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. The IO5V has an A/D sample rate of 210MSamples per Second, which makes it suitable for digital radio applications that would require Digital Down Conversion.
This DDC demo/Framework is for the IO5V FPGA module developed by HUNT ENGINEERING to show how a RF signal can be down converted. The RF signal used 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-IO5V.

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

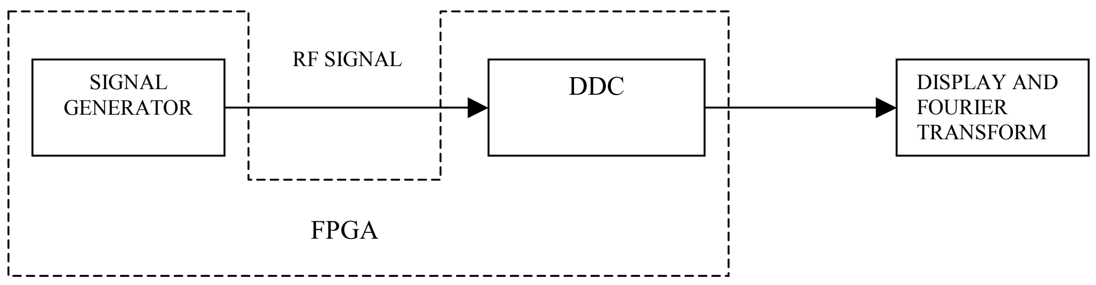


Figure 1 – Demonstration Concept

The demonstration shows how an RF signal, generated within the FPGA and output via DAC 'A' on the IO5V, can be Digitally Down Converted to baseband. The output of the DDC is then Fourier transformed by the host software and compared with the same size Fourier transform of the RF signal.

This allows the demo to be run by simply connecting the DAC 'A' output of the HERON-IO5V to the ADC 'A+' input – 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 IO5V. The second section is the DDC, from the HUNT ENGINEERING DDC example, in the FPGA which takes the signal from ADC 'A' on the IO5V, sampled at 210Msamples 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 blocks data from the HERON interface and displays them 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 420kHz, while after the DDC the resolution is now 410Hz and the structure of the RF signal can now be seen.

The main building blocks that make up the DDC example are Xilinx CoreGen blocks.

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 IO5V to produce an analogue output signal. For this demonstration the DAC-A output should be connected to the ADC 'A+' input on the IO5V as this is the input used for the DDC.

The DAC used on the IO5V has the capability of interpolation to increase the update rate of the DAC output, so producing a smoother waveform. This demonstration uses an interpolation factor of x4 to increase the DAC output update rate from the input data rate to the DAC of 100MSPS to 400MSPS. Control of the DAC's extra functions is achieved over the Serial Peripheral Interface (SPI) bus, more information on the SPI can be found in the document "Using the Hardware Interface Layer in your FPGA Design" to be found on the CD.

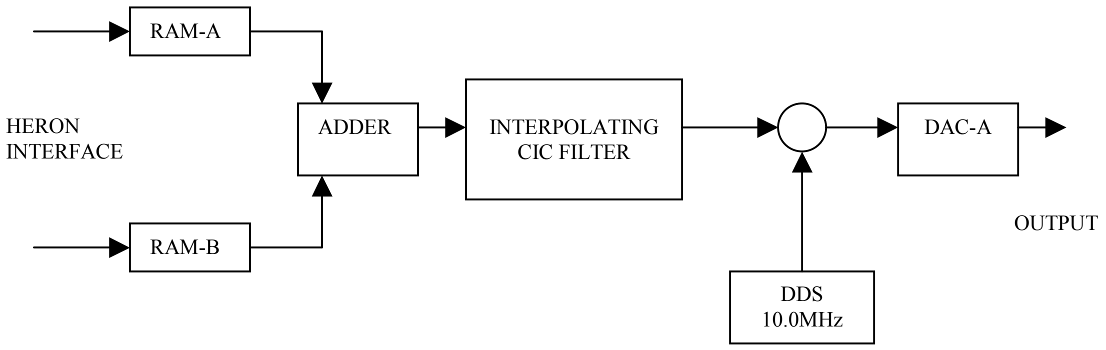


Figure 2 – Signal Generator

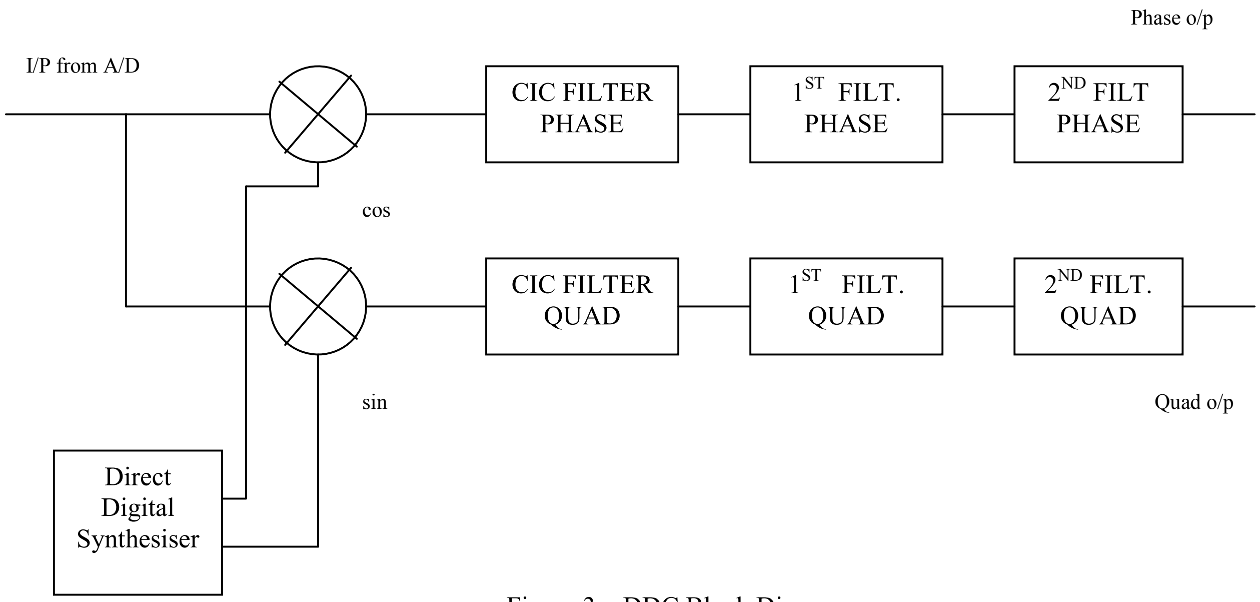


Figure 3 – DDC Block Diagram

DDC

The block diagram in figure 3 shows the basic structure of the DDC implemented in this example. The analogue input signal to the ADC is the AM and FM signal modulated up to 10MHz in the signal generator section of this demonstration.

The ADC samples the 10MHz analogue signal at 210MSPS. The object of the DDC is to shift the frequency band of interest, which is at 10MHz , down to baseband. The first step is to mix the digitised input signal with 10MHz to produce sum and difference frequency components at 20MHz and baseband respectively. In this example the input signal is mixed with both a 10MHz sine and cosine to produce a quadrature and in phase signal respectively.

The Direct Digital Synthesiser (DDS) generates a 10MHz digitised sine and cosine waves to mix, or multiply, the digitised input signal. The same sample frequency for the ADC and for the output of the DDS has to be the same and is 210MSPS in this example.

The low pass filters that follow the mixers are there to remove the sum frequency component and also to ensure that the bandwidth of the signal is suitable for the new sample frequencies after decimation. In this example the sample frequency has been kept to the Nyquist rate or better as the signal progresses through the filter stages, even though the sample rate could be halved because of the in phase and quadrature sampling. This is because only the in phase component of the DDC output signal is transferred to the host program to be displayed and FFT'd.

ADC Interface

The ADC used on the IO5V can sample at frequencies up to 210MSPS, and in this DDC example the sample frequency used is 210MSPS. The digital interface between the ADC and the FPGA is split into two 12Bit busses running at half the sample frequency. The ADC also generates a clock signal, closely related to the timing of the output data, for latching the data into the FPGA. At these high sample frequencies the timing of the data being latched into the FPGA is critical, so the ADC to FPGA interface is taken care of in the Hardware Interface Layer (HIL) of the VHDL supplied with the IO5V. The ADC data is presented for the user at the 'user_ap' interface in the VHDL still as two 12bit busses together with a data clock at half the sample clock frequency. The data busses are 'ADC_A_A' and 'ADC_A_B', and the clock is 'ADC_DCO_A', where if bus 'A' is sample N then bus 'B' is sample N+1.

This DDC example needs the data to be multiplexed onto a single bus running at the full sample clock frequency. A Digital Clock Manager (DCM) is used in the FPGA to double the data clock frequency while retaining the timing relationship to the data busses. The ADC sample clock should **NOT** be used to clock the ADC data. This new data clock is now at the sample frequency, and has the correct timing relationship to the data busses to allow the sample data to be multiplexed onto a single bus. This new clock should be used for clocking all the data that requires the full sample rate frequency.

DDS

The DDS generates the digitised 10MHz sine and cosine signals required to mix with the ADC data giving both sum and difference frequencies. The cosine signal is used to generate the in Phase component, and the sine generates the quadrature component of the complex signal.

The DDS has to be clocked at the sample frequency so that sine and cosine data are presented to the mixers at the same rate as the multiplexed ADC data. The digitised 10MHz sine and cosine waves have a sample frequency of 210MHz so there are 21 samples per cycle.

MIXERS

The mixers, or multipliers, used in this example have to run at the full sample frequency of 210MHz. To achieve these speeds the hardware multipliers in the Virtex II are used in enhanced mode. More information on using the multipliers in the Virtex II can be found in the HUNT ENGINEERING application note "Using Virtex II Multipliers" that can be found on the CD.

The output of the mixer will contain both the sum and difference frequency components at a sample frequency of 210MHz.

CIC FILTERS

CIC filters are low pass filters used to realise sample rate changes, especially in systems that have large excess sample rates. At the ouput of the DDC mixers the difference frequency band of interest is only in the order of 100kHz while the sample frequency is still 210MHz.

CIC filters also have the advantage that they do not require multipliers, consisting only of adders, subtractors and registers. More information on CIC filters can be found in the Xilinx CoreGen CIC Filter data sheet.

In this example the CIC filter is used to decimate the sample frequency by a factor of 32, this reduces the sample rate to (210/32) = 6.5625MSPS. The output of the CIC filter is still clocked at the full sample frequency of 210MHz. Now that the sample rate has been reduced from 210MSPS a lower clock frequency can be used which will relax the required timing constraints. The change in clock frequency is achieved using a fifo. At the input to the fifo data is clocked in using the 210MHz clock at 6.5625MSPS and at the output the data is clocked out using a 100MHz clock at 6.5625MSPS. 100MHz was selected as it is the frequency of the User Oscillator on the IO5V and it gives over (100/6.5625) 15 clock cycles for every sample value

FILT1 and FILT2

These filters are standard Finite Impulse Response (FIR) filters both with 35 taps that are used to filter and to further decimate the sample frequency.

The low pass frequency response of FILT1 has a cut off frequency at (0.048x6.5625) = 315kHz and the sample frequency at the output is decimated by 8 giving (6.5625/8) = 820kSPS.

The low pass frequency response of FILT2 has a cut off frequency at (0.128x820) = 105kHz and a sample frequency at the output is decimated by 4 giving (820/4) = 205kSPS.

INTERFACE TO THE HOST

For this demonstration example blocks of 500 32bit words are transferred over the HERON interface when requested by the HOST software. The HERON interface in the FPGA is part of the Hardware Interface Layer. The example uses fifo's to grab a block of 500 consecutive samples of data. Both the phase and quadrature outputs from the DDC are available in the VHDL.

To demonstrate the effect of using a DDC the blocks of 500 32bit words are not phase and quadrature data from the output DDC, instead 500 consecutive samples from the ADC output are put in the least significant 16bits, and 500 consecutive samples of phase data from the output of the DDC are placed in the most significant 16bits.

ADC SAMPLE CLOCK

Rather than having the requirement for a RF signal generator as a source for the ADC sample clock, this example uses a DCM to generate a 210MHz clock inside the FPGA from the 100MHz User Oscillator clock. The only extra hardware required to run the example is a cable to link DAC 'A' output to ADC 'A+' input.

FILTERING AND DECIMATION.

The sample frequency required to retain all the information in a purely real signal, no phase and quadrature components, was defined by Nyquist to be twice the signal bandwidth. In a DDC the sample frequencies can be much higher than required for the bandwidth of the signal of interest, but before the sample frequency is decimated unwanted frequency components must be removed by filtering. Decimating the sample frequency makes signal processing simpler to implement as the timing constraints can be relaxed, it can also lead to a reduction in the amount of FPGA area required to perform the processing as functions can be performed serially rather than in parallel.

At the input to the filter/decimator there could well be signals outside the frequency band of interest, but because of the higher sample frequency do not interfere with the required signals. If the sample frequency was just decimated without filtering the unwanted signals would be overlaid onto the required signal frequency band. This is why the signal is filtered first, to remove any frequency components that could be overlaid onto the required signal, then the sample frequency is decimated.

In this example the Host software takes just the phase component for an FFT, so the sample frequency at the output of the filter/decimators has been kept to the Nyquist rate. If the full complex signal was used the sample frequency could be halved.

Host Software

We supply a pre-compiled Windows program for the PC, which can be used to communicate with the HERON-IO5V, 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 IO5V required for the demonstration should be fitted to a carrier board such as a HEPC9. There should be a connection between the DAC 'A' output 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-IO5V 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.

DDC Demonstration Bitstreams

The DDC demonstration for the HERON-IO5V is supplied on the HUNT ENGINEERING CD, along with bit streams that can be loaded directly onto the HERON-IO5V. The bit streams can be found under 'fpga\io5v1\ddc_example' and the name of the bitstream file reflects the FPGA part number. I.e. 2v1500fg676 is for the HERON-IO5V with a Xilinx XC2V1500 FPGA when fitted to a HEPC9 carrier board.

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-IO5V 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.

Other User LEDS are also used in the DDC example for the IO5V:

LED(3) =>	DCM used for x2 of the ADC data clock is in lock.
LED(2) =>	SPI transfer to set up the DAC is complete.
LED(1) =>	DCM used for generating the 210MHz sample clock is locked.
LED(0) =>	not used.

LED's 3, 2 and 1 should be lit after the bit stream has been loaded.

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 IO5V.

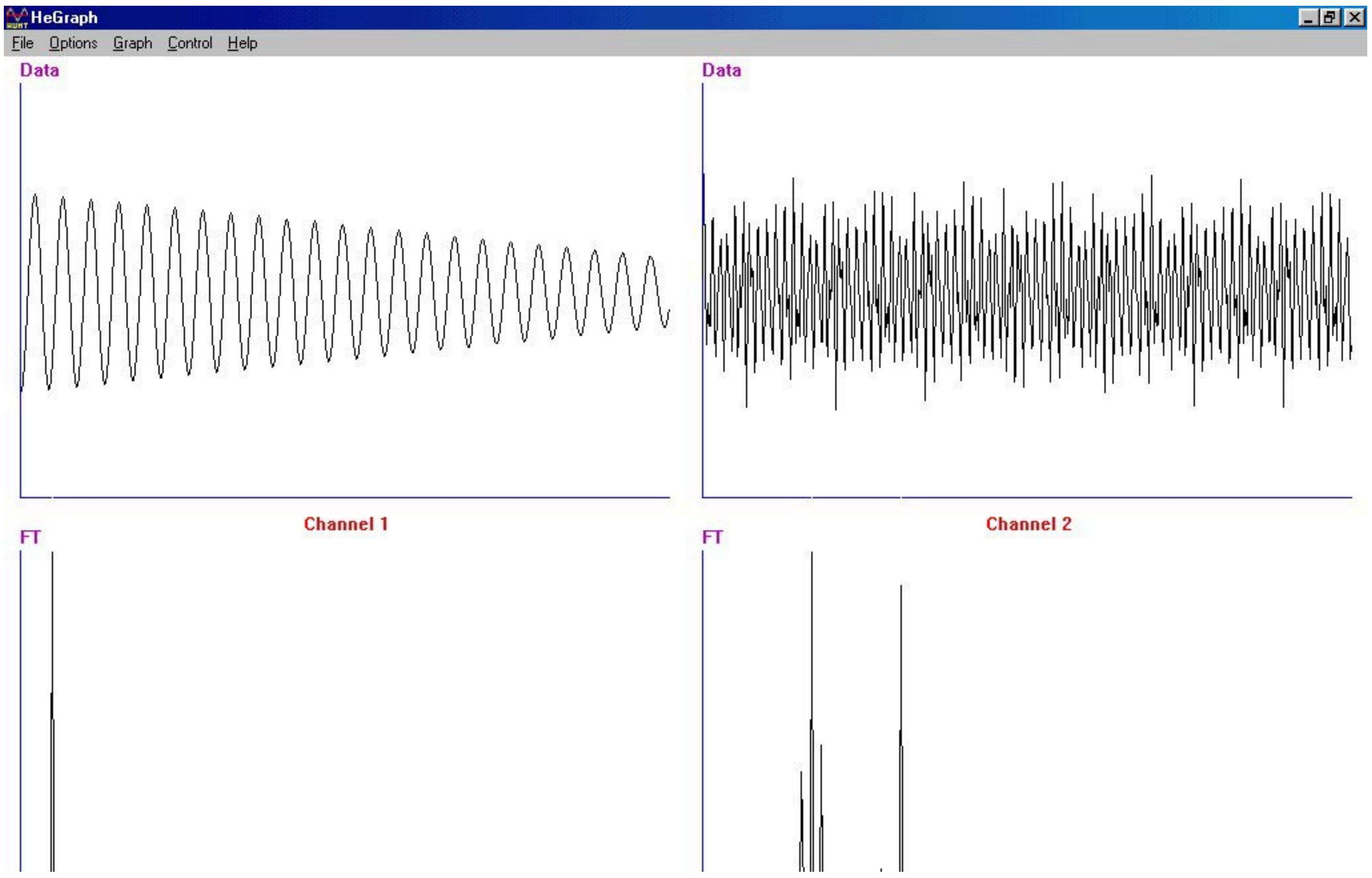


Figure 4 – Demonstration Output Display

The data is displayed both as a time sequences of samples in the upper half of the window and as their Fourier Transform in the lower half. The left hand pair show the ‘RF Signal’ which is from the output of ADC ‘A’, i.e. the input to the DDC, and the right hand pair show the ‘DDC Output’. A typical display is shown in figure 4.

Channel 1, on the left of the output shows 500 samples of the RF Signal digitised by ADC ‘A’ at 210Msamples per Second , 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 105MHz. The output for frequencies between Fs/2 and Fs are a mirror image of those below Fs/2. The RF signal is at 10MHz as shown in the FFT output. The envelope of the time sequence is slowly changing, relative to the 500 sample window, because of the AM and FM signals but this can not be resolved in the FFT output.

With a sample frequency of 210MHz and a 500 point Fourier transform each resolution cell at the output of the transform is (210MHz / 500) = 420kHz wide, but both of the AM and FM signals are in a band of about 100kHz at 10MHz and so can not be individually resolved. To resolve down to a frequency of 1kHz with this sample frequency would require a 210000 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 205kHz, this output is shown on the right hand side of figure 4. The same 500 point Fourier transform now has a resolution of (205kHz / 500) = 410Hz and now the structure of our composite signal is visable. The time sequence now contains many cycles of the composite signal, a 61kHz sine wave would have (500/205 * 61) =148.8 cycles in the 500 point time window.

In the FFT of the DDC output 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 FFT used in the host software automatically adjusts the output amplitude to fit the display.

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.

The input to the DDC is ADC ‘A+’ but this does not have to be connected to to the DAC output, it could be connected to a RF signal generator with a frequency in the range 10.0MHz to 10.1MHz at a level of 0dBm. The DDC will shift the signal to a frequency band between DC and 105kHz.

DDC demonstration Specification

Signal Generator Specification

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
Frequency:	9.999990MHz (not quite 10MHz)
Data width:	26 bits
Phase angle width:	10 bits
Phase increment(const):	0x666660
Accumulator latency:	1
Accumulator width:	26 bits

Note: The DDS output frequency is made not quite 10MHz on purpose. If the DDS in the signal generator and in the DDC are the same frequency and locked to the same clock the relative levels in the phase and quadrature components will remain fixed after reset. Slightly different frequencies allows the relative levels to change slowly.

DAC

Output Update Rate:	400MHz
Digital Input:	16 bit @100MSPS
Interpolation Rate:	x4
Analogue Output:	+/-0.374 Volt max into 50Ohms

Heron Interface

Wave shape words written **to** the IO5V:

MS half word	LS half word
0x ‘0001’(for Channel A RAM)	(16 bit value)
0x ‘0004’(for Channel B RAM)	(16 bit value)

Data **from** the IO5V:

MS half word	LS half word
(16 bit of DDC Phase output)	(12 bit of RF signal) & ‘0000’

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 Specification

DDS210 -- generating 10MHz sine and cosine for mixers

O/P Frequency:	10MHz (from 210MHz clock)
O/P Width:	12bits both sine and cosine
Implementation:	Block ROM, pipelined
Data Width:	26bits
Phase angle width:	10bits
Accumulator width:	26bits
Phase increment (const)	‘30C30C’ Hex
Noise:	none

CIC210 – first stage of filtering and decimation after mixers

Sample Rate Change:	x32 Decimation
I/P Bus Width:	16bits
Number of stages:	4
Number of channel:	1
Differential Delay:	1

[Latency 1 cycle, Result width 35bits]

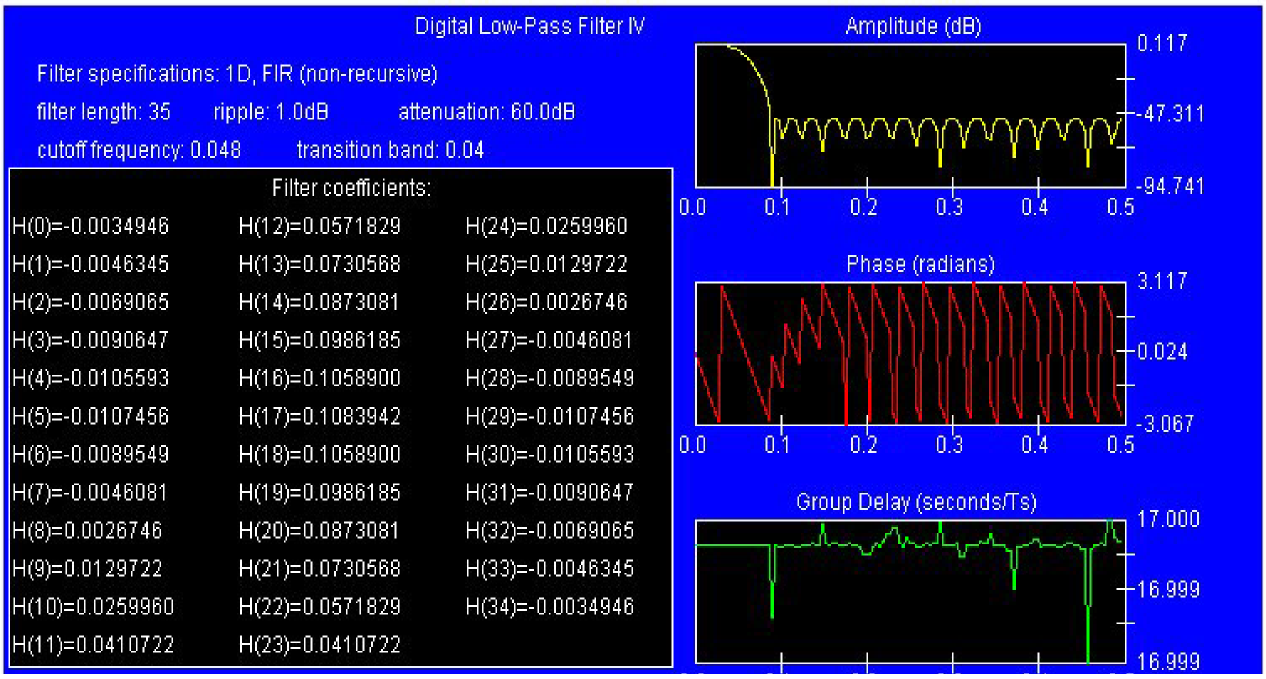
FIFO511X32

Memory type:	Block memory
I/P Width:	32bits
Length:	511
Flags:	Almost Full, Almost Empty
	Full, Empty

FILT1 – first FIR filter

Sample Rate Change:	x8 Decimation
Number of Taps:	35
Coefficients:	symmetric
	14bit signed
	file: ..\\scr\filt1_35_048.coe
	optimised
I/P Data:	14bit Signed
Clock cycles/O/P sample:	3
O/P:	Registered
Option:	Reset pin

[Latency = 15cycles]



Digital Low-Pass Filter IV

Filter specifications: 1D, FIR (non-recursive)

filter length: 35 ripple: 1.0dB attenuation: 60.0dB

cutoff frequency: 0.048 transition band: 0.04

Filter coefficients:

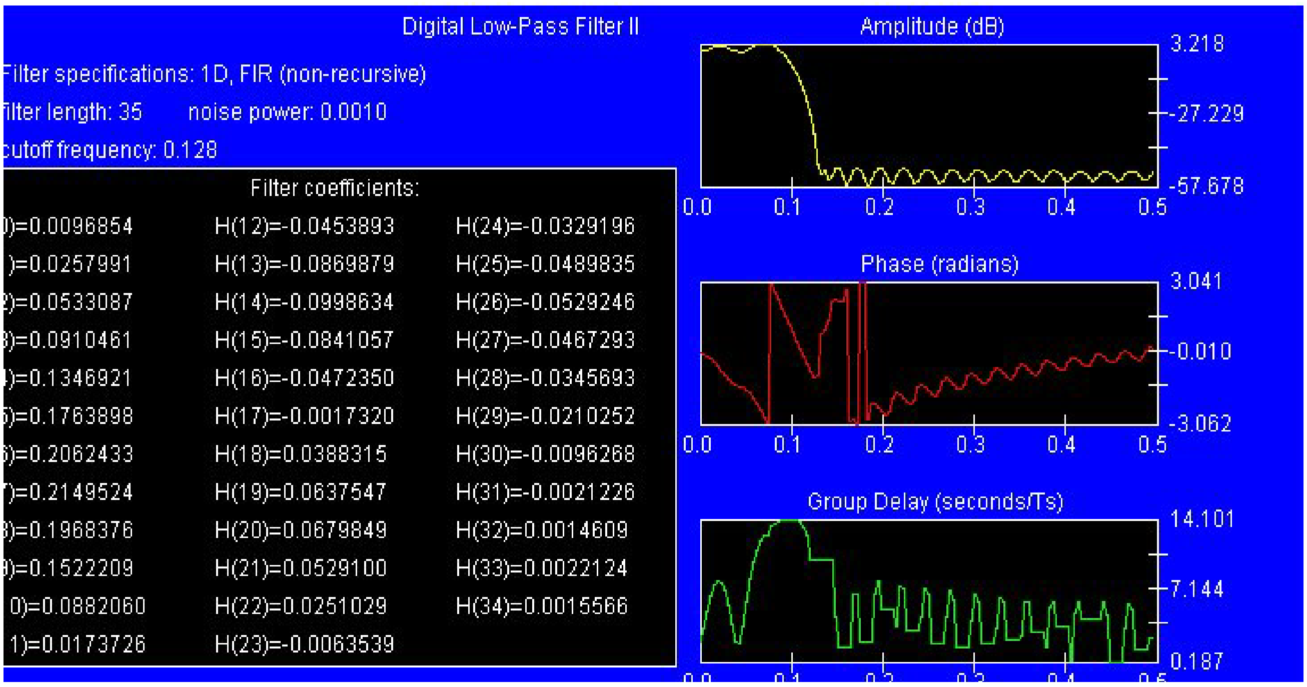
H(0)=-0.0034946	H(12)=0.0571829	H(24)=0.0259960
H(1)=-0.0046345	H(13)=0.0730568	H(25)=0.0129722
H(2)=-0.0069065	H(14)=0.0873081	H(26)=0.0026746
H(3)=-0.0090647	H(15)=0.0986185	H(27)=-0.0046081
H(4)=-0.0105593	H(16)=0.1058900	H(28)=-0.0089549
H(5)=-0.0107456	H(17)=0.1083942	H(29)=-0.0107456
H(6)=-0.0089549	H(18)=0.1058900	H(30)=-0.0105593
H(7)=-0.0046081	H(19)=0.0986185	H(31)=-0.0090647
H(8)=0.0026746	H(20)=0.0873081	H(32)=-0.0069065
H(9)=0.0129722	H(21)=0.0730568	H(33)=-0.0046345
H(10)=0.0259960	H(22)=0.0571829	H(34)=-0.0034946
H(11)=0.0410722	H(23)=0.0410722	

First FIR Filter Responce

FILT2 – second FIR filter

Sample Rate Change:	x4 Decimation
Number of Taps:	35
Coefficients:	non symmetric
	14bit signed
	file: ..\\src\filt2_35_128.coe
	optimised
I/P Data:	14bit Signed
Clock cycles/O/P sample:	4
O/P:	Registered
Option:	Reset pin

[Latency = 12cycles]



Digital Low-Pass Filter II

ilter specifications: 1D, FIR (non-recursive)

ilter length: 35 noise power: 0.0010

utoff frequency: 0.128

Filter coefficients:

```
)=0.0096854      H(12)=-0.0453893      H(24)=-0.0329196
)=0.0257991      H(13)=-0.0869879      H(25)=-0.0489835
)=0.0533087      H(14)=-0.0998634      H(26)=-0.0529246
)=0.0910461      H(15)=-0.0841057      H(27)=-0.0467293
)=0.1346921      H(16)=-0.0472350      H(28)=-0.0345693
)=0.1763898      H(17)=-0.0017320      H(29)=-0.0210252
)=0.2062433      H(18)=0.0388315       H(30)=-0.0096268
)=0.2149524      H(19)=0.0637547       H(31)=-0.0021226
)=0.1968376      H(20)=0.0679849       H(32)=0.0014609
)=0.1522209      H(21)=0.0529100       H(33)=0.0022124
0)=0.0882060     H(22)=0.0251029       H(34)=0.0015566
1)=0.0173726     H(23)=-0.0063539
```

Second FIR Filter Responce

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. The first is just Example 2 on the CD without any changes, this gives the basic device utilization for the HUNT ENGINEERING interfaces.

Example 2 Device Utilization

	XC2V1500-6FG676		:	XC2V3000-6FG676	
MULT18x18	0 of 48	0%	:	0 of 96	0%
RAM16's	7 of 48	14%	:	7 of 96	7%
Slices	563 of 7680	7%	:	563 of 14336	3%

The second situation is for the DDC example, here most of the HUNT ENGINEERING interfaces are still used.

DDC Example Device Utilization

	XC2V1500-6FG676		:	XC2V3000-6FG676	
MULT18x18	3 of 48	6%	:	3 of 96	3%
RAM16's	10 of 48	20%	:	10 of 96	10%
Slices	5697 of 7680	74%	:	5697 of14336	39%

This exercise has shown that the DDC Example is using about 5135 slices.

The choice of how to implement a DDC will depend on the application. 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.