Implementation Results of a Windowed FFT

Grant Hampson

July 12, 2002

Introduction

This document describes a FPGA implementation and results of the FFT window component of the IIP Radiometer RFI processor described in [1]. The FFT component has been previously described in [2]. A spectral window is required to reduce RFI side-lobe leakage into adjacent bins of the FFT. This document describes the window hardware and some implementation results.

1 Windowing FPGA Hardware

Figure 1 illustrates the hardware required to implement a windowing function. The window ROM contains 1024 words, each 10-bits wide. (The ROM could be changed to RAM at a later date if the window is required to change in real-time.) The window ROM block contains an address counter which cycles through the window locations sequentially. (For symmetrical windows it could be possible to have a ROM half the length.)

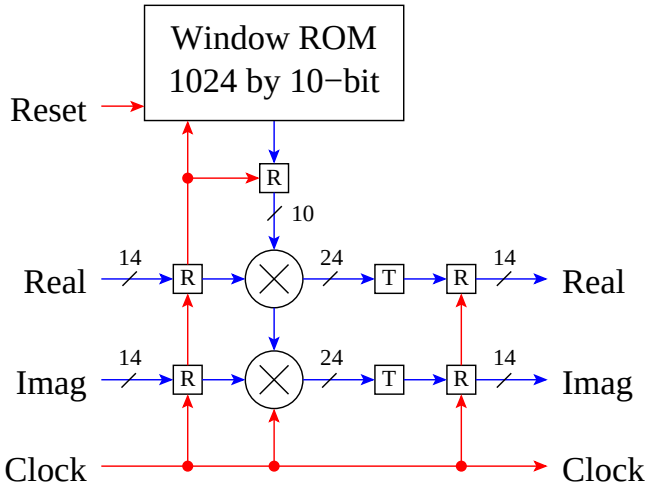

Figure 1: The hardware for the window consists of a ROM and two multipliers. 'R' denotes registers and 'T' denotes truncation. The window ROM contains an address counter which accesses each window value. The *reset* signal zeros this address. The multipliers are pipelined due to the fact that they operate at 100MSPS. The ROM is implemented inside the FPGA. Control signals are shown in red and data paths in blue.

A Bartlett (triangular) window has been tested in this report. Given that the FFT length is 1024, the ROM contains the numbers 0 to 511 (step 1) and 511 to 0 (step -1). Since signed multipliers are used, a 10-bit window value is required with the sign bit is set to 0. Note that the Bartlett window could be simply implemented using an up/down counter, saving significant FPGA resources. For other windows a ROM is required.

Appendix A contains the AHDL source code to the window function. The window function consumes 436 logic elements and 10 ESB bits. The maximum operating frequency of the circuit is 123MHz. Appendix B lists the Matlab code to generate the window.

2 Windowing Implementation Results

The windowing function was integrated with the single FFT processor described in [3]. In this design a state machine controls the sequencing of the window reset and the FFT write/start/done signals. The results of the implementation are shown in Figure 2. Three sets of data were captured within minutes of each other. The first test captured enough raw data to perform 25600 integrations in Matlab. The second and third are FFTs (with

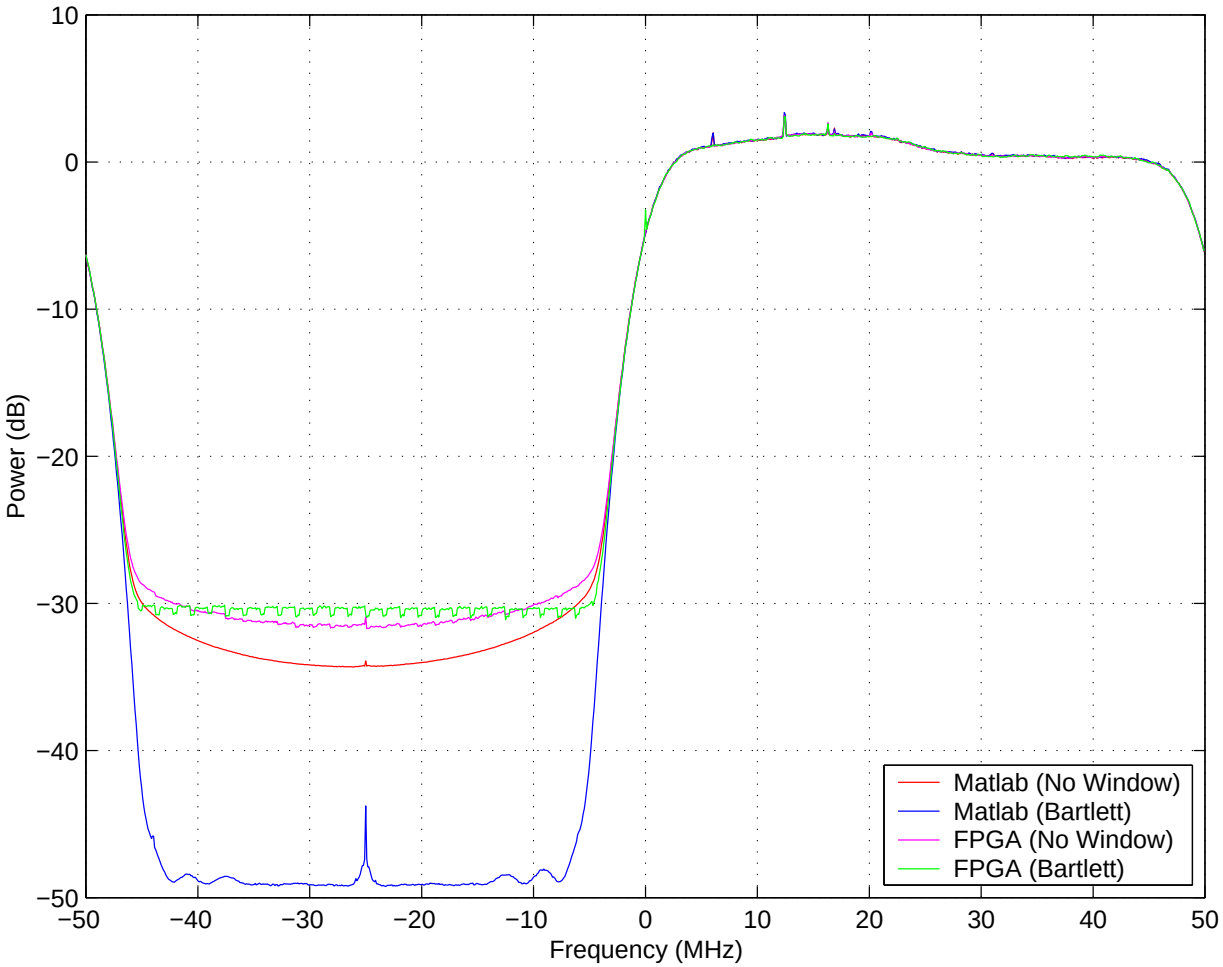

Figure 2: Implementation results of the FFT processor (with and without a Bartlett window.) The FPGA results are compared to 14-bit Matlab FFTs of captured raw data. Each trace is 25600 FFT integrations. The results are normalized in the pass-band.

and without a window) integrated in LabWindows. For the case where no windows are used the Matlab and FPGA results are within $2.5dB$ of each other in the stop-band of the digital IF filter [4]. In the pass-band the results are almost identical.

For the case where a Bartlett window is applied, the Matlab and FFT results differ by almost 20dB in the stop band. This is mainly due to the limited dynamic range of the FPGA FFT output. The FPGA output in the stop band is flat due to the equal distribution of quantisation noise. The FPGA FFT with window results have a steeper transition zone suggesting that the window function is working correctly.

Summary and Conclusions

This report has illustrated a possible implementation of a windowing function for the FFT processor. The window can be implemented in many different ways depending on whether coefficients need to be updated in real time, fixed in a ROM, and the type of window function.

Implementation results were also presented for verification. These results indicate that that the windowing operation is functioning correctly.

References

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

[2] G. A. Hampson, "A Possible 100MSPS Altera FPGA FFT Processor," March 12 2002. http://esl.eng.ohio-state.edu/rfse/iip/fftproc.pdf.

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

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

Appendix A: Window AHDL Source Code

```
-- Windowing of FFT input data
-- Grant Hampson 11 July 2002

include "lpm_mult.inc";
include "lpm_rom.inc";

PARAMETERS (N=14);   -- width of data inputs

SUBDESIGN Window
(
   clk,
   reset,
   realin[(N-1)..0],
   imagin[(N-1)..0]
      : INPUT;

   realout[(N-1)..0],
   imagout[(N-1)..0]
      : OUTPUT;
)

VARIABLE
   windowrom : lpm_rom WITH (lpm_width = 10,
                             lpm_widthad = 10,
                             lpm_numwords = 1024,
                             lpm_file = "window.mif",
                             lpm_address_control = "unregistered",
                             lpm_outdata = "unregistered");

   multr, multi : lpm_mult WITH(LPM_WIDTHA = 10,
                                LPM_WIDTHB = N,
                                LPM_WIDTHP = N+10,
                                LPM_WIDTHS = N+10,
                                LPM_REPRESENTATION = "SIGNED",
                                LPM_PIPELINE = 2);

   realin_reg[(N-1)..0],      -- register for real input
   imagin_reg[(N-1)..0],      -- register for imaginary input
   win_cntr[9..0],            -- counter for window value
   win_reg[9..0],             -- register for window value
   multr_reg[(N-1)..0],       -- register for windowed real component
   multi_reg[(N-1)..0] : DFF; -- register for windowed imaginary component

BEGIN
   realin_reg[].clk = clk;    -- connect input registers
   imagin_reg[].clk = clk;
   realin_reg[].d = realin[];
   imagin_reg[].d = imagin[];

   win_cntr[9..0].clk = clk;  -- connection of window ROM
   win_cntr[9..0].clrn = reset;
   win_cntr[9..0].d = win_cntr[9..0].q + 1;
   windowrom.address[] = win_cntr[9..0].q;
   win_reg[].clk = clk;
```

```
   win_reg[] = windowrom.q[];

   multr.clock = clk;         -- Connect up window multiplier
   multi.clock = clk;
   multr.dataa[] = win_reg[];
   multi.dataa[] = win_reg[];
   multr.datab[] = realin_reg[];
   multi.datab[] = imagin_reg[];

   multr_reg[].clk = clk;     -- connection of output registers
   multi_reg[].clk = clk;
   multr_reg[].d = multr.result[22..9]; -- Multiply result by 2 due to window
   multi_reg[].d = multi.result[22..9];
   realout[] = multr_reg[];
   imagout[] = multi_reg[];
END;
```

Appendix B: Matlab Window Source Code

```
% Script to generate Window ROM contents
% Grant Hampson 11 July 2002

clear all

N = 1024;                     % length of window
B = 10;                       % number of bits (including sign bit)
w = bartlett(N);              % window type
w = w/max(w);                 % normalise 0 to 1
w = round(w.*((2^(B-1))-1));  % make it a B-bit positive integer

plot(1:N,w),grid
xlabel('Window Index'), ylabel('Window Value')

filename = 'window.mif';                % output filename check window.tdf
fid = fopen(filename,'wt');
fprintf(fid,'DEPTH = %d;\n',N);
fprintf(fid,'WIDTH = %d;\n',B);
fprintf(fid,'ADDRESS_RADIX = DEC;\n');
fprintf(fid,'DATA_RADIX = HEX;\n');
fprintf(fid,'CONTENT\n');
fprintf(fid,'BEGIN\n');
for i=1:N
   fprintf(fid,'%d:%s;\n',i-1,dec2hex(w(i),4));
end
fprintf(fid,'END;\n');
fclose(fid)
```