Implementation Results of the Digital IF Processor

Grant Hampson

May 15, 2002

Introduction

This document presents results from an implementation of the IIP Radiometer Digital IF processor described in [1]. A photo of the digital IF processor board is shown in Figure 1. The input sample rate of the AD9410 is $F_s = 200MHz$ providing a band width of 100MHz. The spectrum is shifted down in frequency by 50MHz (FS/4 down) and filtered to remove the image component. The spectrum is then shifted up in frequency by 25MHz (FS/4 up.) The output of the digital IF processor is a 100MHz complex output with slightly less than 50MHz of band width.

The first section of this document discusses the implementation. Secondly, the experimental setup is described, and in the third section measured results are presented.

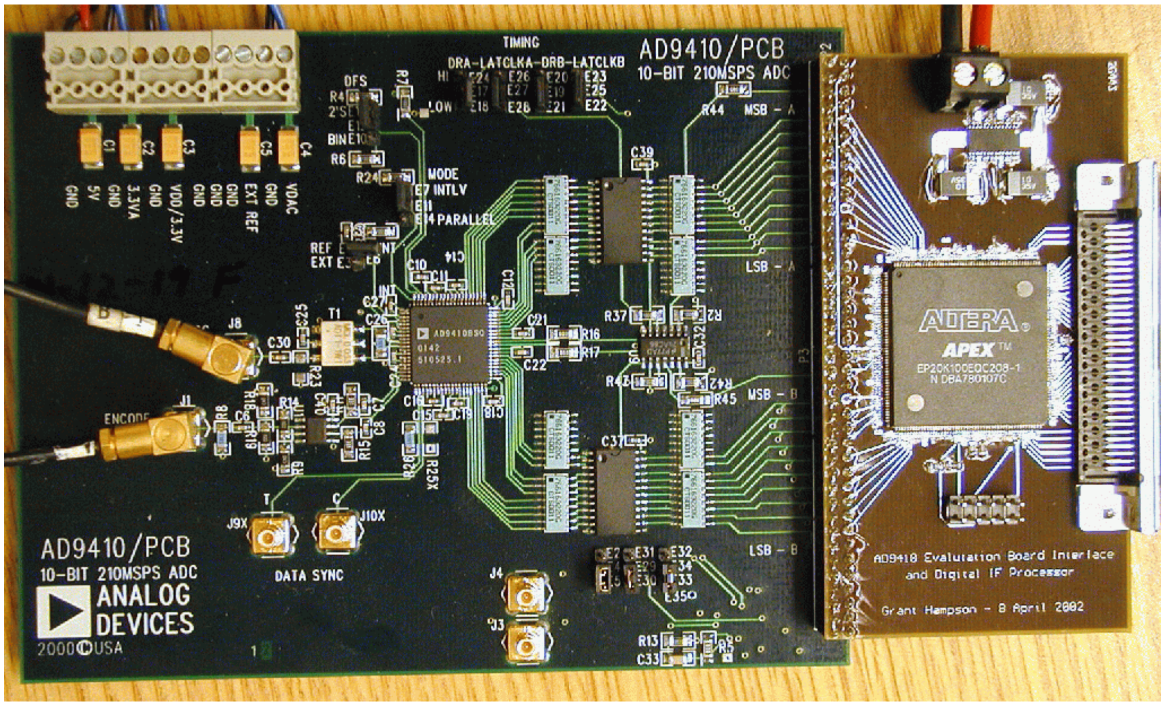


Figure 1: AD9410 evaluation board and Digital IF processor implemented in a \$150 APEX FPGA (EP20K100EQC208-1). The APEX FPGA consumes approximately 800mA (1.8V and 3.3V) when operational. The processor output is connected to a capture card which interfaces to the PC.

1 Digital IF Processor Implementation

The digital IF processor presented here is a slightly different to that described previously in [1], in that it is described using AHDL text file instead of a graphical representation. The functionality of the design has remained the same though. Refer to Appendix A for the AHDL source code. There are two lines of this code which have particular impact on the design, which are:

```
realfilt_reg[15..0] = realfilter.fir_result[19..4]; -- [23..0] full res
imagfilt_reg[15..0] = imagfilter.fir_result[19..4]; -- [23..0] full res
```

These two lines select which bits of the FIR filter [2] output are used. The output resolution of Digital IF processor is limited to 16-bits real and imaginary. The output of the filter is 24-bits wide and consequently some truncation is required (MSBs and/or LSBs.) A measurement of the processor outputs with a full scale sinusoidal input revealed the output only occupied 11 LSBs. It was decided to truncate 4-MSB's and 4-LSB's to gain better precision (all 16-bits are effectively occupied then.)

The digital IF processor target FPGA is the Altera APEX FPGA, part number EP20K100EQC208-1. Using this FPGA the design occupies 3543 logic elements (out of a possible 4160, or 85%) and has an estimated maximum clock speed of 133MHz.

The schematic and circuit board layout can be found in Appendix B and Appendix C, respectively. The board has a very simple layout and no complications were encountered. One problem however occurred with the power regulation as it didn't have enough capacity to power the FPGA. Consequently, power regulation is external to the PCB. Voltages sources of 3.3V and 1.8V are soldered directly to the circuit board.

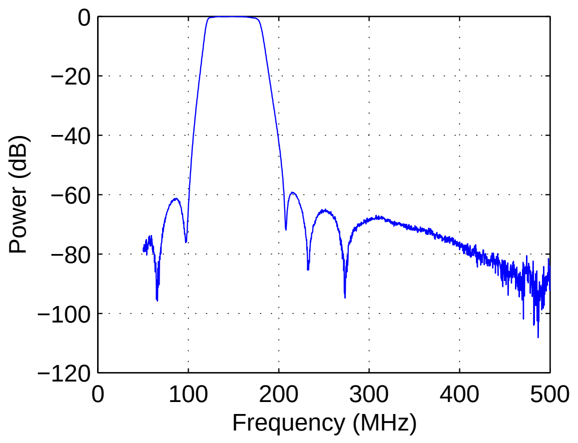
2 Experimental Setup

The experimental setup for the digital IF system requires a brief description here for clarity. The processing chain is as follows: an IF signal is filtered by an anti-aliasing band-pass filter with cut off frequencies of 120-180MHz. This enters the ADC in the second Nyquist zone and then to the APEX FPGA. The FPGA has two modes; the first is to pass raw data and the second is the digital IF processor. This data is then captured and transferred to the PC. This experimental setup will be further referred to as *the system*. The anti-aliasing filter characteristics are shown in Figure 2. The pass band has a very small ripple (<0.1dB).

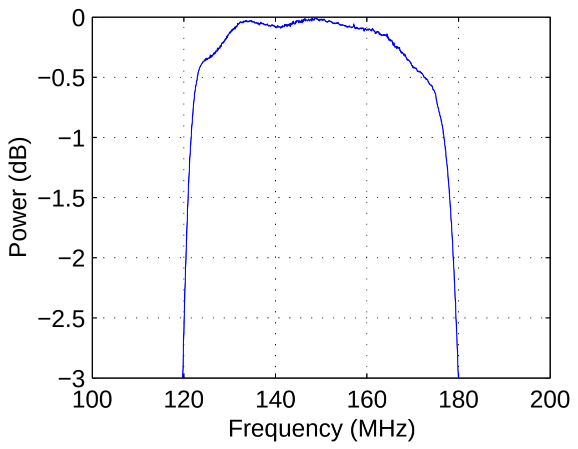
3 Experimental Results

The first simple experiment to be conducted was to inject a 135MHz sinusoid into the system. The result of this simple experiment is shown in Figure 3(a) where the pass and stop bands can be clearly identified, as well as the sinusoid. The location of the sinusoid can be predicted. Firstly, since we are operating in the second Nyquist zone the input frequencies fold. Secondly the spectrum is shifted down 50MHz, filtered and then shifted up 25MHz. This can be expressed by the following equation:

$$\begin{aligned} F_{out} &= (200 - F_{in}) - 50 + 25 \\ &= 175 - F_{in} \end{aligned} \quad (1)$$

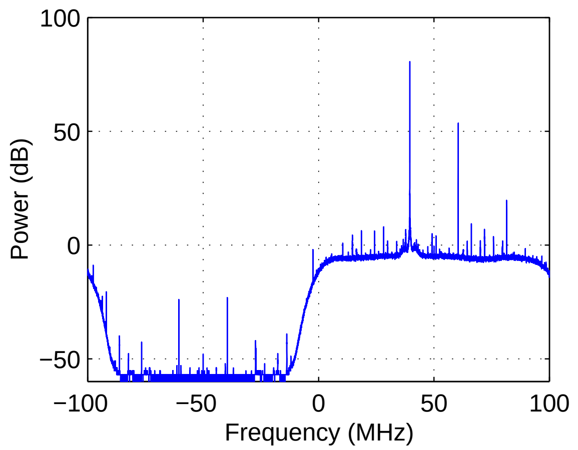


(a) Anti Alias Filter Response

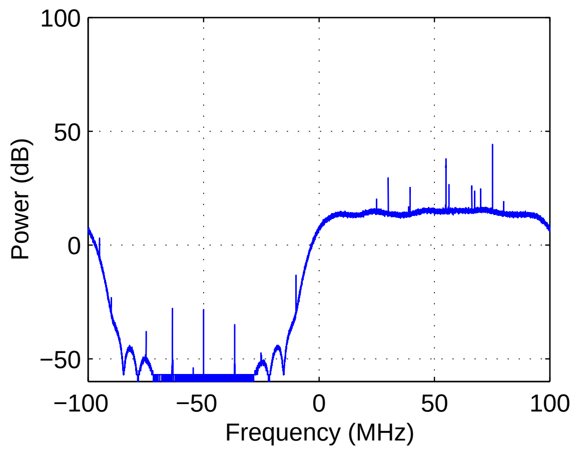


(b) Close up of Filter Pass band

Figure 2: Anti-alias filter responses over the ADC analogue band width. This is measured using an Agilent 8722ET network analyzer.



(a) Measured Response to a 135MHz sinusoid



(b) Measured Response to wide band noise

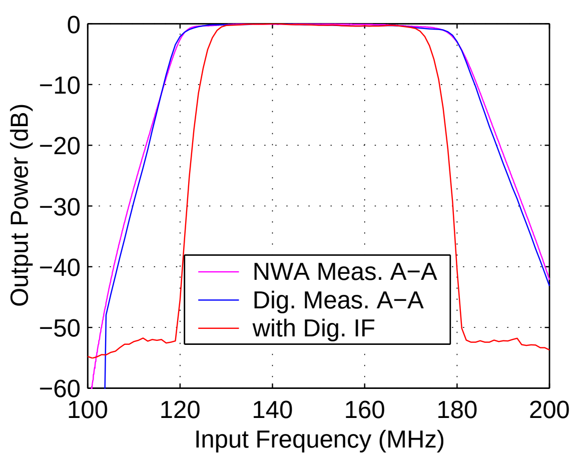
Figure 3: Measured responses of the system with a sinusoidal and noise inputs. The sinusoid power has insufficient noise to *map* out the complete stop band. A wide band noise input has sufficient noise power across the band to see the side lobes of the digital IF filter. These results were obtained by integrating 100 length 32k FFTs.

Hence the *new location* of the 135MHz sinusoid is $F_{out} = 175 - 135 = 40$MHz.

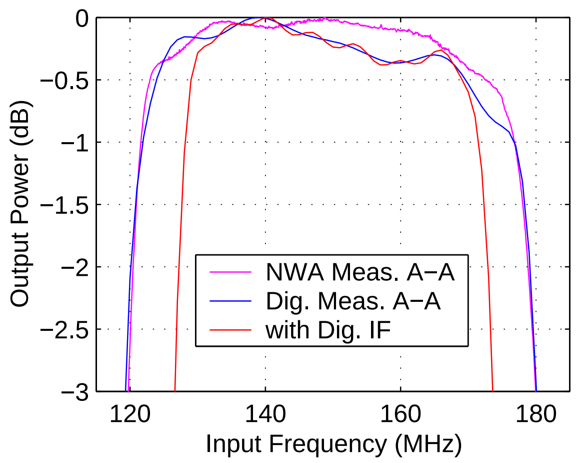
Note that it is difficult to see the true shape of the filter stop band responses in the system as the input noise is below the noise floor. The second input into the system was wide band noise [3] amplified by two amplifiers (Mini-circuits ZFL-500HLN amplifiers) in series. The result is shown in Figure 3(b) where the first two side lobes of the filter stop band can be seen. Unfortunately, adding more gain resulted in oscillations. The stop band attenuation is greater than the desired 60dB.

Next, the pass band shape of the system response was of interest. To do this a HP 8350B sweep oscillator was setup to output frequencies between 100 and 200MHz in 1MHz increments. 32k samples were recorded for each frequency. From each data set the main lobe power was estimated (the sum of ± 15 FFT bins around the fundamental peak.) The results are shown in Figure 4. Two experiments were conducted: with and without the digital IF processor. The resulting curves are plotted on the same graph as the network analyzer measurement of the anti-aliasing filter.

The *digitally* measured pass band of the anti-aliasing filter is in close agreement with the network analyzer measurement. The digital IF processor filter has approximately 47MHz of band width, which is the designed band width. Note that the pass band ripple of the digital IF processor is less than 0.5dB, which is a great result.



(a) Digitally Measured Pass Band Response



(b) Magnified Pass Band Response

Figure 4: (a) The measured NWA response of the anti-aliasing filter (from Figure 2.) The response of the anti-alias filter with and without the digital IF filter are also plotted. The digital IF filter response is much sharper due to the larger number of taps (63). (b) A close up of (a) to determine the amount of filter ripple. The half power points of the digital IF filter are approximately 127MHz to 174MHz; a total bandwidth of 47MHz. The digital IF filter has a faster ripple than the anti-alias filter.

4 Summary and Conclusions

This document has shown the implementation results of the digital IF processor for various input stimuli. A new implementation method using a AHDL text file was shown first. Schematic and PCB layouts for the digital IF processor were also shown. The design fits nicely on an APEX FPGA which cost \$150. Its estimated maximum operating frequency is well above the desired 100MHz.

Various stimuli were used to probe the pass and stop bands of the anti-aliasing filter and digital IF processor. The results are in firm agreement with the desired specifications.

The digital radiometer specifications [4] call for two of the digital IF processors discussed here. Several options exist for this implementation. It could be possible to implement both channels on the same FPGA, or two individual FPGAs with an interconnecting bus (for the summation.) Note that when this stage occurs we will no longer use the AD9410 evaluation board. Two AD9410 will be integrated on the same PCB as the FPGA (or possibly using two connectors.)

References

[1] G. A. Hampson, "An FPGA Implementation of the Digital IF Processor," March 7 2002. http://esl.eng.ohio-state.edu/rfse/iip/digitalif.pdf.

[2] *FIR Compiler MegaCore Function*, Altera Corporation, December 2001. http://www.altera.com/literature/ug/fircompiler_ug.pdf.

[3] *RAS-10/2000 Calibrated Noise Source*, Radio Astronomy Supplies. http://www.nitehawk.com/rasmit/jml0.html.

[4] S. W. Ellingson, "Design of the IIP Radiometer Digital IF Section," February 11 2002. http://esl.eng.ohio-state.edu/rfse/iip/iipdigif.pdf.

Appendix A: Digital IF Processor AHDL Code

```
-- Digital IF processor
-- Grant Hampson 30 April 2002

INCLUDE "lpm_add_sub.inc";
INCLUDE "realfilter_st.inc";
INCLUDE "imagfilter_st.inc";

SUBDESIGN ad9410interface
(
   dra,         -- clock from AD9410 development board for Data bus M
   drb,         -- clock from AD9410 development board for Data bus N
   dm[9..0],    -- Data bus M from AD9410 development board
   dn[9..0]     -- Data bus N from AD9410 development board
      :INPUT;

   control1,    -- control lines to general connector
   control2,
   real[15..0], -- data outputs for general connector (to FIFO, APBE, FFT)
   imag[15..0]
      :OUTPUT;
)

VARIABLE
   dm_register[9..0], dn_register[9..0],     -- input registers
   sync_dm_reg[9..0],                        -- synchronisation register
   negm_reg[9..0], negn_reg[9..0],           -- registers after negation
   add_delay_n[9..0],                        -- delay due to filter lengths
   realfilt_reg[15..0], imagfilt_reg[15..0], -- registers after filters
   swapn_reg[15..0], swapm_reg[15..0],       -- registers after swap stage
   outreal_reg[15..0], outimag_reg[15..0],   -- output registers
   cont_sm[1..0] : DFF;                      -- controller statemachine

   neginput, swap, negimag, negreal : NODE;  -- controller outputs

   negator_m,
   negator_n : lpm_add_sub WITH(LPM_WIDTH = 10,
                               LPM_REPRESENTATION = "SIGNED",
                               LPM_DIRECTION = "SUB",
                               LPM_ONE_INPUT_IS_CONSTANT = "YES");

   negator_real,
   negator_imag : lpm_add_sub WITH(LPM_WIDTH = 16,
                                  LPM_REPRESENTATION = "SIGNED",
                                  LPM_DIRECTION = "SUB",
                                  LPM_ONE_INPUT_IS_CONSTANT = "YES");

   realfilter : realfilter_st WITH();   -- FIR filters
   imagfilter : imagfilter_st WITH();

BEGIN
   dm_register[].clk = dra;             -- Latch inputs using correct clock
   dm_register[].d = dm[];
   dn_register[].clk = drb;
   dn_register[].d = dn[];
```

```
sync_dm_reg[].d = dm_register[].q; -- Now both data paths syncronised
sync_dm_reg[].clk = drb;

negator_m.dataa[] = GND;
negator_m.datab[] = sync_dm_reg[];
negator_n.dataa[] = GND;
negator_n.datab[] = dn_register[];

if neginput == B"0" then
   negm_reg[].d = negator_m.result[];
   negn_reg[].d = dn_register[];
else
   negm_reg[].d = sync_dm_reg[];
   negn_reg[].d = negator_n.result[];
end if;
negm_reg[].clk = drb;
negn_reg[].clk = drb;

add_delay_n[].d = negn_reg[].q;      -- delay for different filter lengths
add_delay_n[].clk = drb;

realfilter.clk = drb;                -- instance of the real filter
realfilter.data_in[9..0] = negm_reg[9..0];
realfilter.rst = GND;
realfilter.clk_en = VCC;
realfilt_reg[15..0] = realfilter.fir_result[19..4]; -- [23..0] full res
realfilt_reg[].clk = drb;

imagfilter.clk = drb;                -- instance of the imag filter
imagfilter.data_in[9..0] = add_delay_n[9..0];
imagfilter.rst = GND;
imagfilter.clk_en = VCC;
imagfilt_reg[15..0] = imagfilter.fir_result[19..4]; -- [23..0] full res
imagfilt_reg[].clk = drb;

swapn_reg[].clk = drb;               -- swapping of real and imaginary
swapm_reg[].clk = drb;
if swap == B"0" then
   swapm_reg[].d = realfilt_reg[];
   swapn_reg[].d = imagfilt_reg[];
else
   swapm_reg[].d = imagfilt_reg[];
   swapn_reg[].d = realfilt_reg[];
end if;

negator_real.dataa[] = GND;          -- negation of real or imaginary
negator_real.datab[] = swapm_reg[];
negator_imag.dataa[] = GND;
negator_imag.datab[] = swapn_reg[];

outreal_reg[].clk = drb;             -- connection of ouput registers
if negreal == B"0" then
   outreal_reg[].d = swapm_reg[];
else
   outreal_reg[].d = negator_real.result[];
   end if;

   outimag_reg[].clk = drb;
   if negimag == B"0" then
      outimag_reg[].d = swapn_reg[];
   else
      outimag_reg[].d = negator_imag.result[];
   end if;

   TABLE  -- State Machine for controlling Digital IF Processor
   cont_sm[].q => cont_sm[].d, neginput, swap, negimag, negreal;
         B"00" =>      B"01",     B"0", B"1",    B"0",    B"0";
         B"01" =>      B"10",     B"1", B"0",    B"0",    B"1";
         B"10" =>      B"11",     B"0", B"1",    B"1",    B"1";
         B"11" =>      B"00",     B"1", B"0",    B"1",    B"0";
   END TABLE;
   cont_sm[].clk = drb;

   real[15..0] = outreal_reg[];
   imag[15..0] = outimag_reg[];
   control1 = drb;
   control2 = GND;
END;
```

Appendix B: Digital IF Processor Schematic

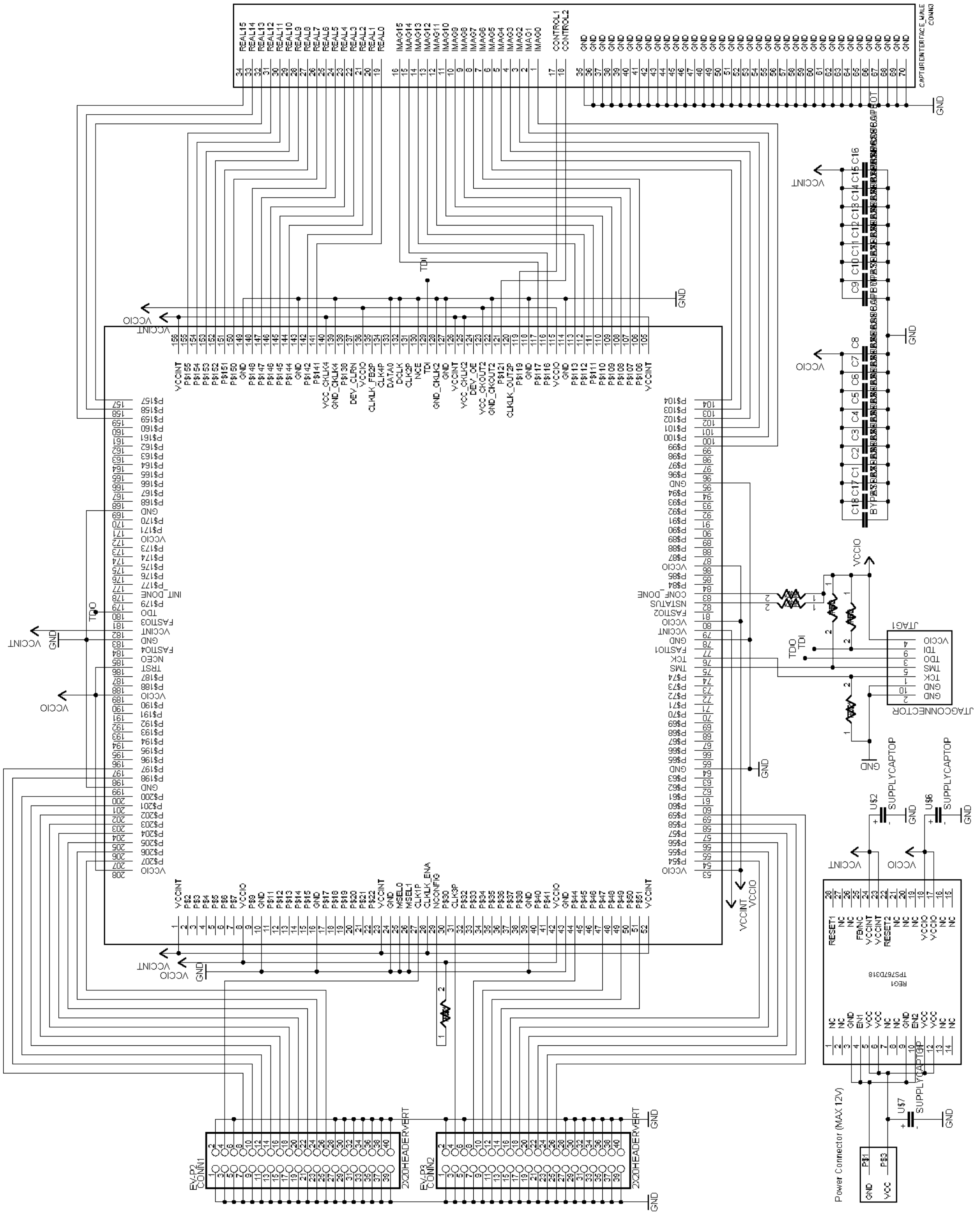
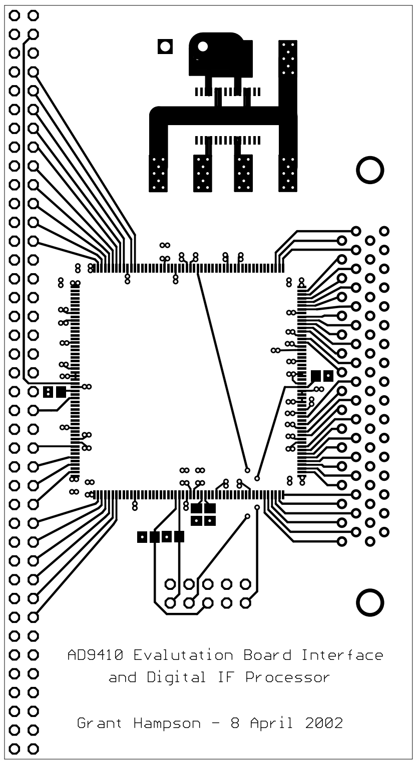
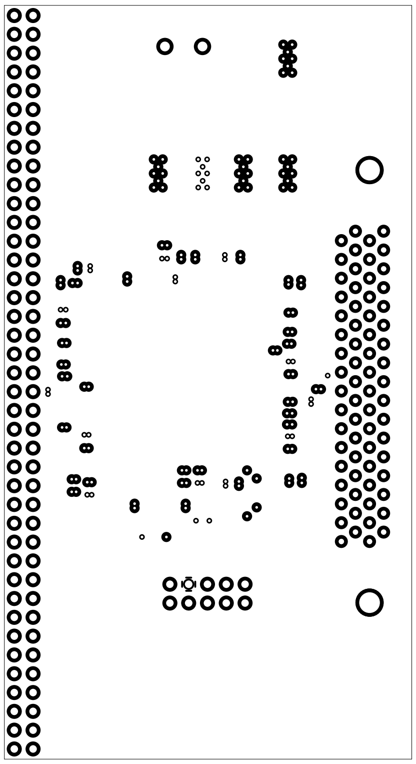


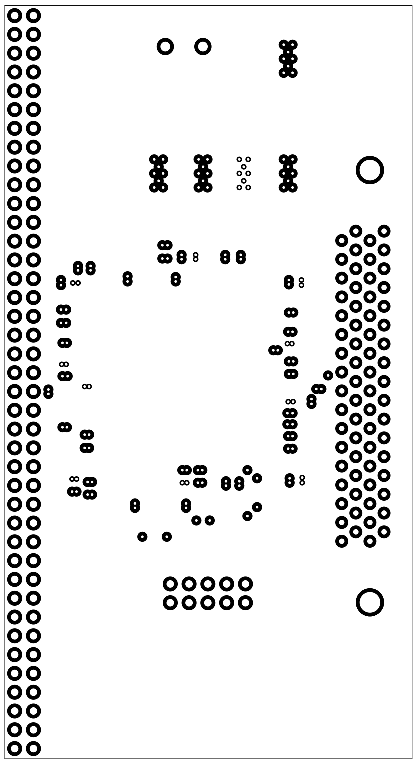
Figure 5: The schematic of the Digital IF processor board consists of an APEX FPGA and two connectors. Power regulation now occurs external to the PCB.

Appendix C: Digital IF Processor Layout Plots


AD9410 Evalutation Board Interface
and Digital IF Processor
Grant Hampson - 8 April 2002

(a) Component Side


(b) VCCIO (Negative Image)


(c) VCCINT (Negative Image)

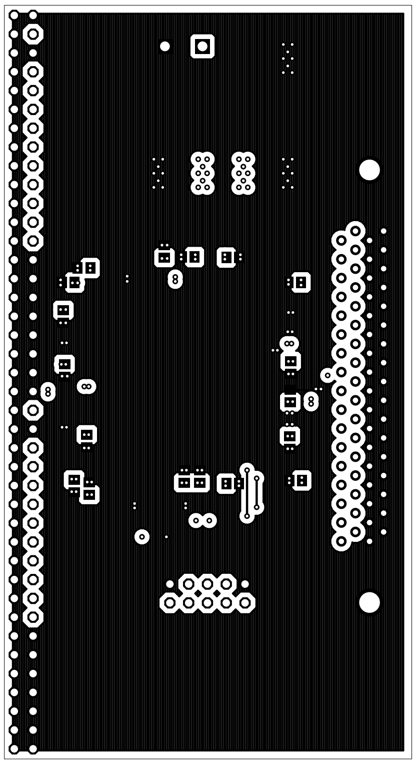

(d) Solder Side

Figure 6: The four layers of the Digital IF processor board. This board is manufactured by PCBexpress (`www.pcbexpress.com`) and costs $75.