# High Performance, Low Area, Interpolator Design for the XC6200

XAPP 081 May 7, 1997 (Version 1.0)

Application Note by Beth Cowie

***Summary***

An Interpolator FIR filter design for the XC6200 is discussed.

***Xilinx Family***

XC6200

***Demonstrates***

The suitability of the XC6200 for CIC Filters and general DSP.

The Hardware/Software co-design process with the XC6200 PCI Board.

## Overview

This note describes the implementation of a digital linear-phase, finite impulse response (FIR) filter for Interpolation (sampling rate increase) on the XC6200. Interpolation and Decimation (sampling rate decrease) filters have a broad range of applications in DSP. The described algorithm is used in digital radios, cell phones, base stations, etc. Interpolation also has a vast range of applications in Image Processing. The filter is based on the paper, ‘An Economical Class of Digital Filters for Decimation and Interpolation’ (Ref. 1). The filters require no multipliers and consist solely of adders with feedback and feedforward registers. The output sampling rate of the filters may be as high as 50MHz and the design therefore requires a high degree of pipelining. The XC6200 series, with its high register count architecture, is well suited to the implementation of these high clock rate filters. A complete description of the benefits of the XC6200 architecture is given in the section: Why use the XC6200? The note also describes the co-design process involved in verifying the design with the XC6200DS.

The sections covered in this note are:

## Filter Description

A complete description of the Interpolation Filters described in this note can be found in Ref. 1. These filters require no multipliers and are known as cascaded integrator-comb (CIC) filters. Their structure consists of a comb section, operating at a low sampling rate, combined with an integrator section, operating at a high sampling rate. Figure 1 shows the basic structure of the CIC interpolation filter. Figure 2 shows the basic building blocks which make up the design.

The comb section operates at the low sampling rate $f_S/R$, where $f_S$ is the high sampling rate and R is the ratio of low sampling rate to high sampling rate. The comb section consists of N comb stages with a differential delay of M samples per stage. The differential delay is a filter design parameter used to control the filter’s frequency response. The differential delay usually takes a value of 1 or 2.

The system function for a single comb stage with reference to the high sampling rate is:

$$H_C(z) = 1 - z^{-RM}$$

**Eqn 1.**

The integrator section of a CIC filter consists of N ideal digital integrator stages operating at the high sample rate, $f_S$. Each stage is implemented as a one-pole filter with a unit feedback coefficient. The system function for a single integrator is:

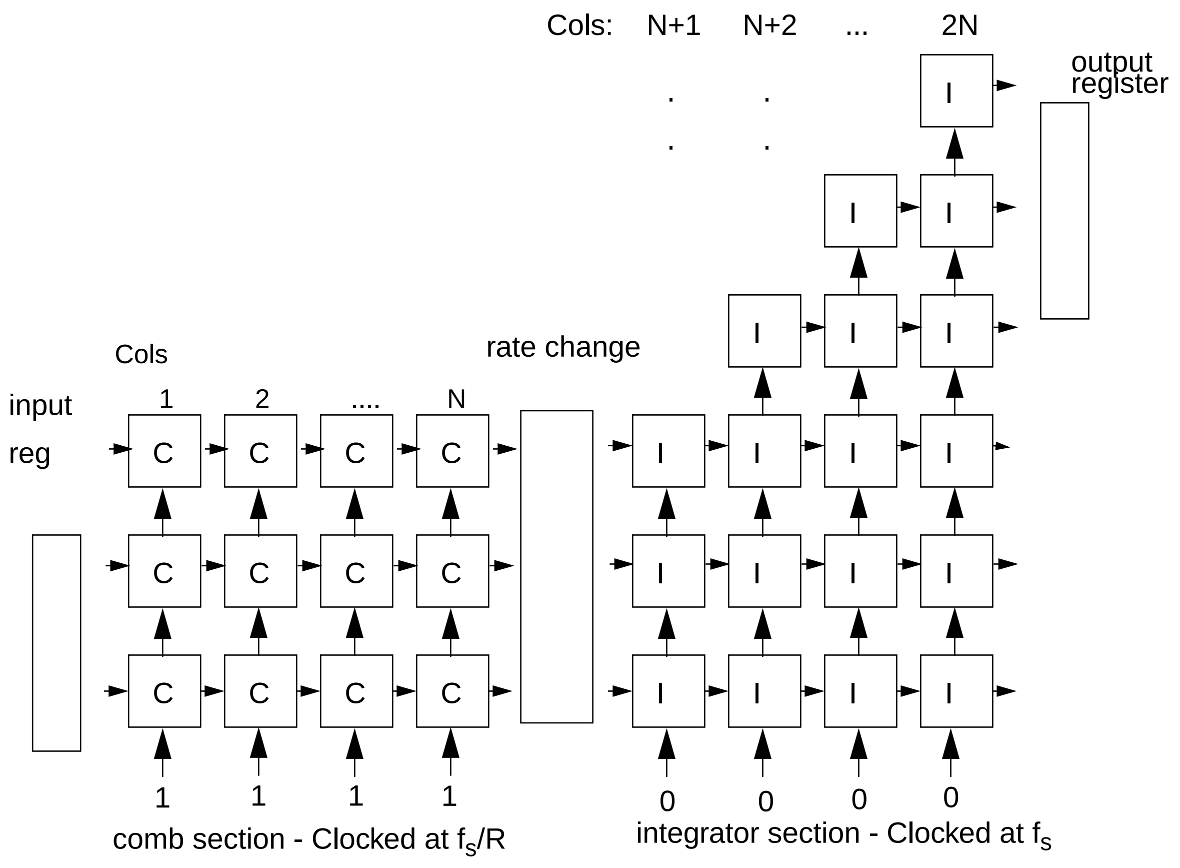


**Figure 1.** **Interpolator Structure**

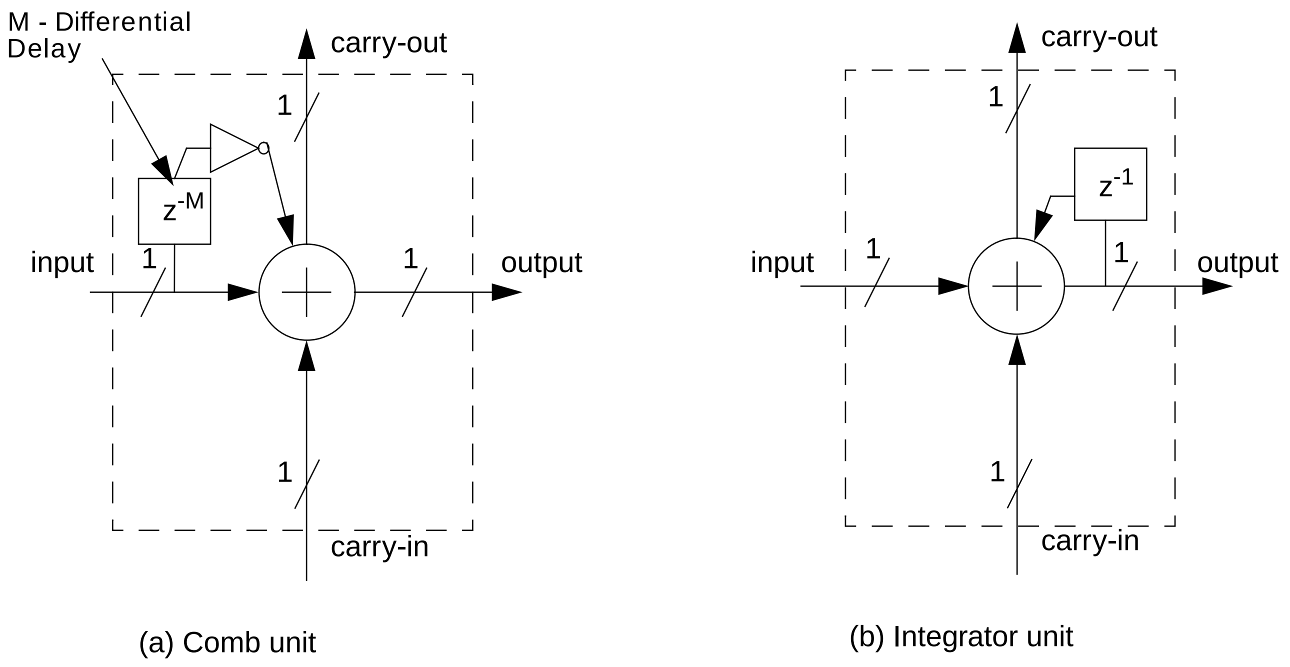


**Figure 2.** **Basic building blocks of the comb and integrator sections**

$$H_I(z) = \frac{1}{(1-z^{-1})}$$

**Eqn 2.**

There is a rate change switch between the two filter sections. The rate increase by a factor of R is obtained by inserting R-1 zero valued samples between consecutive samples of the comb section output.

Combining the system function for the two stages of the filter, the system function for the entire CIC filter with reference to the high sampling rate, $f_S$, is:

$$H(z) = H_I(z)H_C(z) = \frac{(1-z^{-RM})^N}{(1-z^{-1})^N}$$

$$= \left[\sum_{k=0}^{RM-1} z^{-k}\right]^N$$

**Eqn 3.**

This shows a CIC filter is the same as a cascade of N uniform FIR filters.

## Why use the XC6200?

- **Speed** - DSP and micro-processors cannot support the high clock rates required for these filters.
- **Ease of design** - ASIC designs, although a feasible alternative, have a slow turnaround time. This design, using the XC6200 software support, took less than a week to implement and test. The FPGA design is also easily modified for different frequency responses and rate change factors. This is not possible with an ASIC design.
- **Ease of Design Modification** - Three parameters N, M and R, control the filter performance. These parameters can easily be modified in a VHDL or schematic description, and the complete range of filters can be easily accommodated on a XC6216 (with very minor modifications to layout).
- **Multiplier free** - No multipliers are required. This makes the design very appropriate for FPGA architectures.
- **Fine Grain Architecture** - The fine grain architecture of the XC6200, with abundant registers and gates, easily supports the CIC design. DSP filters like this are often highly structured designs, allowing easy and efficient routing between the cells. The register-per-cell architecture significantly reduces area and makes timing requirements easier to meet. Each comb/ integrator unit requires only a single row of four cells on the XC6200 device, 0.16% of the XC6216.
- **WYSIWYG** - Once the design is placed and routed, it is still easy to understand how it is mapped into the XC6200 array; see Figure 5. The impact of any required design changes can be easily calculated prior to implementation.
- **Pipelining Support** - For high sampling rates, the adders within the design must be pipelined i.e registers inserted to reduce the critical paths. The Register-Rich architecture makes pipelining easy to implement in the XC6200
- **Register-Rich** - One of the main drawbacks of CIC filters in general is viewed to be the large register lengths required on the output of the final integrator block. The Register-Rich architecture of the XC6200 series can more than accommodate the required register lengths The XC6264 could accommodate an output register of length 128, allowing a sample rate conversion of the order of $2^{38}$!
- **Dynamic Reconfiguration** - One filter can be used to implement a large range of rate changes, R. The XC6200 allows the user to easily implement one large register on the output of the integrator block from which the appropriate output bits can be selected. Having random access to the configuration bitstream allows partial reconfiguration of the XC6200. Simple circuit updates of gates or routing can be dynamically implemented. The routing from the output register to the IOB pads can be dynamically reconfigured to select the output bits corresponding to the required rate change factor.
- **High Gate Count** - An example Interpolator design requires approximately 10,000 gates and occupies significantly less than half the XC6216. The Interpolator filter can be easily modified to function as a Decimator filter by a simple reversal of the

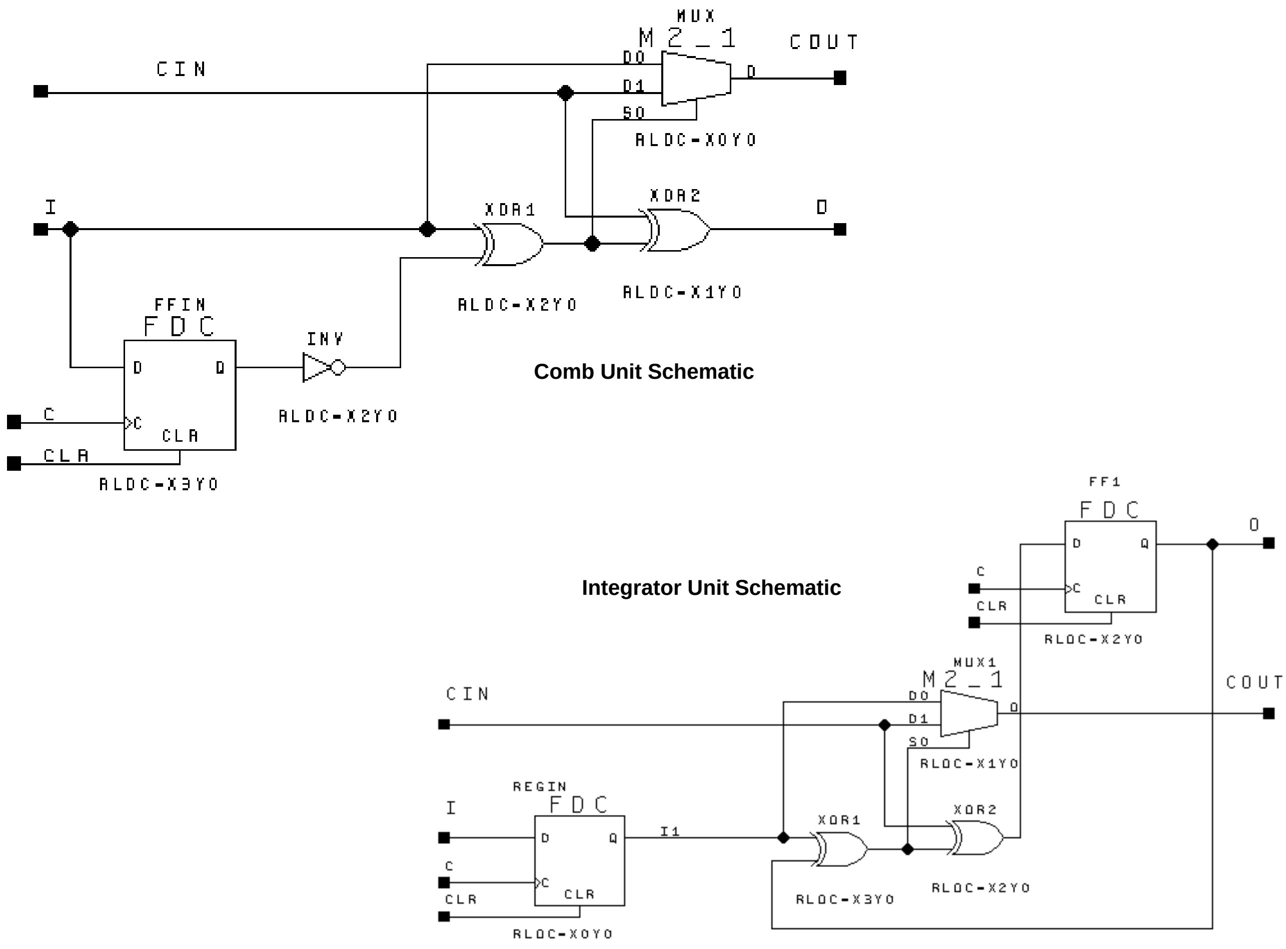


**Figure 3. Schematic representation of the basic building blocks of the comb and integrator sections**

comb and integrator blocks and a slight modification of the rate change block. Both designs, with a total gate count of over 20,000 gates, could be simultaneously fitted in a single XC6216.

- **FastMAP™ Processor Interface -** The FastMAP™ interface is a parallel microprocessor interface which allows efficient access to the internal state of the XC6200. The interface allows read and write access to registers in the user design without the need to route signals to IOB's and pads. In this example, a processor can write data to the input register and read from the output register. These registers are memory-mapped into processor address space. No additional circuitry is required to implement this!

- **Co-Design Verification** - With the XC6200 Development System PCI board, the completed design can be completely verified on real silicon.

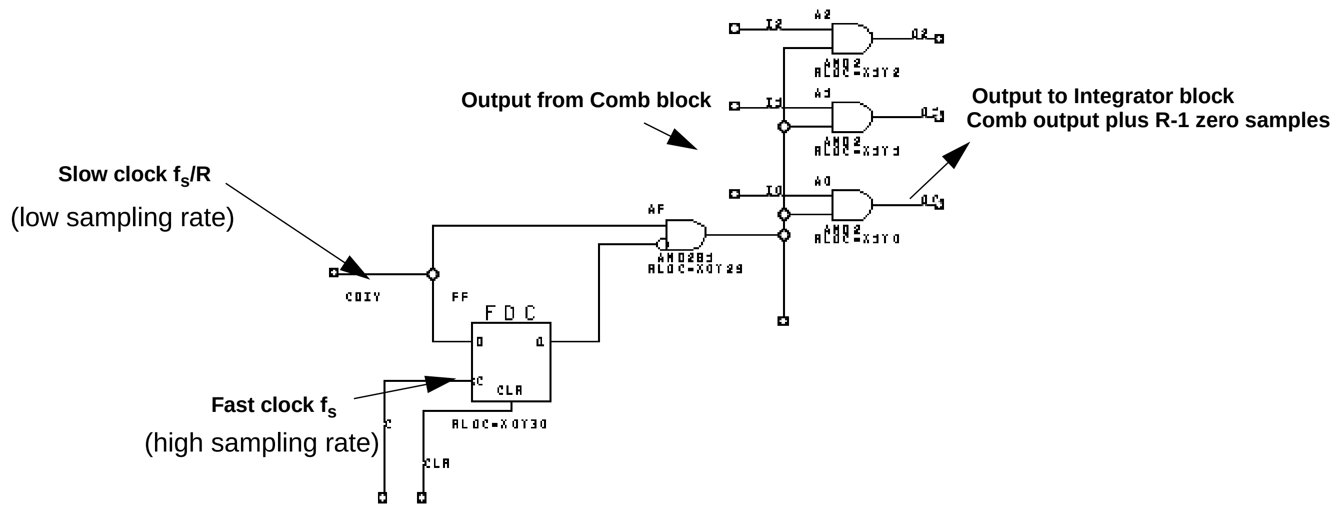


**Figure 4. Rate change block schematic**

## XC6200 Implementation

An Interpolation CIC filter can be easily implemented on the XC6216 chip. The structure consists of a comb filter section followed by the rate change block and finally, the interpolator section.

### Comb

The comb section consists of N columns. Each column consists of a vertical block of the basic comb unit (see Figure 2). The number of comb units per column depends on the input register size and the number of columns, N. For a 16-bit input and N=4, the comb unit only requires 4x4x20 cells, a small percentage of the 4096 cells available on the XC6216.

The basic building block for the comb filter consists of an adder with carry-in and carry-out plus a feedforward delay register or registers, depending on the differential delay, M. The signal from the feedforward register is negated with an invertor as a subtraction is required on the feed-forward path; see Eqn 1. The comb unit, the basic building block of the comb column, is shown in Figure 2 (a), with its schematic in Figure 3.

The basic comb unit can be implemented in four cells of a XC6200. The entire comb column of the Interpolator consists of a simple stack of the required number of comb units. The actual number of comb units required depends on the number of input bits and the number of comb columns, N.

The maximum number of rows for the jth column can be shown to be

$$G_j = \begin{cases} 2^j, \; j = 1, \ldots, N \\ \dfrac{2^{2N-j}(RM)^{j-N}}{R}, \; j = N+1, \ldots, 2N \end{cases}$$

**Eqn 4.**

Minimum register width is:

$$W_j = [B_{in} + \log(G_j)]$$

**Eqn 5.**

(where $B_{in}$ is the input register length).

The comb section is clocked at the lower sampling rate, therefore no pipelining is required either vertically on the ripple-carry or horizontally between the comb columns.

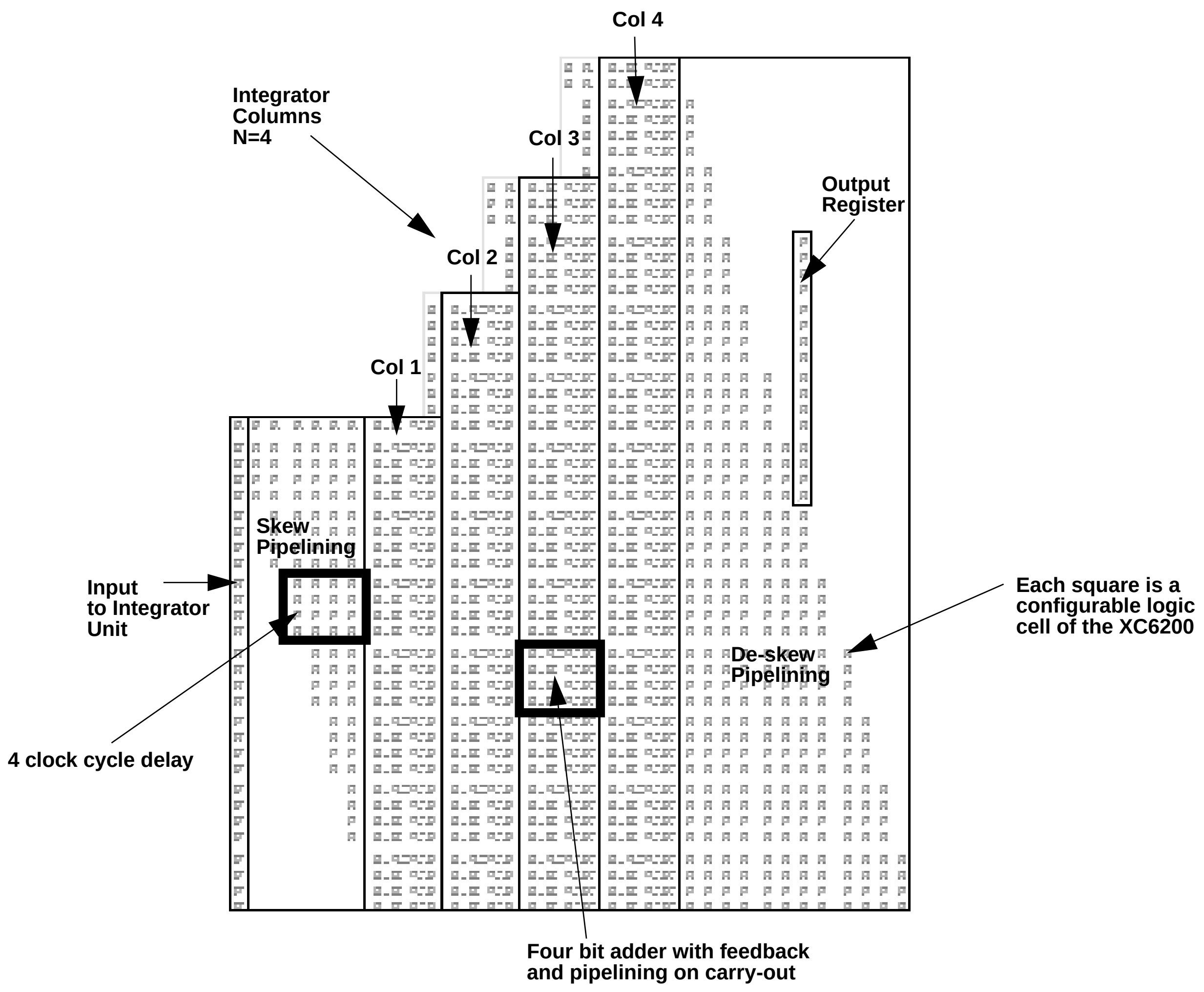


**Figure 5.** **Layout of the Integrator block on the XC6200 clearly showing the skew on the input signal and the de-skew on the output**

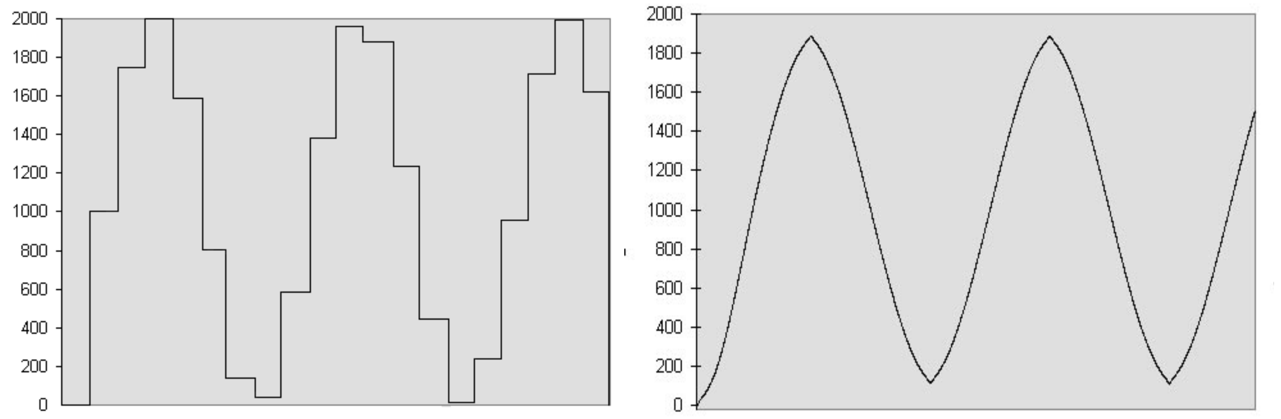
## Rate change

There is a rate change between the two filter sections of the design. The switch causes a rate increase factor of R and is achieved by inserting R-1 zero valued samples between consecutive samples of the comb section output. Figure 4 shows the schematic for the rate change block. The slow clock is combined with a one fast clock cycle delay of itself to give a simple pulse output. The pulse output is combined with the comb section output to produce the comb output interspersed with R-1 zeroes. This example assumes a certain timing relationship between the Slow clock and the Fast clock. Other schemes could be employed where this is not the case.

## Integrator

The integrator section of the design is clocked at the high sampling rate. The integrator section consists of N integrator columns. Each column consists of a stack of the basic integrator unit. The basic integrator unit is a simple adder with a carry-in and carry-out plus a feedback register; see Figure 2 (b). The schematic for the integrator unit is shown in Figure 3.

A basic four bit adder block with carry has a critical path of around 20ns. Dividing the integrator blocks into groups of four and pipelining the carry between each adder section allows the integrator section to be clocked at up to 50MHz. In order to accommodate the carry pipelining, the data on the input to the Integrator section must be skewed by the appropriate



**(a) 20 point sine wave input to CIC Filter** **(b) 20 point x 256 output from CIC Filter**

**Figure 6. Sine wave input and output from XC6200DS**

number of pipelines per adder block. Each four block adder within the integrator column requires one additional clock cycle delay of the input. A 28 bit integrator column requires a pipeline delay of 7 clock cycles. Figure 5 clearly shows the skew effect of the pipelining. This skew pipelining is only required at the input to the integrator block. The data moves through the N integrator columns as skewed data and is de-skewed on the final Nth integrator output. For a 48-bit high final integrator column, 12 clock delays are required on the lowest adder block output. The layout for a four block Integrator section on the XC6200 is shown in Figure 5. The skew and de-skew pipelining is clearly visible.

The size of the integrator column increases across the design according to Eqn 5. Thus for a 16-bit data input and a rate change factor of 256, the length of the final register will be 40 bits. The same design can be used for any rate changes up to a maximum limit depending on the length of the final integrator column. The output data must be selected from the correct location in the column output register. If the actual number of bits output is $B_{out}$, then the number of LSB's discarded is given by:

$$B_T = W_{2N} - B_{out}$$

**Eqn 6.**

For a 16-bit data in, 16-bit data out design with a rate change factor, R, of 256 and the number of columns, N, set to 4, the design occupies 420 cells for the comb section and 1440 cells for the integrator. The total design occupies significantly less than half of the 4096 cells available on the XC6216.

## Co-design on the XC6200DS

The XC6200 Development System (Ref. 3) comprises some low-level software and a PCI board containing an empty XC6216 and a XC4013E to implement the PCI protocols and some other functions. The Xilinx XC6200 chip contains a number of unique features to support hardware/software co-design. The FastMAP$^{TM}$ interface is a parallel microprocessor interface which allows efficient access to the internal state of the chip. With the XC6200DS software, reads and writes of up to 32 bits can be made to any 32 registers in a column. The value on any gate, multiplexer or register output may be read from the circuit, which looks like an SRAM to the host processor. The design can be modified to use built in features of the XC6200 which allow the circuit to be clocked once every time the software writes to a particular memory address.

A complete Interpolator for sample rate conversion of up to times 256 (R = 256) was implemented in a XC6216. The number of data bits input and output was 16 and the length of the final integrator column was 40 bits. The differential delay was set to 1 (M = 1) and the number of columns used was 4 (N=4). The design occupied less than half of the XC6216. The design was implemented using WorkView office schematic input and completely functionally tested using the XC6200 Development System (see Ref. 3). The design was clocked using register writes on the 16-bit register input. Each write corresponded to a cycle of the high sample rate clock. In order to achieve the high clock rate, the same value was written to the input register R times (where R is the ratio of the high sampling rate to low sampling rate), i.e. the input was only changed at the slow clock rate. A simple counter design implemented the slow clock. The rate change factor could be varied by a write to a register with a comparator tied to the counter output.

Through a very simple C$^{++}$ interface, a complete functional test of the entire design was carried out. The output was read from the appropriate bits (depending on the rate change factor) in the output column. Standard wave files were written to the filter and the resulting high sample rate files were read from the register output and saved to a wave file. Figure 6 shows the result of inputting 20 points of a sine wave and the resulting output sine wave. The C$^{++}$ interface Code is outlined in Appendix A: XC6200DS C$^{++}$ interface code for the CIC Filter.

## Summary

The regular, multiplication-free, CIC filters are highly suited to implementation in the XC6200 architecture. There are many other similarly structured DSP problems which are also well suited to the XC6200 architecture. The XC6200DS PCI development system provides an out-of-the box platform on which designs can be tried and tested on real silicon before the final implementations in the user system.

## References

1. An economical Class of Digital Filters for Decimation and Interpolation, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-29 No.2, April 1981 pp 155 - 162
2. Xilinx XC6200 Field Programmable Gate Arrays Product Description.
3. Xilinx XC6200 Development System Datasheet

## Limitations And Restrictions

## Appendix A: XC6200DS C$^{++}$ interface code for the CIC Filter.

```
/* CIC Filter XC6200DS C Interface */

        board->initialize();
        board->clock_on();
        board->reset();

        // PCI CLOCK at 66MHz
        ref_freq = 16.0;
        des_freq = 66.0;
        board->set_ref(0);
        freq = board->set_clock_freq((float)ref_freq,(float) des_freq);

        // reset the board
        board->reset();

        // load in cal file
        if (board->load_cal_file(CALFILE)){
                cout << “Problem loading CAL file “ << CALFILE << “: Exiting!” <<  endl;
                exit(1);
        }
        board->set_bus_width(16);
        board->set_mask(all32Bits);

        // clear out input register
        board->set_map(mask_in_low,mask_high);
        board->set_column(filt_in,0);

        // reset counter
        board->set_map(mask_rst,mask_high);
        board->set_column(rst,0x1);

        // assert GCLR through write to bit 16 in input reg
        board->set_map(mask_clr,mask_high);
        board->set_column(clr,0x1);

        // buff clk
        board->set_map(mask_in_low,mask_high);
        for(i=0;i<9;i++)
                board->set_column(filt_in,0);

        // clear low
        board->set_map(mask_clr,mask_high);
```

```
board->set_column(clr,0x0);

// set clock generation to appropriate value
board->set_map(mask_cdiv_l,mask_cdiv_h);
rt_val = UP_SAMPLE/2 -1;
board->set_column(cdiv_set,rt_val);

// unreset counter
board->set_map(mask_rst,mask_high);
board->set_column(rst,0x0);



WaveRead = FALSE;
while(TRUE){
        // get data from input wave file
        data = (short)(128 - (BYTE)*(pSndCard++));
        if(i >= DataPoints){
                WaveRead = TRUE;
                break;
        }
        for(k= 0;k<UP_SAMPLE;k++){
                board->set_map(mask_in_low,mask_high);
                // write to the input register
                board->set_column(filt_in,data);
                board->set_map(mask_out_low,mask_out_high);
                // read data from the output register
                data_out = (int) board->get_column(filt_out);
                // update the output wave file
                *pDataOut = (BYTE)(128 - (data_out &0xFFFF));
                pDataOut++;
        }
}
```