A Post FFT Spectral Integration Processor Implementation and Measurement Results

Grant Hampson

September 20, 2002

Introduction

This document describes an initial post-FFT spectral integration processor for the IIP digital radiometer. To date, a single FFT processor [1] and post-FFT power estimator [2] have been implemented. The specification of the IIP digital radiometer [3] calls for total power radiometry, however an intermediate stage is spectral integration.

Figure 1 illustrates the spectral FFT processing chain. The FFT processor produces 1024 complex values which are transformed using the FFT. The ABS block performs i^2+q^2 to calculate power. The spectral integrator then integrates these values, bin by bin, over a given integration period. Due to the fixed point arithmetic, the integrator data path grows. For example, integrating 1024 fixed point values would require an extra 10-bits to maintain full precision. However, the final mean power would be the full precision value truncated by 10-bits. To limit the hardware required to calculate the mean power the number of integrations is limited to powers of two.

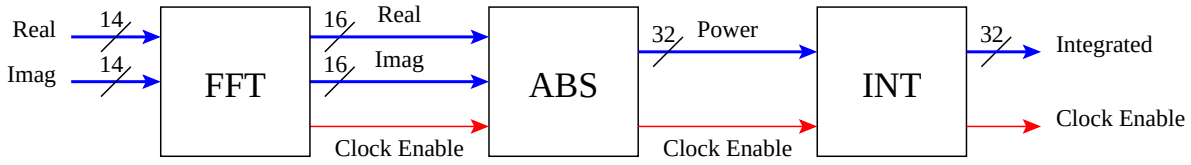
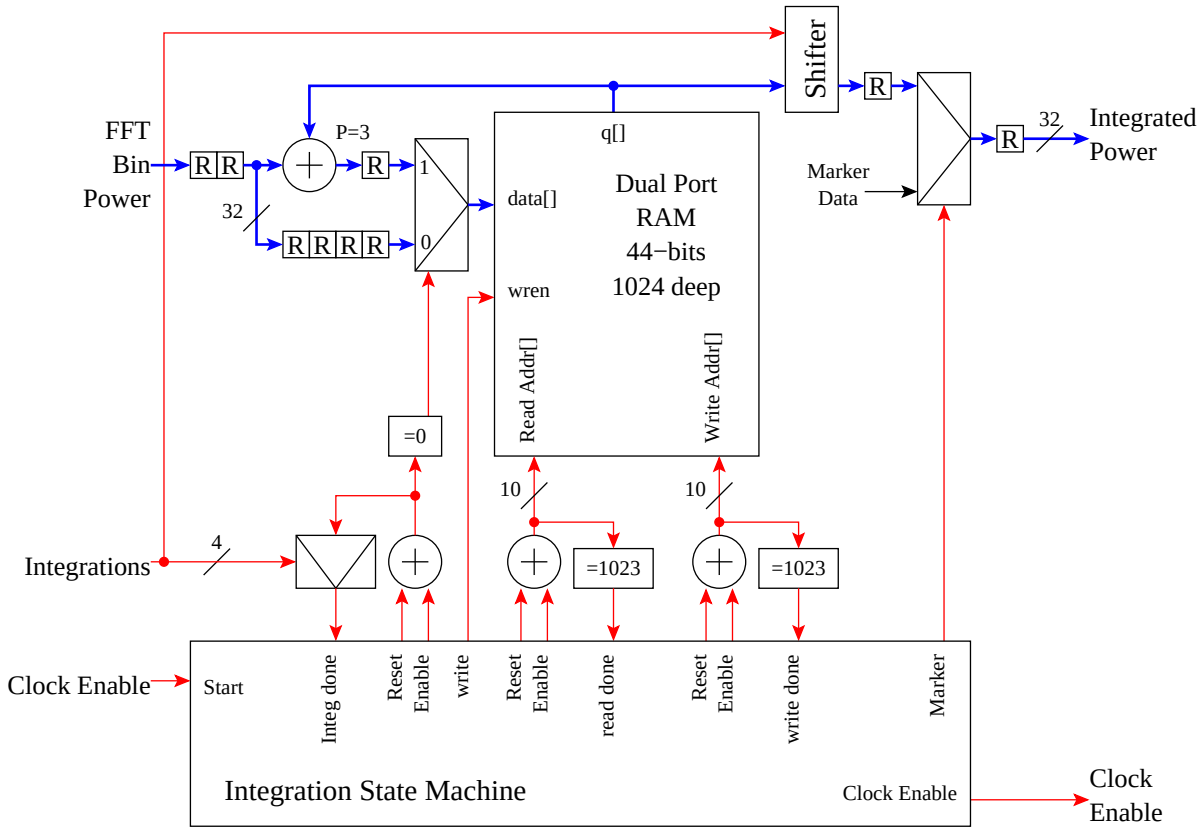


Figure 1: IIP Digital Radiometer stages: FFT, Absolute Value and spectral integration.

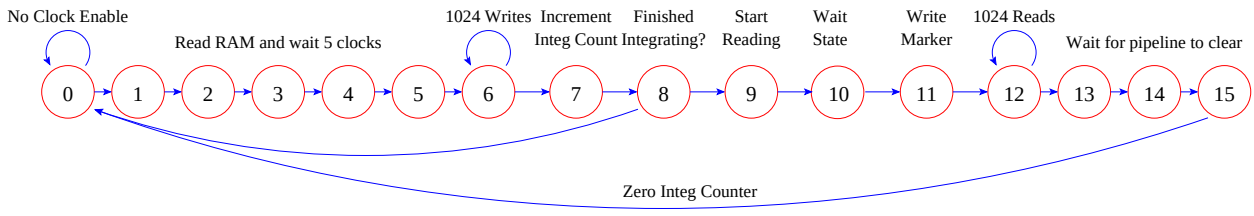
Given that the integrator data path grows with the number of integrations, an upper limit has to be set so that the integrator can still perform at 100MHz and also sufficient FPGA memory is still available. This limit has been found to be 4096 integrations for the current FPGA size. It is therefore possible to have integrations times up to $4096 \times 1024/100\,\text{MSPS} \approx 42\text{ms}$. Furthermore, using an integration block size of 4096 results in an output sample rate equal to $100\,\text{MSPS}/4096 \approx 25$ kSPS, which is easily read into the PC. The PC then performs any final post-integration/processing and storage. Using this technique it is possible to store 254 integrations in the capture FIFO which corresponds to a total integration time of $254 \times 0.042 = 10.7$ seconds, which is adequate.

1 Spectral Integration Processor

The Spectral Integrator Processor (SIP) is shown in Figure 2. The SIP resource requirements are dominated by the large amount of dual port RAM required to store the integration results. Two ten bit counters provide read and write addresses for the RAM. The adder has 3 pipeline stages to account the large data path width of 44-bits. When the integration counter is zero the FFT data passes directly in to the RAM, other wise it is added to accumulated data. When the required number of integrations is reached a marker is written, as well as 1024 scaled (right shifter) integrations. The amount of right shift applied is proportional to the number of integrations. The state machine for the SIP is also shown in Figure 2 for reference. The SIP consumes 624 LE (15%) and 45312 RAM bits (85%) and has a maximum operating speed of approximately 110MHz.



(a) SIP Hardware



(b) SIP State Machine

Figure 2: Spectral Integrator Processor (SIP) block diagram and state machine.

2 System Characteristics

With the completion of the integration stage of the IIP Radiometer, it is possible to integrate relatively large quantities of data in a relatively short time (as opposed to software capture-FFT-integrate.) Figure 3 illustrates a total system check to confirm the results of the ADC-DIGIF-APB-WIN-FFT-ABS-INT-FIFO processing chain of the IIP Radiometer (4 FPGA stages.) This confirmation is in the form of a Spectrum analyzer measurement of the input to the ADC.

Two different measurements are performed - firstly for the antenna connected to the LNA and secondly with the LNA terminated (using a RF switch [4].) The first measurement provides a system baseline, which can then used to remove the system component from the antenna measurement. Both the spectrum analyzer and IIP radiometer produce similar results. It is believed that the radiometer radiates RFI which produces two spikes in the antenna measurement. These two spikes (25MHz apart) are still present in the terminated measurement, although significantly smaller.

The ATC radar is also present in the antenna measurement, at the frequency of 25MHz (the zero frequency in these plots corresponds to an RF frequency of 1356MHz, +25MHz is 1331MHz, and +50MHz corresponds to 1306MHz.) There is a noticeable roll off with frequency between 25 and 50MHz due to the fact that we are operating on the edge of the RF input band (1325-1725MHz RFI filter).

3 Calibration Source Check

A dynamic range check was performed next using the calibration source input in the RF front-end electronics. Here a calibration source can be added to the LNA input using a directional coupler. The source parameters used here a 1331MHz CW signal of variable power (same frequency as the ATC radar.)

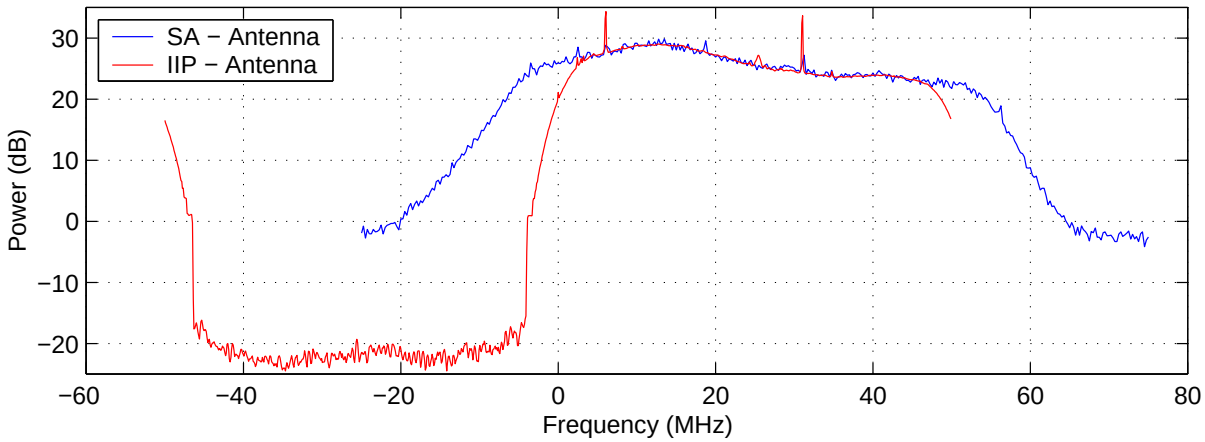
Figure 4 illustrates the results of the calibration source measurement for two types of window - Bartlett and Rectangular. The window produces the expected results in the frequency domain. Note that the CW frequency drift is a source instrumental drift and not due to the radiometer.

It is of interest to note the calibration tone is clearly visible over the 50dB input power variation. The maximum CW power into the IIP Radiometer in this configuration is approximately -30dBm, beyond this the radiometer result is non-linear and unpredictable.

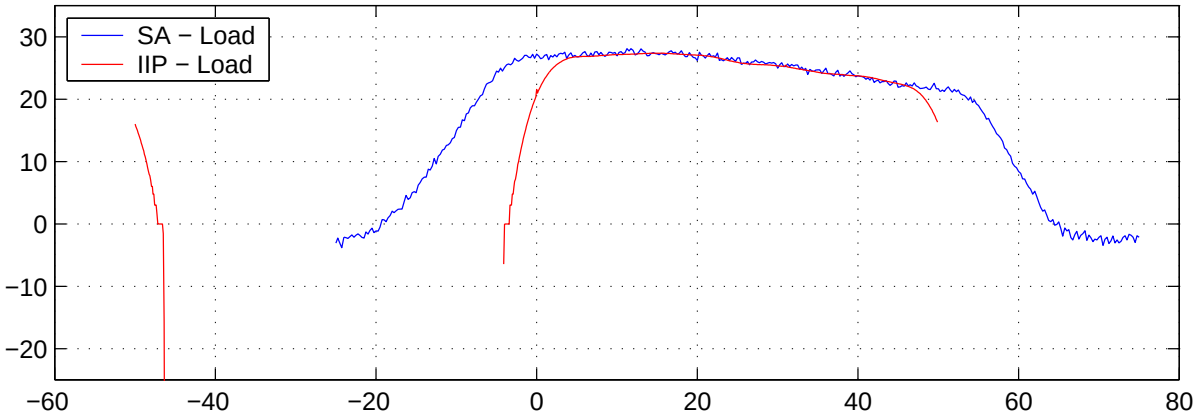
4 Pulse Blanker Results

Since it is now possible to perform long integrations, it is also possible to clearly see the repetition rate of the ATC radar. Figure 5 illustrates the scenario where the pulse blanker is disabled and ATC radar pulses pass through. The ATC radar was captured using four different integration lengths between 512 and 4096 integrations.

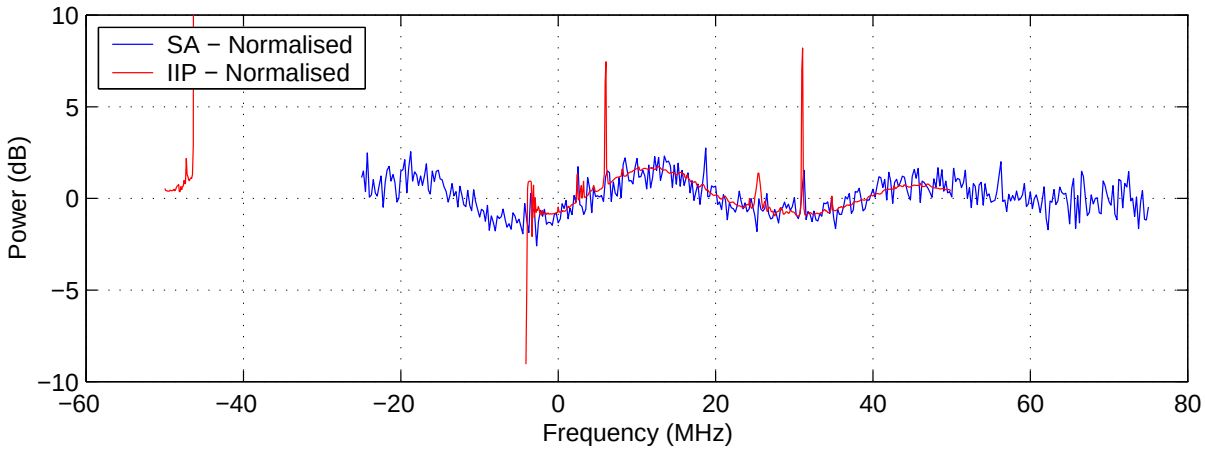
Using 512 integrations per output it is possible to see the ATC pulse in detail. The pulse is visible for almost half a second, of its approximate 10 second rotational period.



(a) Antenna Input

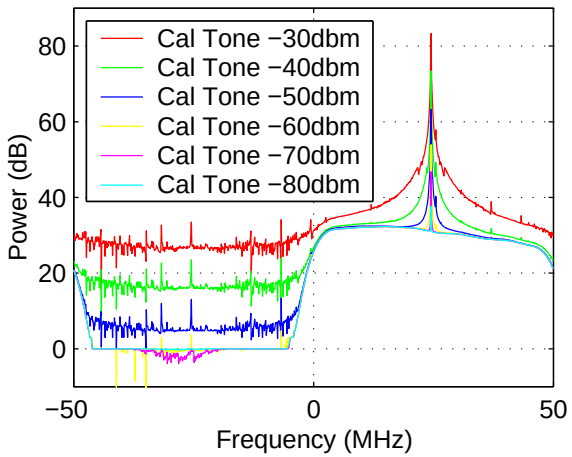


(b) Terminated Input (50Ω load)

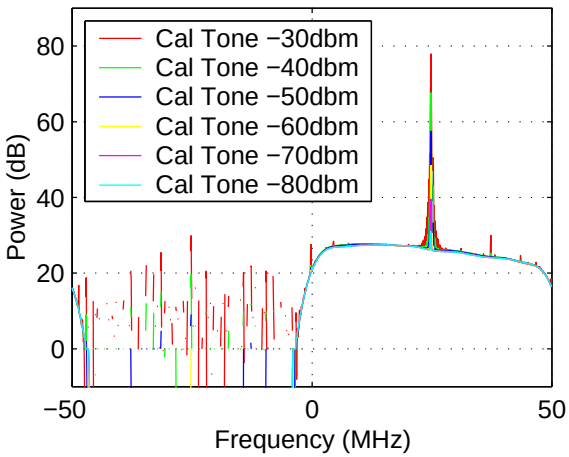


(c) Normalized Output

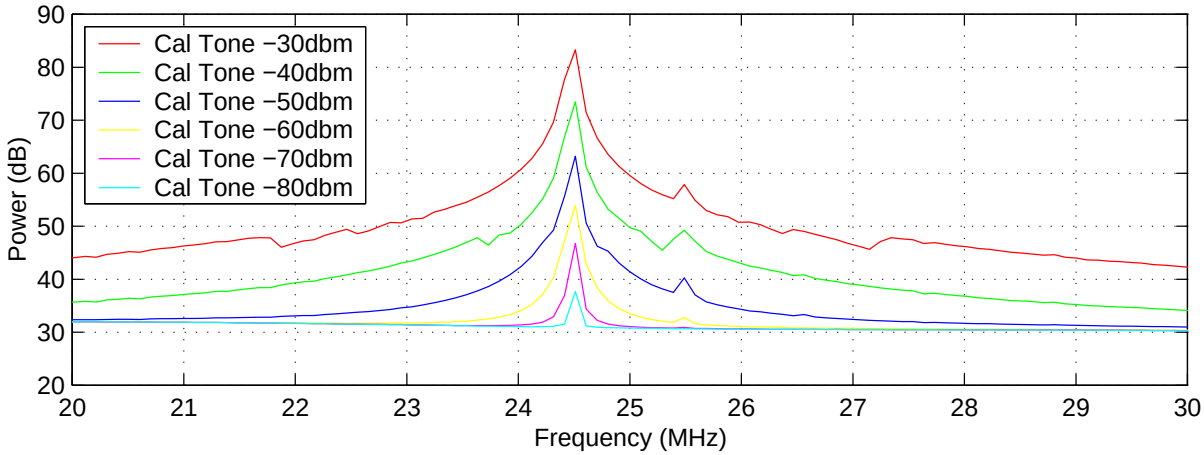
Figure 3: Results from the Spectrum Analyzer (SA) and IIP Radiometer for a 50Ω load and antenna measurements. The number of integrations used in IIP radiometer is 260096. The spectrum analyzer parameters are: $F_{start} = 100$MHz, $F_{stop} = 200$MHz, 100kHz RBW, and 100 averages.



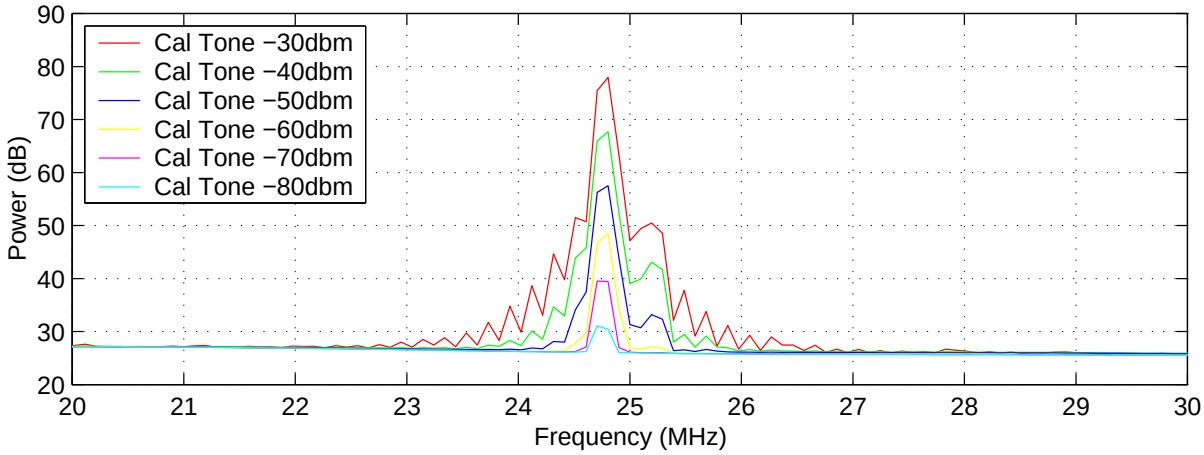
(a) Rectangular Window



(b) Bartlett Window

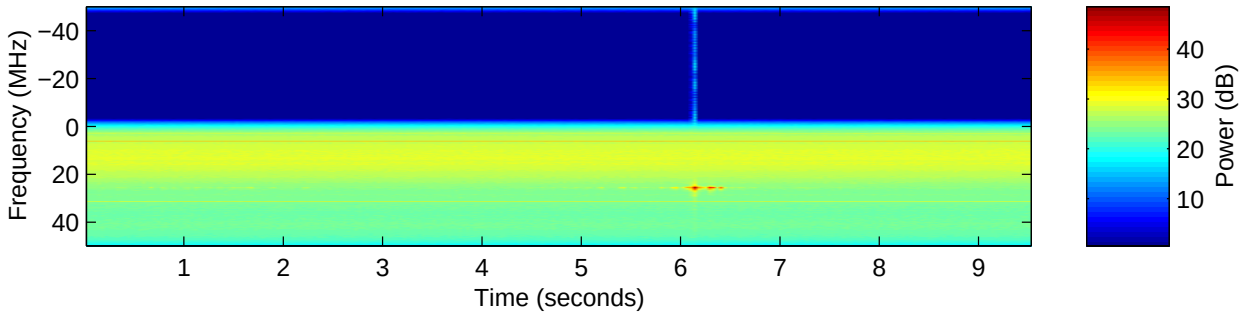


(c) Rectangular Window

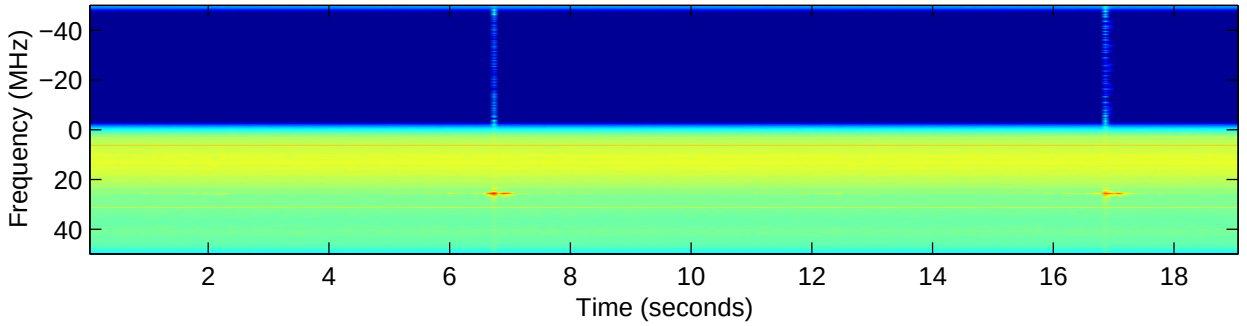


(d) Bartlett Window

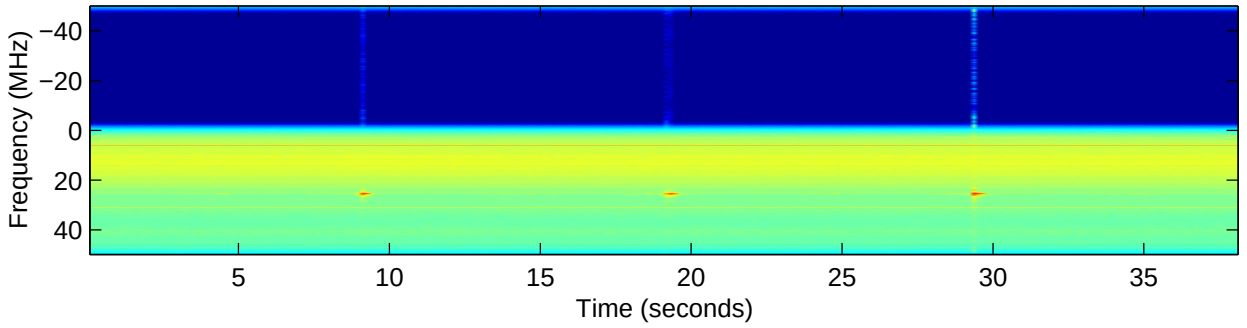
Figure 4: IIP Radiometer measurements of a calibration tone of varying power injected into the RF front end at 1331MHz. The user can choose to use Bartlett window before the hardware FFT.



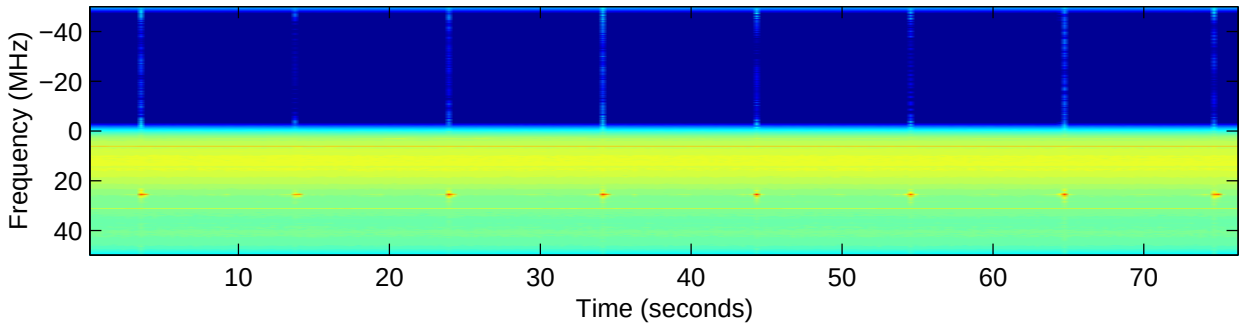
(a) 512 integrations per sample



(b) 1024 integrations per sample

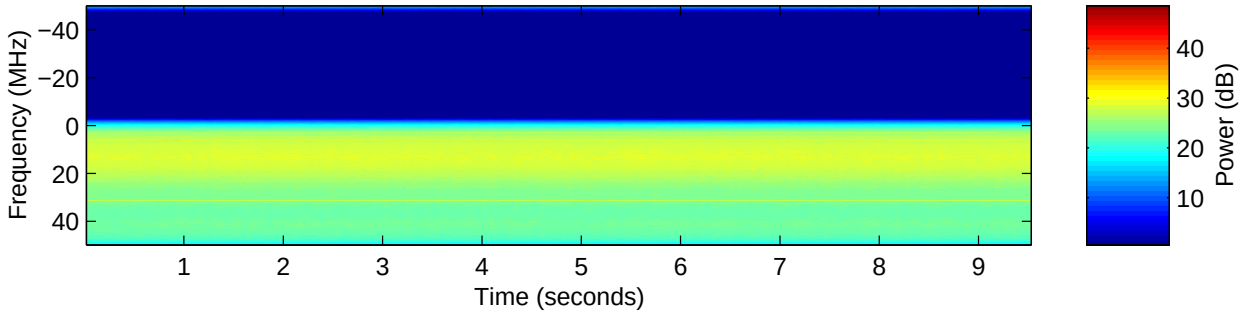


(c) 2048 integrations per sample

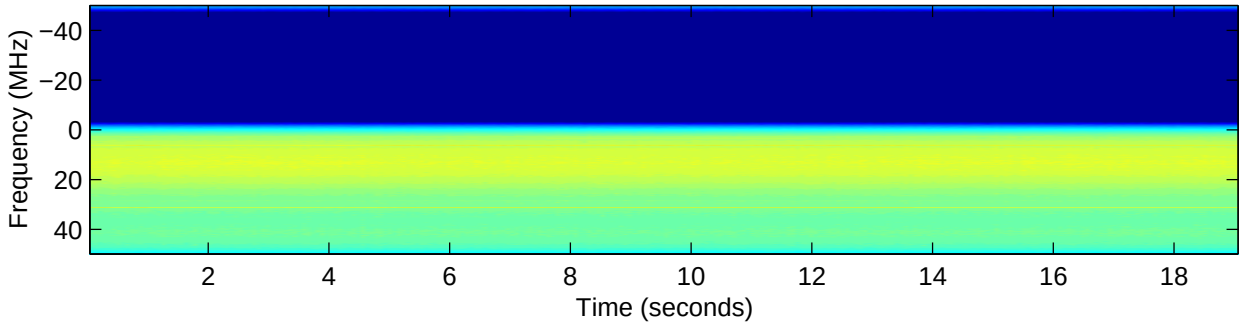


(d) 4096 integrations per sample

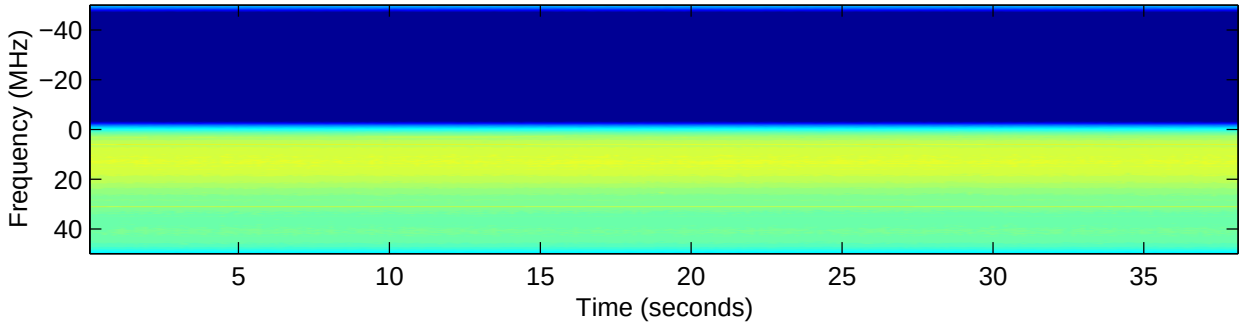
Figure 5: IIP Radiometer measurements with APB blanking disabled. ATC pulses are visible approximately every ten seconds. From (a) it is visible that the ATC radar is strong for half a second of each rotation.



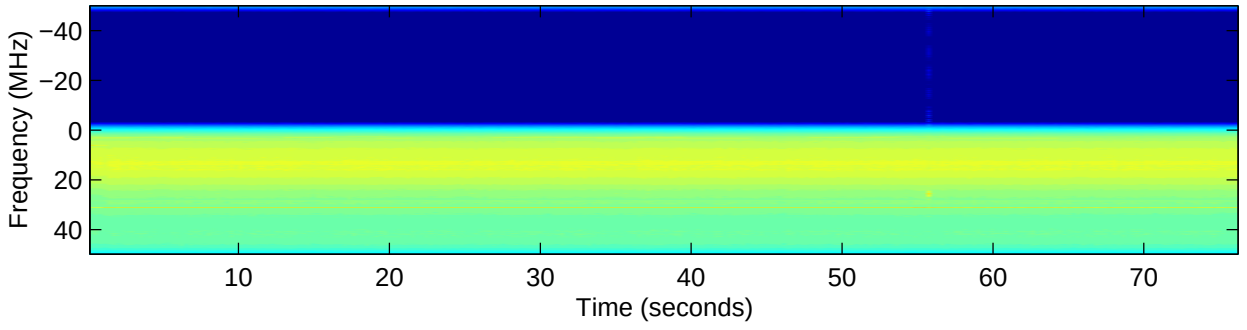
(a) 512 integrations per sample



(b) 1024 integrations per sample



(c) 2048 integrations per sample



(d) 4096 integrations per sample

Figure 6: IIP Radiometer measurements with APB blanking enabled. The blanking threshold is set to $\beta^2 = 90$ (with only one BTR implemented.) No interference is visible, except for one weak radar pulse in (d) which is due likely to a false blanker trigger.

Due to the floating point arithmetic of the FFT, when a pulse is present, the FFT exponent changes significantly causing a visible line in stop band of the Digital IF. This occurs as there is insufficient dynamic range to cover the ATC pulse and stop band completely. The shift in exponent does not effect the pass band, since it is within the FFT dynamic range.

Next, the pulse blanker was enabled and the results are presented in Figure 6. Clearly the pulse blanker is removing a majority of the ATC radar pulses. Note that Figures 6 and 5 are plotted using the same scale. Note that the output power has dropped a few tenths of a dB due to the blanking. A scheme is being developed to compensate the power loss for blanking.

5 FFT Variance

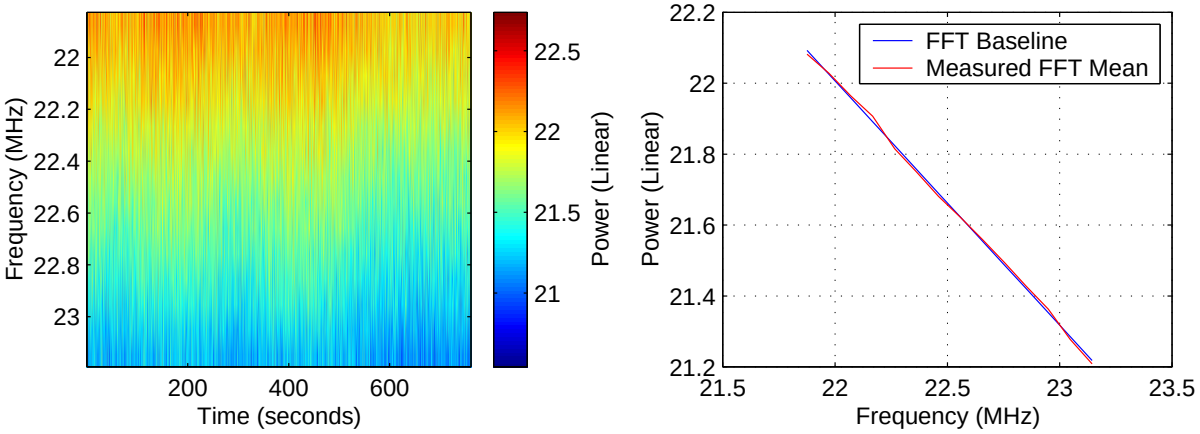
It is of significant interest to also know the stability of the IIP radiometer. This can be measured by integrating FFT results over a long time period. This is now possible since a large number of FFT results can be integrated in hardware in a relatively short amount of time.

Integration data was recorded from the system for a period of approximately 14 minutes (with a FFT duty cycle of 14% results in a total integration time of 100 seconds.) The integration length was set to 4096 FFTs, which meant that it was possible to record 10 million hardware FFT integrations during this period. These FFT integrations are shown in Figure 7(a).

In software the mean value of these FFTs was calculated (a baseline), and a 1MHz wide part of the spectrum which is relatively flat shown in Figure 7(b). In this part of the spectrum the response has a constant slope which can be approximated by a straight line.

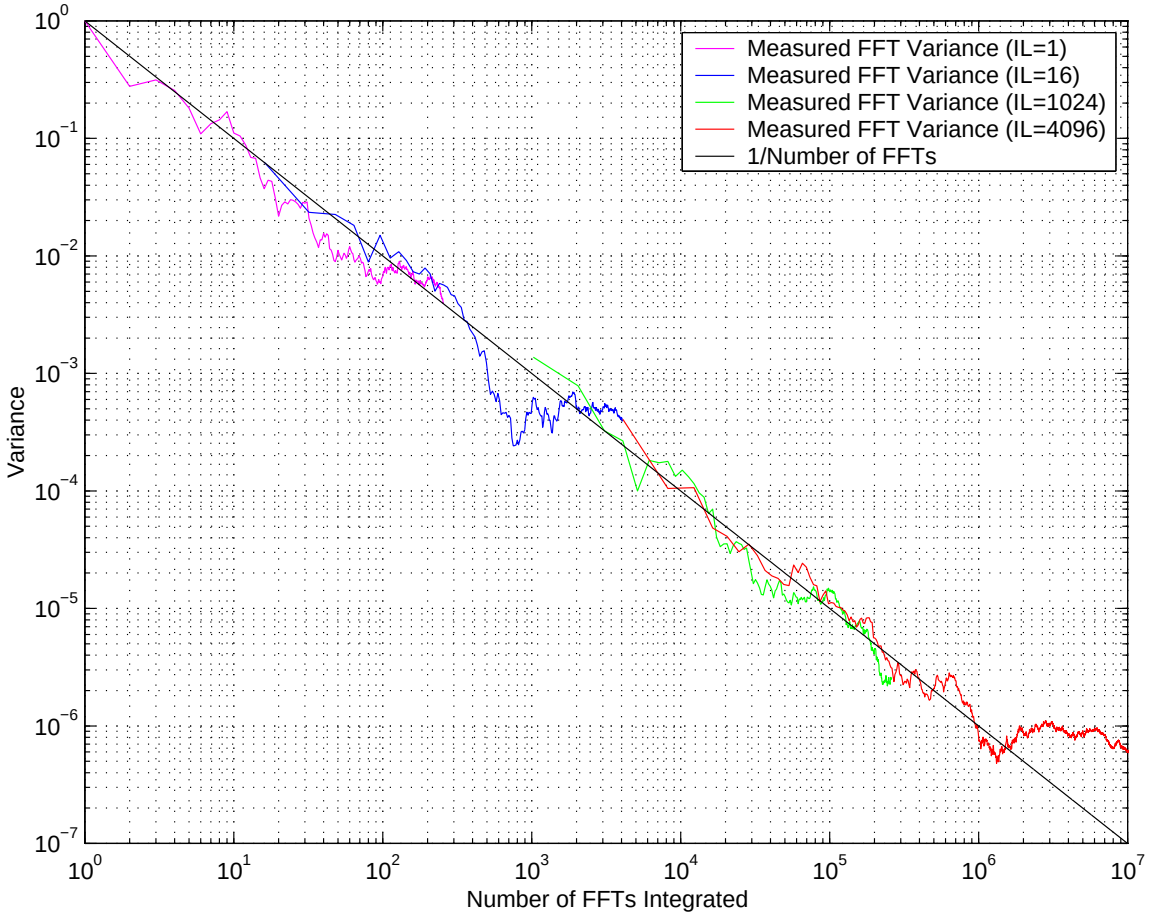
Iteratively, each hardware integration result is accumulated in software, and after each iteration the baseline is removed and the variance across these FFT bins calculated. The result of this procedure is shown in Figure 7(c). Note that the same procedure was used to estimate the variance for a smaller number of FFT integrations (< 4096.)

The variance of the measurements decreases with the number of integrations, until approximately one million FFT integrations. At this point a system limit has been reached. This is a great result for the IIP digital radiometer. It will be interesting to compare this result when the FFT processor is processing 100% of the input data.



(a) 800 seconds of measured FFT data

(b) 1MHz of spectrum which is used



(c) Measured FFT Variance

Figure 7: IIP Radiometer FFT integration measurements for a terminated system. (a) 2540 integrations results are captured over 800 seconds (although only 14% of the data is FFT'd, representing approximately 110 seconds of real integration.) (b) A 1MHz part of the spectrum can be approximated using a straight line (baseline.) (c) After removing the baseline from the measured data the FFT variance can be calculated. The variance is proportional to $1/N$, where N is the number of FFTs integrated. Various integration lengths are used to determine the variance from one FFT to 10 million FFTs. The system is very stable up to one million FFTs (10 seconds of data.)

Summary and Conclusions

This report has presented an implementation and measurement of a post-FFT spectral integration processor. Details of the processor were given, as well as source code. The spectral integrator is computationally simple, but requires significant FPGA memory to store intermediate results. The design can operate at 100MHz and consume all FFT results.

The entire IIP radiometer spectral output has been compared to a Spectrum Analyzer measurement with full success. This is a great milestone for the radiometer.

The dynamic range of the radiometer was also tested using a calibration tone. The measured 50dB dynamic headroom of the system is sufficient to accommodate a nearby ATC radar. Measurements of this radar were also shown, with and without blanking. When the blanker was activated, a majority of pulses were suppressed. This is another great result.

Finally, the overall radiometer performance/stability was measured by recording 100 seconds of integrated FFT data. From this data the system was found to be stable up to one million FFT integrations. This corresponds to a range of 3 decades (sensitivity is proportional to $\sqrt{(}$Number of FFT's) which reduces the system sensitivity (temperature) from approximately 300K to 0.3K - which is very close to the desired 0.1K measurement accuracy. This is another great result for the radiometer.

References

[1] G. A. Hampson, "Implementation of a Single FFT Processor," July 3 2002. http://esl.eng.ohio-state.edu/rfse/iip/fftimplem.pdf.

[2] G. A. Hampson, "A Post-FFT Power Estimator Implementation," April 17 2002. http://esl.eng.ohio-state.edu/rfse/iip/i2q2proc.pdf.

[3] S. W. Ellingson, "Design Concept for the IIP Radiometer RFI Processor," January 23 2002. http://esl.eng.ohio-state.edu/rfse/iip/rfiproc1.pdf.

[4] S. W. Ellingson and G. A. Hampson, "On-Air Test of the IIP Receiver Using Observations of an ATC Radar," June 29 2002. http://esl.eng.ohio-state.edu/rfse/iip/test020618.pdf.

Appendix A: Integration Stage AHDL Source Code

```
-- Single Integration Stage (up to 1024 integrations)
-- Grant Hampson 25 July 2002

include "lpm_add_sub.inc";   -- include file headers
include "lpm_ram_dp";
include "lpm_mux";

PARAMETERS (N=32,            -- width of data input
            M=12);           -- growth due to integrations

SUBDESIGN integ_stage
(
   clk,                      -- system clock
   clken_in,                 -- clock enable (start of data)
   integrations[3..0],       -- Number of integrations (log2)
   data[(N-1)..0]            -- input data
      : INPUT;
   clken_out,                -- data out valid
   integ_data[(N-1)..0]      -- integrated data
      : OUTPUT;
)

VARIABLE
   integ : lpm_add_sub WITH(LPM_WIDTH = N+M,
                            LPM_REPRESENTATION = "UNSIGNED",
                            LPM_DIRECTION = "ADD",
                            LPM_PIPELINE = 3);

   integ_RAM : lpm_ram_dp WITH(LPM_WIDTH = N+M,
                               LPM_WIDTHAD = 10,
                               LPM_INDATA = "REGISTERED",
                               LPM_OUTDATA = "REGISTERED",
                               LPM_RDADDRESS_CONTROL = "REGISTERED",
                               LPM_WRADDRESS_CONTROL = "REGISTERED");

   integ_mux : lpm_mux WITH (LPM_WIDTH = 1,
                             LPM_SIZE = M+1,
                             LPM_WIDTHS = 4);

   integ_shift : lpm_mux WITH (LPM_WIDTH = N,
                               LPM_SIZE = M+1,
                               LPM_WIDTHS = 4);

   delay[6..0][(N-1)..0],      -- delay registers
   integ_reg[(N+M-1)..0],      -- integration register before RAM
   integshift_reg[(N-1)..0],   -- output of shifter
   scaled_reg[(N-1)..0],       -- final output register
   read_address[9..0],         -- read address for dual port RAM
   write_address[9..0],        -- write address for dual port RAM
   integ_sm_out[5..0],         -- integration state machine outputs
   integ_sm[3..0]     : DFF;   -- integration state machine

   integ_count[M..0] : DFFE;   -- integration counter M+1-bits
   integ_sm_in[3..0] : NODE;   -- integration state machine inputs
```

```
BEGIN
   delay[][].clk = clk;                        -- Connection of Input delay
   delay[0][].d = data[];
   delay[1][].d = delay[0][];
   delay[2][].d = delay[1][];
   delay[3][].d = delay[2][];
   delay[4][].d = delay[3][];
   delay[5][].d = delay[4][];
   delay[6][].d = delay[5][];

   integ.dataa[(N-1)..0] = delay[2][];         -- Connection of Integrator
   integ.dataa[(N+M-1)..N] = GND;
   integ.datab[] = integ_RAM.q[];
   integ.clock = clk;
   integ_reg[].d = integ.result[];
   integ_reg[].clk = clk;

   if integ_count[] == 0 then                  -- Connection of RAM input mux
      integ_RAM.data[(N+M-1)..N] = GND;
      integ_RAM.data[(N-1)..0] = delay[6][];
   else
      integ_RAM.data[] = integ_reg[];
   end if;

   integ_RAM.rdclock = clk;                    -- Connection of Dual Port RAM
   integ_RAM.wrclock = clk;
   integ_RAM.wraddress[] = write_address[].q;
   integ_RAM.rdaddress[] = read_address[].q;

   read_address[].clk = clk;                   -- Connection of FFT-bin counters
   read_address[].d = read_address[].q + 1;
   write_address[].clk = clk;
   write_address[].d = write_address[].q + 1;

   integ_count[].clk = clk;                    -- Connection of Integration counter
   integ_count[].d = integ_count[].q + 1;
   integ_mux.sel[] = integrations[];
   integ_mux.data[M..0][0] = integ_count[M..0];

   integ_shift.sel[] = integrations[];         -- Connection of Output Scaler
   for i in 0 to M generate
      integ_shift.data[i][] = integ_RAM.q[(N-1+i)..i];
   end generate;
   integshift_reg[].clk = clk;
   integshift_reg[].d = integ_shift.result[];

   if integ_sm_out[5] == 0 then                -- Connection of output mux
      scaled_reg[].d = integshift_reg[];
   else
      scaled_reg[].d = B"10000000000000001000000000000000";
   end if;
   scaled_reg[].clk = clk;
   integ_data[] = scaled_reg[];
```

```
   integ_sm_in[0] = clken_in;                -- Connection of Controller inputs
   integ_sm_in[1] = (write_address[] == B"1111111111");
   integ_sm_in[2] = integ_mux.result[0];
   integ_sm_in[3] = (read_address[] == B"1111111111");

   read_address[].clrn = integ_sm_out[0];    -- Connection of Controller inputs
   write_address[].clrn = integ_sm_out[1];
   integ_RAM.wren = integ_sm_out[2];
   integ_count[].ena = integ_sm_out[3];
   integ_count[].clrn = integ_sm_out[4];
   clken_out = integ_sm_out[4];

   TABLE
     integ_sm[].q, integ_sm_in[] => integ_sm[].d, integ_sm_out[].d;
       B"0000",       B"XXX1"    =>   B"0000",      B"110000"; -- wait for clken_in
       B"0000",       B"XXX0"    =>   B"0001",      B"110001"; -- data arriving, read RAM
       B"0001",       B"XXXX"    =>   B"0010",      B"110001"; -- through 2nd delay reg
       B"0010",       B"XXXX"    =>   B"0011",      B"110001"; -- through adder P=1
       B"0011",       B"XXXX"    =>   B"0100",      B"110001"; -- through adder P=2
       B"0100",       B"XXXX"    =>   B"0101",      B"110001"; -- through adder P=3
       B"0101",       B"XXXX"    =>   B"0110",      B"110001"; -- output register
       B"0110",       B"XX0X"    =>   B"0110",      B"110111"; -- write RAM 1024 writes
       B"0110",       B"XX1X"    =>   B"0111",      B"111000"; -- increment integ counter
       B"0111",       B"XXXX"    =>   B"1000",      B"110000"; -- wait for increment
       B"1000",       B"X0XX"    =>   B"0000",      B"110000"; -- more integrations to go
       B"1000",       B"X1XX"    =>   B"1001",      B"110000"; -- finished integrating
       B"1001",       B"XXXX"    =>   B"1010",      B"110001"; -- start read counter
       B"1010",       B"XXXX"    =>   B"1011",      B"110001"; -- wait for pipeline
       B"1011",       B"XXXX"    =>   B"1100",      B"110001"; -- finished, write marker
       B"1100",       B"0XXX"    =>   B"1100",      B"000001"; -- wait for 1024 reads
       B"1100",       B"1XXX"    =>   B"1101",      B"000000"; -- integration done
       B"1101",       B"XXXX"    =>   B"1110",      B"000000"; -- wait for
       B"1110",       B"XXXX"    =>   B"1111",      B"000000"; -- pipeline
       B"1111",       B"XXXX"    =>   B"0000",      B"100000"; -- to finish
   END TABLE;

   integ_sm[].clk = clk;
   integ_sm_out[].clk = clk;
END;
```