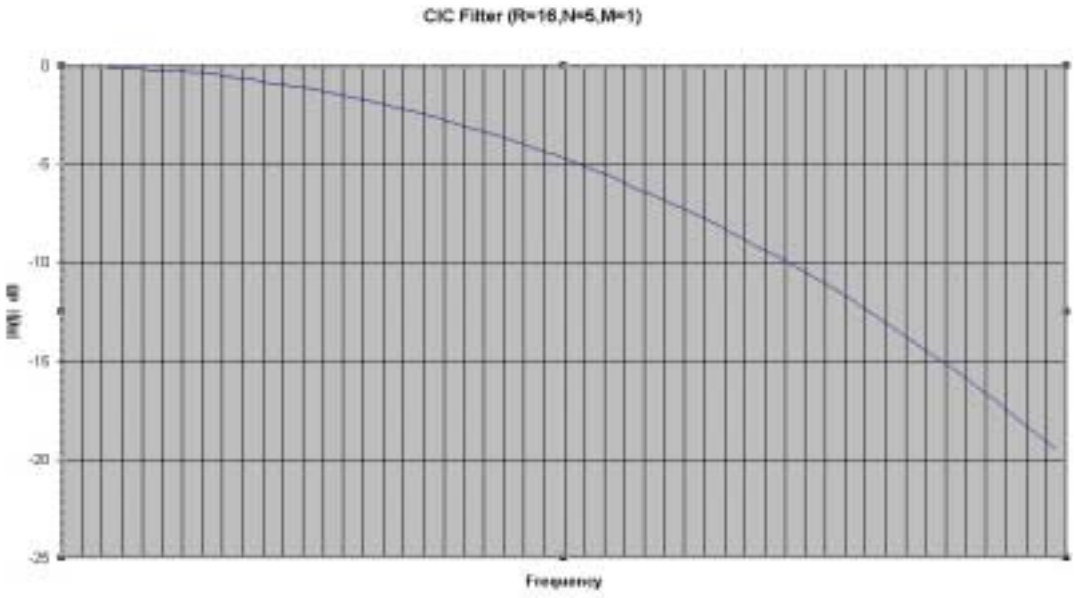


FIGURE 19 -- CIC Filter Frequency Responce

If the frequency band of interest was wider, say 0.5*(Fs/16), then the droop across the band would be 20 dB's as shown in figure 20, and the first alias would also be only 20dB's down at the edge of the band. This is why CIC filters are only useful for narrowband signals.

The pass band amplitude characteristic has a fairly gentle roll off, which is fine for narrow band signals where the droop over the band of interest is small, but for wider band signals the roll off in the pass band may cause problems, this can be overcome using a separate FIR filter with an amplitude characteristic to correct the droop. Figure 20 shows the frequency response of the pass band in more detail out to a frequency of 0.5Fs/16.



0 0.5Fs/16

FIGURE 20 - CIC Filter Passband Response.

The CIC filter with the above frequency response can be generated using the ‘CoreGen’

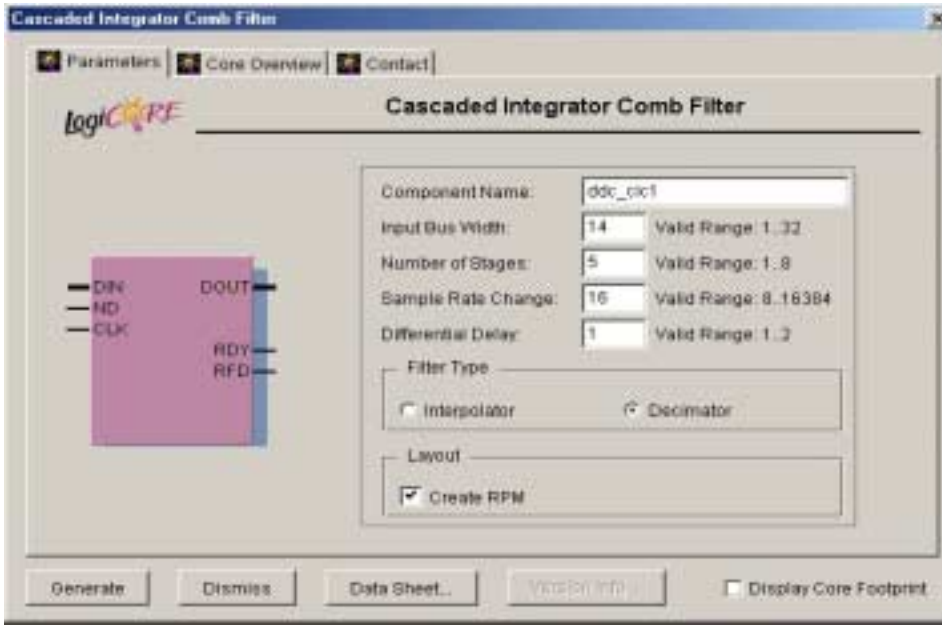


FIGURE 21 -- CoreGen CIC Filter GUI

Input bus width: The input to the CIC filter has been set to 14 bits.

Number of stages: The number of stages to give the frequency characteristics as shown in figure 13 is 5.

Sample rate change: The required decimation rate for this filter is 16.

Differential delay: The delay selected in the differentiator stages is 1.

The output bus width for the above CIC filter is 32 bits signed, in this example the resolution will be reduced to 14 bits before the first FIR filter stage.

First Filter Stage

The first filter stage is a 35 stage Finite Impulse Response (FIR) filter. The sample frequency from the CIC filter is (100/16) = 6.25MHz, so for a time window of 35 samples, the length of the filter, the minimum cut off frequency for a low pass filter will be approximately (6.25/35) = 178kHz. To get a reasonable cut off response the cut off frequency must be higher than this, in this example a cut off frequency of (0.048*6.25) = 300kHz has been selected, the frequency response is shown in figure 22. This design is from http://www.nauticom.net/www/jtaft site which uses Java design tools to design the filter on line. This is a symmetric filter so that only half of the coefficient values need to be stored. The filter coefficients are not in the form required for the 'CoreGen' filter block generator as it is expecting a 14 bit signed integer, the values in the file .\\src\filt1.coe have been normalised to have the maximum 14 bit signed value (+8191) for the largest coefficient.

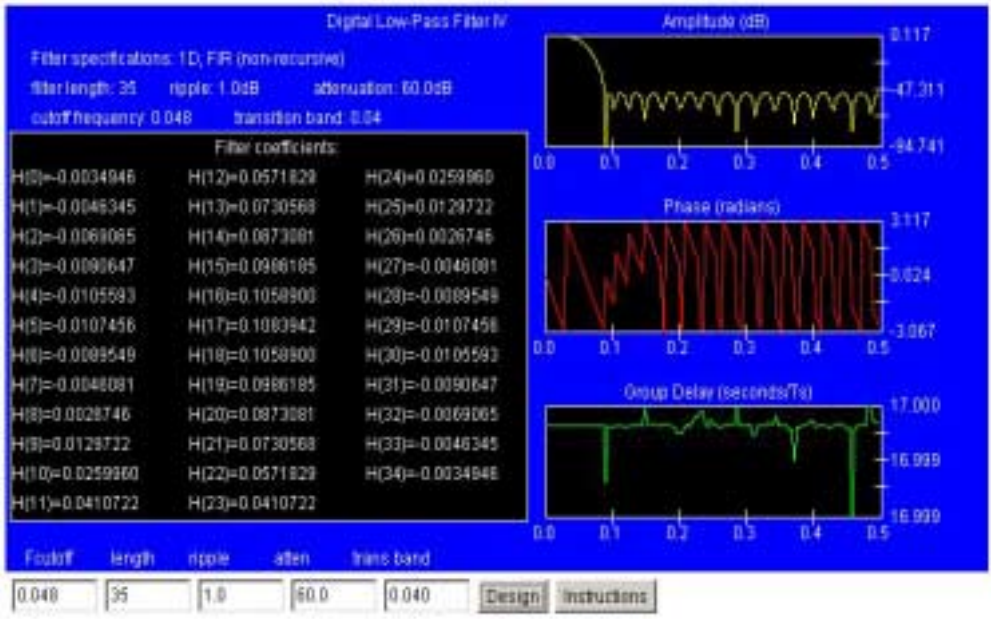


FIGURE 22 -- First FIR Filter Responce

The filter has a fairly gentle roll off, with a linear phase response. The stop band attenuation is approximately 50dBs'

The FIR filter with the above frequency response can be generated using the 'CoreGen'.

FIGURE 23 -- First FIR 'CoreGen' GUI Page 1

Filter Type: The filter type selected is decimation filter to further reduce the sample rate.

Filter Options: A decimation rate of 8 has been selected.

Structure: To give the filter response shown in figure 22 a symmetric filter with 35 taps has been selected. Selecting the 'Coefficients' button brings up the GUI shown in figure 24.

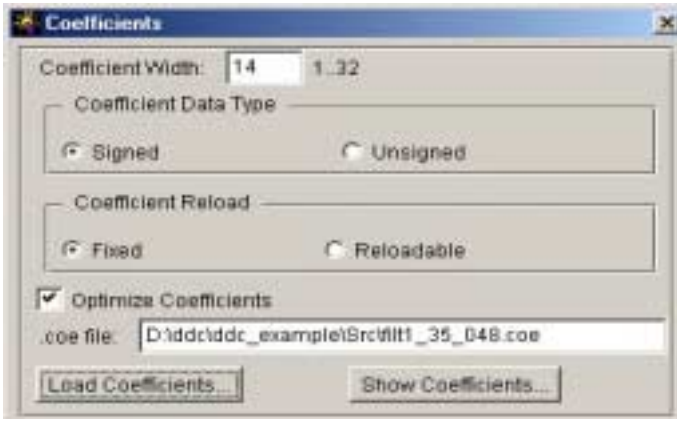


FIGURE 24 -- Coefficients 'CoreGen' GUI

Coefficient Width: 14 bits has been selected to match the 14 bit data input.

Coefficient Data Type: The coefficient data type is signed.

Coefficient Reload: For this filter the coefficients are fixed.

Optimize Coefficients: This has been selected.

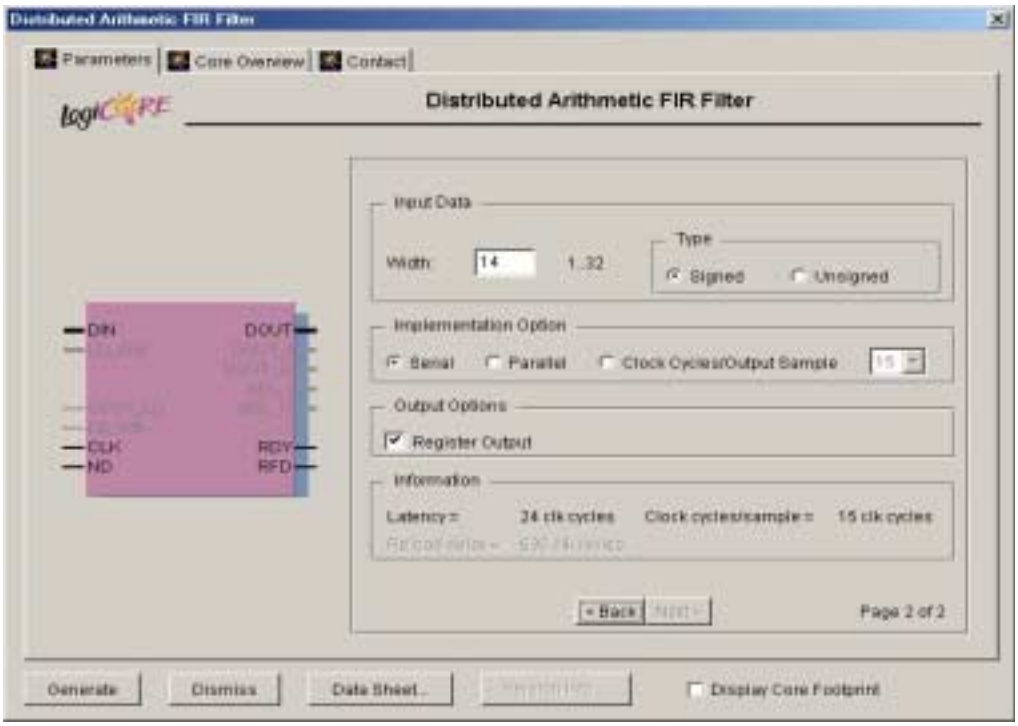


FIGURE 25 -- First FIR 'CoreGen' GUI Page 2

Input Data: 14 bit signed has been selected.

Implementation Option: The original sample frequency for this example is 100MHz, and this sample clock is used to clock the filter with New Data(ND) only being supplied every 16 Cycles because the data rate is decimated by 16 in the CIC filter stage. The Information window shows that for the serial implementation the filter requires 15 clock cycles/sample, which is less than the 16 cycles available, so the filter can have serial implementation. This saves on the amount of FPGA area required to implement filter.

Output Options: The output from the filter is registered. This is the safe option.

Second Filter Stage

The data sample rate at the input to the second filter has been reduced by the first filter to 781.25kSamples/second. A low pass filter with a cut off frequency of 100kHz is implemented for this final stage, which relative to the sample frequency is 0.128Fs. A 35 tap filter will now give a sharp filter response at 100kHz. The filter response is shown in figure 26 which is from http://www.nauticom.net/www/jtaft site which uses Java design tools to design the filter on line

FIGURE 26 -- Second FIR Filter Responce

The coefficients file for the ‘CoreGen’ filter design block will be normalised to give a maximum14 bit signed integer value for the maximum coefficient value H(7). The values in the file ..\src\filt2.coe have been normalised to have the maximum 14 bit signed value (+8191) for the largest coefficient.

FIGURE 27 -- Second FIR ‘CoreGen’ GUI Page 1

Filter Type: In Figure 27 the filter type is single rate, but this has been changed for the demo to a decimation rate of 4.

Filter Options: For this example the number of channels selected is 1. In a system design with both Phase and Quadraure channels it would be more efficient in terms of FPGA area to set the number of channels to 2 as common elements can then be shared.

Structure: This is a non symmetric filter with 35 taps. The coefficients button opens another GUI as shown in figure 26.

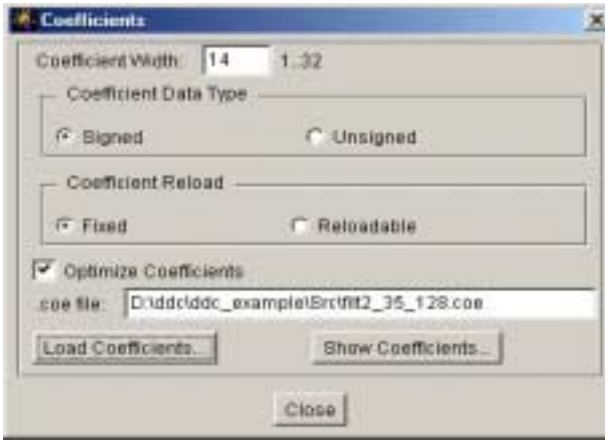


FIGURE 28 -- Coefficients 'CoreGen' GUI

Coefficient Width: This is set to 14 bits to match the input data.

Coefficient Data Type: The coefficients are signed.

Coefficient Reload: In this example the coefficients are fixed.

Optimise Coefficients: This is selected.

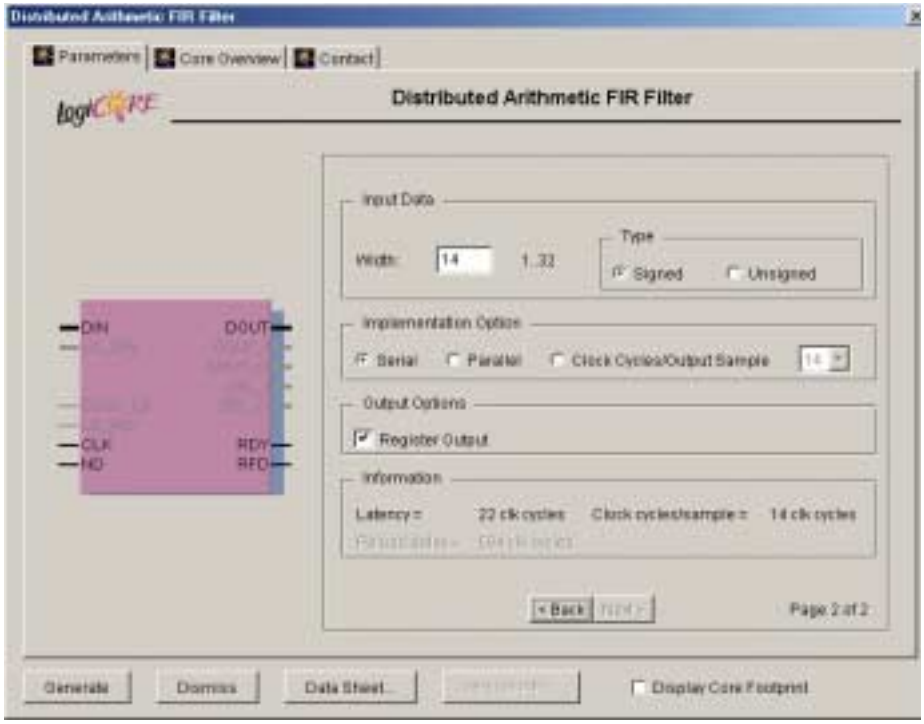


FIGURE 29 -- Second FIR 'CoreGen' GUI Page 2

Input Data: The input data from the first filter is 14 bit signed.

Implementation Option: The Filter is clocked at the100MHz original sample frequency but at this stage the sample frequency is down to 781.25kSamples/sec, which makes the serial implementation option possible as it only requires 14 clock cycles/sample. This gives savings in the required FPGA area over a parallel implementation.

Output Options: The register output option has been selected.

CoreGen DDC Block

This CoreGen block was not used in the example, but was generated to give a comparison of the amount of space required within the FPGA if the DDC was implemented using separate CoreGen blocks to make up the DDC relative to this CoreGen DDC block.

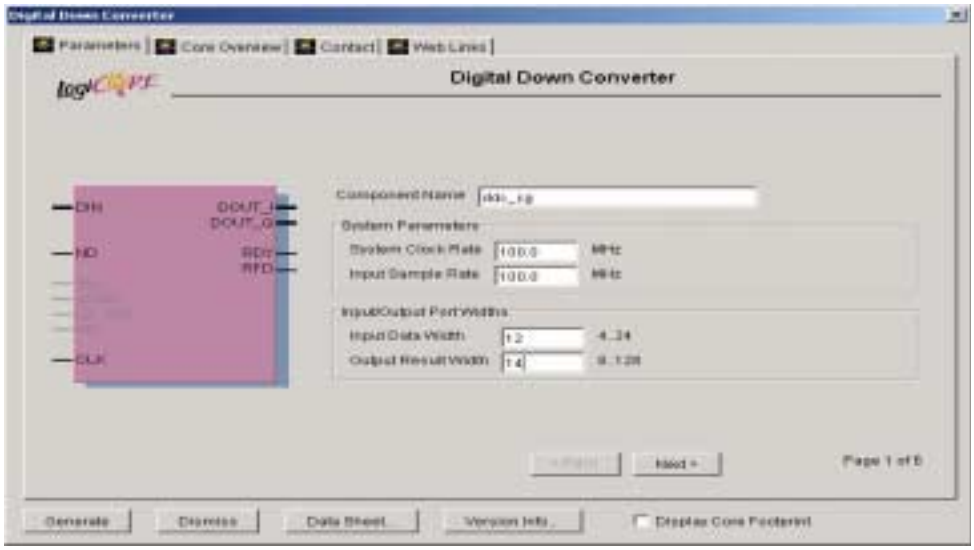


FIGURE 30 -- 'CoreGen' DDC GUI Page 1

System Clock Rate: The system clock frequency is 100MHz.

Input Sample Rate: The ADC converts at a rate of 100 Msamples per Second.

Input Data Width: The ADC is a 12 bit converter.

Output Data Width: To match the example, and also the DAC's, a 14 bit output has been selected.

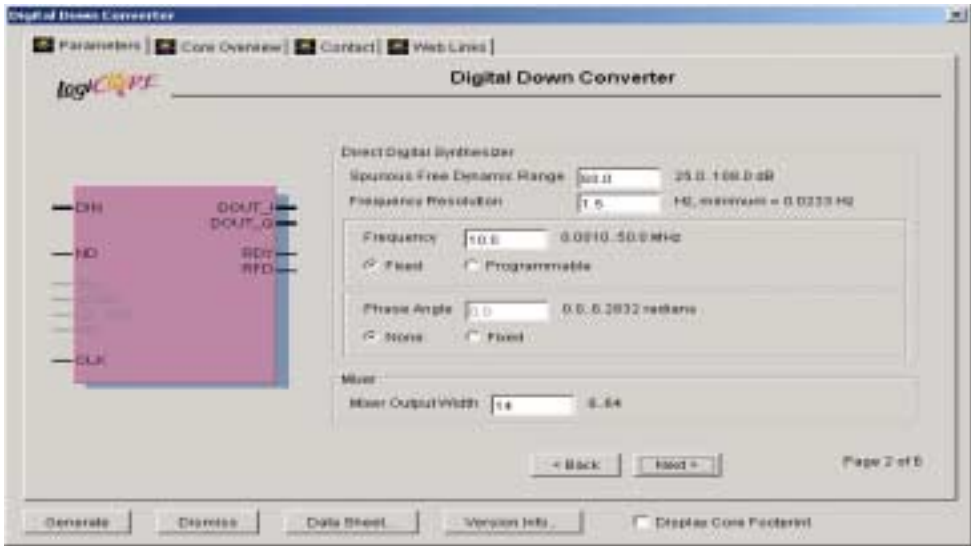


FIGURE 31 -- 'CoreGen' DDC GUI Page 2

DDS Spurious Free Dynamic Range: This has been set at 60 dB's to match the DDS in the example.

DDS Frequency Resolution: This has been set at 1.5Hz to match the DDS in the example.

DDS Frequency: This has been simplified from the DDS in the example to a fixed frequency of 10.0MHz.

DDS Phase: The phase offset has been set to None as in the DDS in the example.

Mixer Output Width: set at 14 Bits.

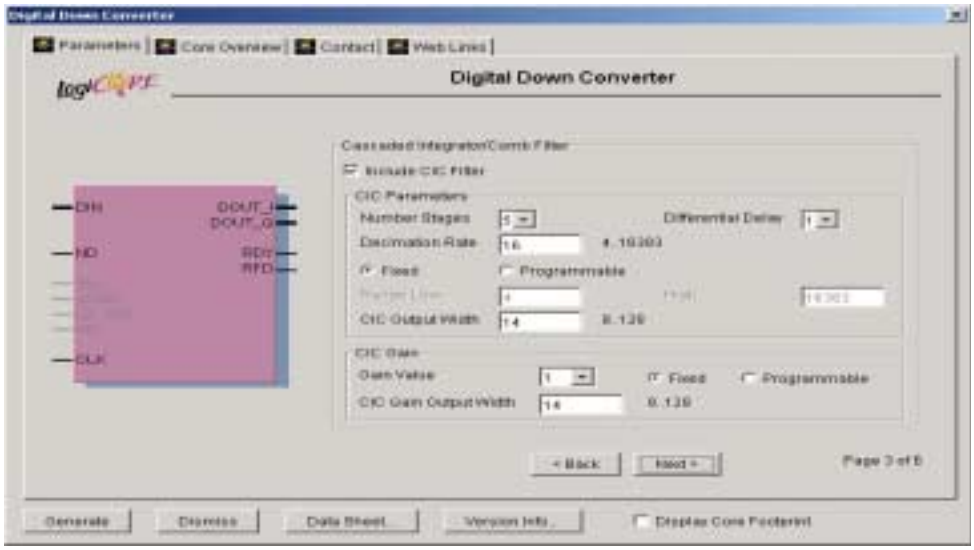


FIGURE 32 -- 'CoreGen' DDC GUI Page 3

CIC Filter: This has been set up, as far as possible, to match the CIC filter used in the example.

Number stages: Five stages selected.

Differential Delay: Set at 1.

Decimation Rate: A fixed decimation rate of 16 has been selected to match the Example.

CIC Output Width: 14 bits has been selected. In the example the output width is greater, 32 bits, of which 14 bits are selected to give a suitable gain.

CIC Gain: A fixed gain value of 1 has been selected.

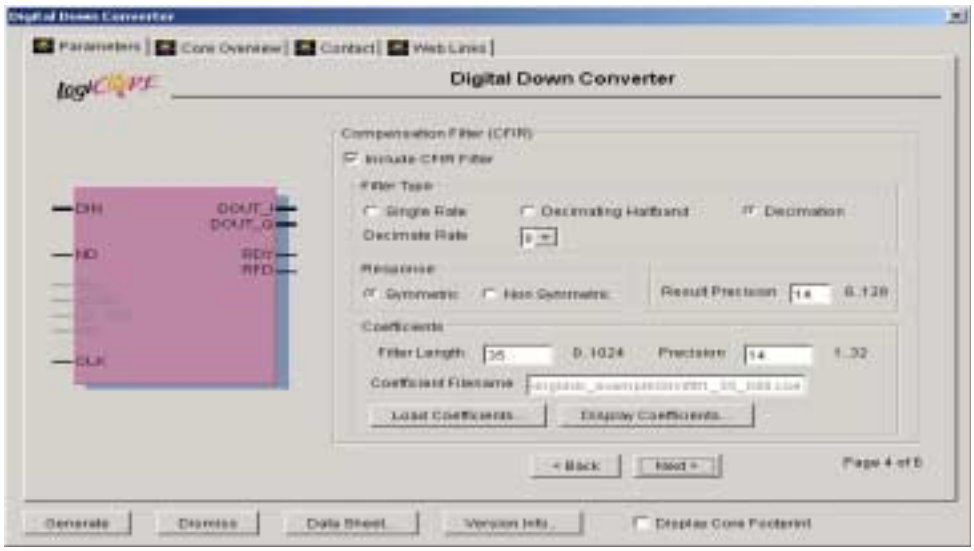


FIGURE 33 -- 'CoreGen' DDC GUI Page 4

Compensation Filter (CFIR): This is equivalent to the first filter stage in the example, and has been set up to be the same.

FilterType: Decimation filter with a decimation rate of eight.

Response: This is a symmetric filter

Result Precision: 14bits.

Coefficients: As with the filter in the example it has 35 taps and uses the same coefficient file as the first filter in the example.

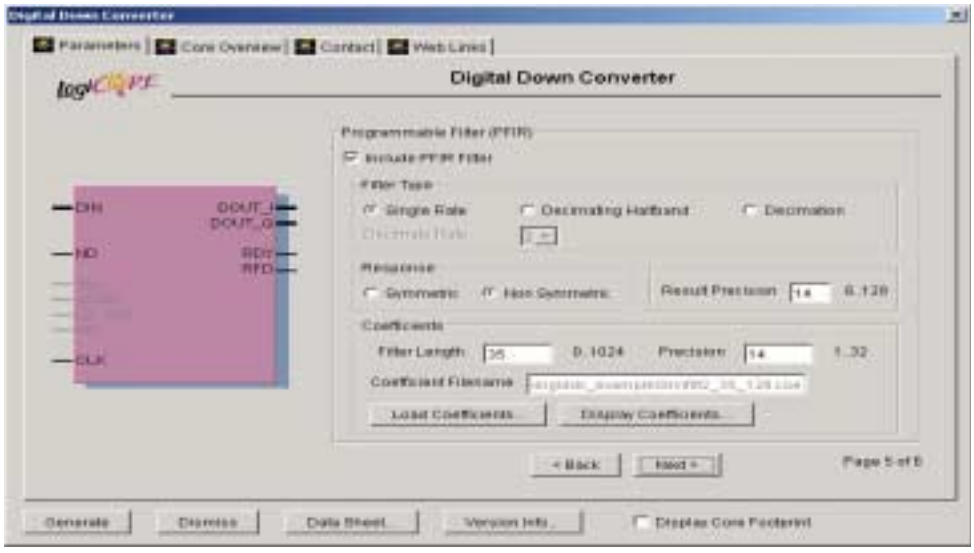


FIGURE 34 -- 'CoreGen' DDC GUI Page 5

Programmable Filter: This is equivalent to the second filter in the example, and has been set up to be the same.

Filter Type: This is a single rate filter.

Response: The filter coefficients are non symmetric.

Result Precision: 14 bits.

Coefficients: This filter is the same length as the second filter in the example and is loaded with the same coefficient file.

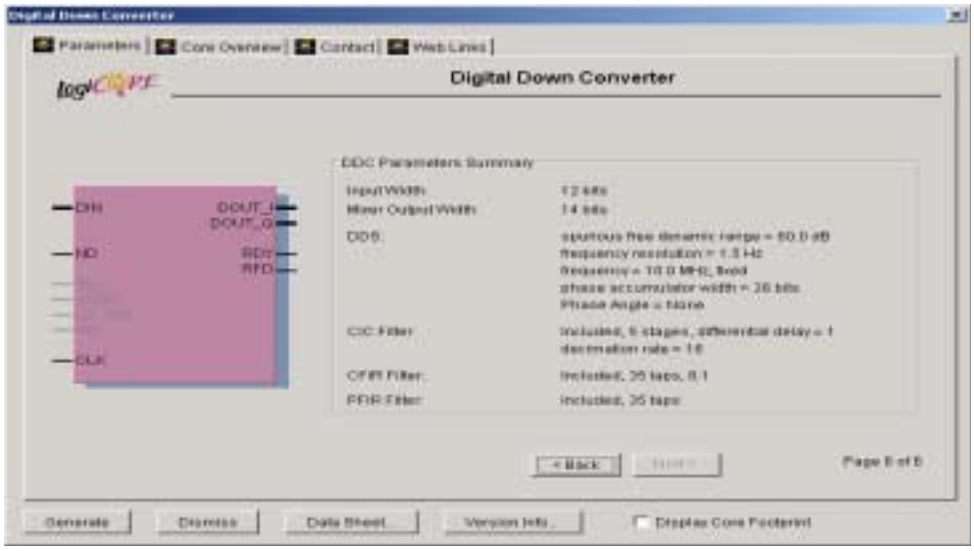


FIGURE 35-- 'CoreGen' DDC GUI Page 6

This page summarises the inputs on the previous five pages.

Comparison of FPGA Areas Used

To give an approximate comparison of the FPGA area used with the DDC example the device utilization has been noted for three different build situations. The first is just Example 2 off the CD without any changes, this gives the basic device utilization for the HUNT ENGINEERING interfaces.

Example 2 Device Utilization

MULT18x18	0 of 40	0%
RAM16's	8 of 40	20%
Slices	427 of 5120	8%

The second situation is for the DDC example, here most of the HUNT ENGINEERING interfaces are still used with the exception of the HERON Write as no data is written to the IO2V2 over the HERON interface. This is why the number of Block rams has reduced.

DDC Example Device Utilization

MULT18x18	2 of 40	5%
RAM16's	4 of 40	10%
Slices	2163 of 5120	42%

The Last situation is when the CoreGen DDC was implemented instead of the individual blocks as in the DDC Example. This is a slightly simpler implementation, as none of the signal multiplexing used in the DDC Example is present.

CoreGen DDC Device Utilization

MULT18x18	2 of 40	5%
RAM16's	4 of 40	20%
Slices	1811 of 5120	35%

This exercise has shown that the DDC Example is using about 1736 slices relative to the CoreGen simplified equivalent of 1384 slices. So there is a saving in logic when you use the Coregen block.

The choice of how to implement a DDC will depend on the application. If the FPGA area used is not a problem but development time is a problem then use the CoreGen DDC, if it will do the job for you. If the CoreGen DDC does not do what is required then the next step back is to use individual CoreGen blocks as in the DDC Example. If the application requires the DDC to be tailored to fit the FPGA to optimise on the speed and/or silicon area used then it will require a specific customised design, which will require a longer development time, but as shown by this example – not much longer.