A Post-FFT Power Estimator Implementation

Grant Hampson

April 17, 2002

Introduction

This document describes an Altera FPGA design and simulation of a Post-FFT Power Estimator (PFPE) component of the IIP Radiometer RFI processor described in [1]. The PFPE will be capable of processing 100% of the input bandwidth (100 MSPS.) The output of the PFPE is calculated by

$$\text{Power (per FFT bin)} = i^2 + q^2 \tag{1}$$

This document describes a possible implementation in an Altera FPGA, including code and simulated performance.

1 PFPE Implementation

Figure 1 illustrates a possible implementation of Equation 1. The AHDL code (Altera Hardware Description Language) which implements this block diagram is listed in Appendix A. Multiplier and adder cores are freely available from Altera [2] and the PFPE structure is easily implemented. The cores are parametized and it is possible to increase or decrease the amount of pipelining in order to speed up/down the operational frequency. Thus a clock input is shown in the block diagram for the multiplier and adder.

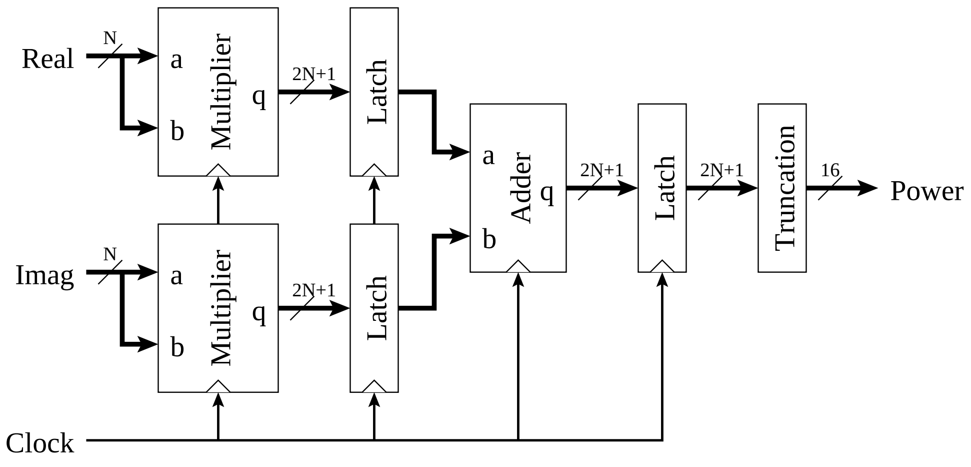
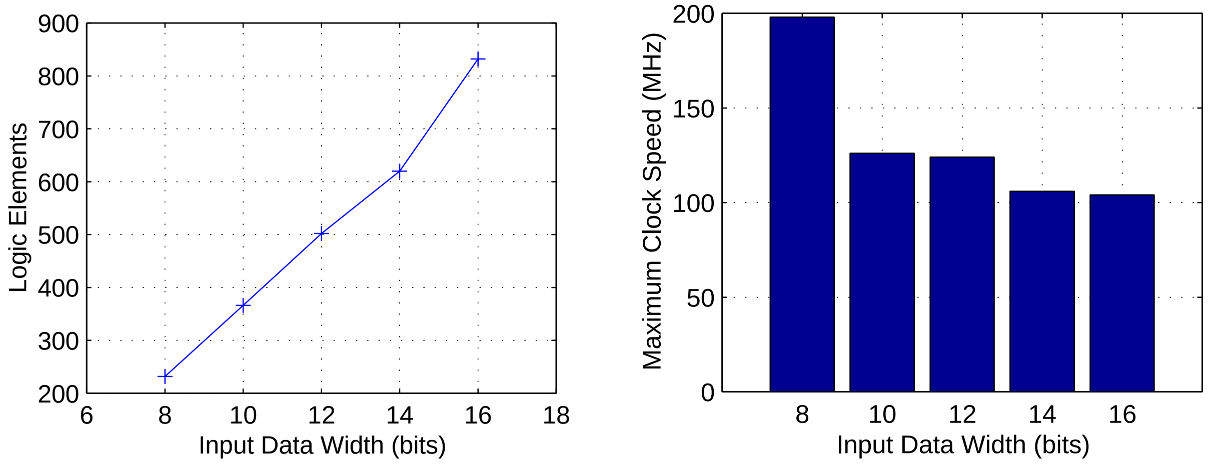


Figure 1: Block diagram of the Post-FFT Power Estimator. Input width (N) can be from 8 to 16-bit. Addition is done at full precision. Output data is truncated to 16-bits.

Operation of the PFPE is as follows: data from the FFT processor [3] flows in a word serial format, i.e., each bin of the FFT arrives on each rising edge of the clock. The PFPE is capable of producing a new result every clock cycle, even though there is a latency of four clock cycles to compute each result (assuming one level of pipelining in the multiplier and adder.) Since the FFT produces data at 100 MSPS - the PFPE must also be capable of processing 100 MSPS.

The reader should note that when two N-bit numbers a multiplied the result is $(2N+1)$-bits wide. For example, using 12-bit outputs from the FFT results in the power of each bin being 25-bits wide. To limit the result to an acceptable width - the result is truncated to 16-bits. If more dynamic range (>100dB) is required this could be expanded.

Figure 2 illustrates some performance statistics of the PFPE. Here one level of pipelining is used in the multiplier and adder. The number of logic elements is almost directly proportional to the number of input bits. The maximum clock speed naturally decreases as the number of input bits is increased. Additional pipelining levels could be used to increase the operational speed (as well as increasing the overall size.)

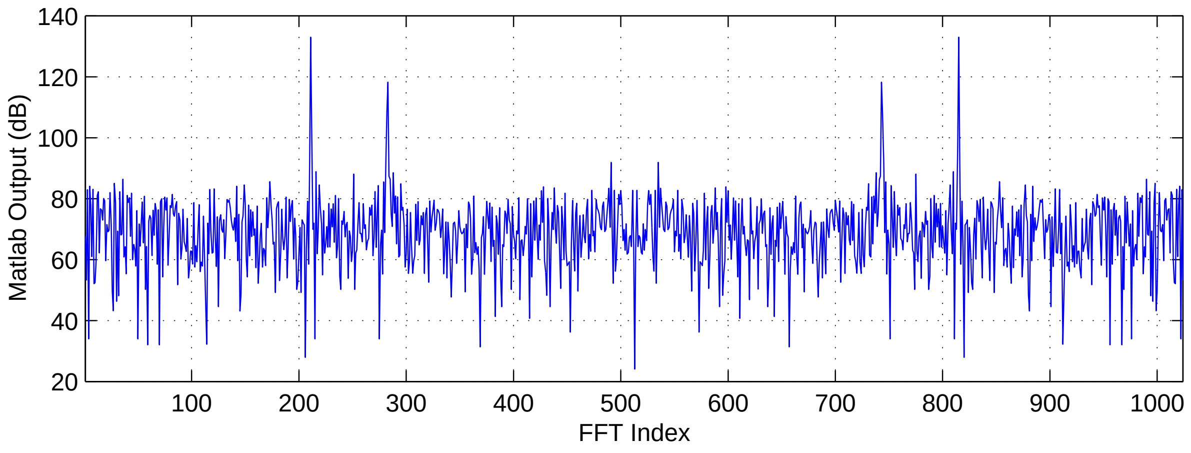


(a) FPGA Size in Logic Elements (LE) (b) Operating Frequency

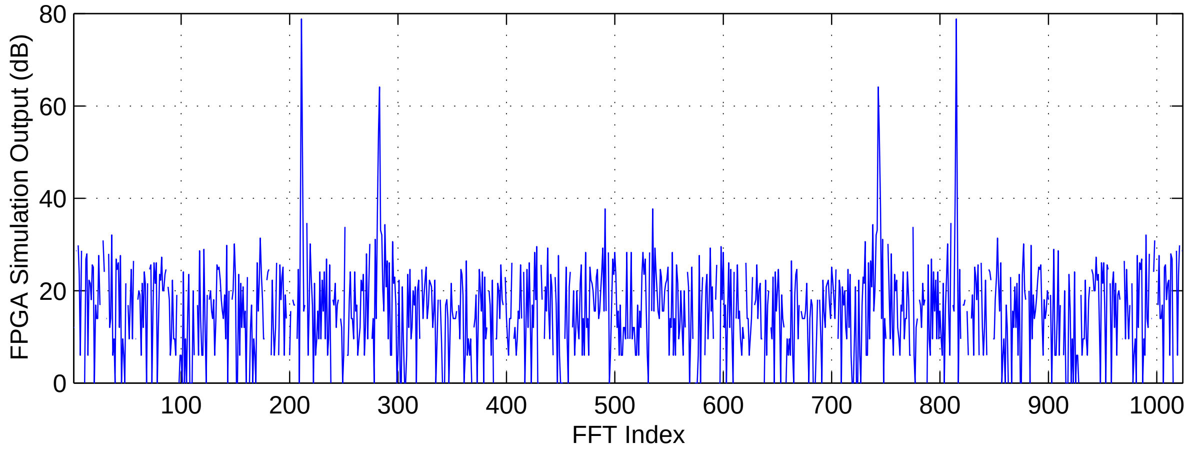
Figure 2: Size and performance of the PFPE. The clock performance is based on using an EP20K100EQC208-1 device, and one level of pipelining in the multiplier and adder. Increasing the pipelining in the multiplier to 4 and in the adder to 2 results in a design of 960 logic elements operating at 134MHz.

2 PFPE Simulation

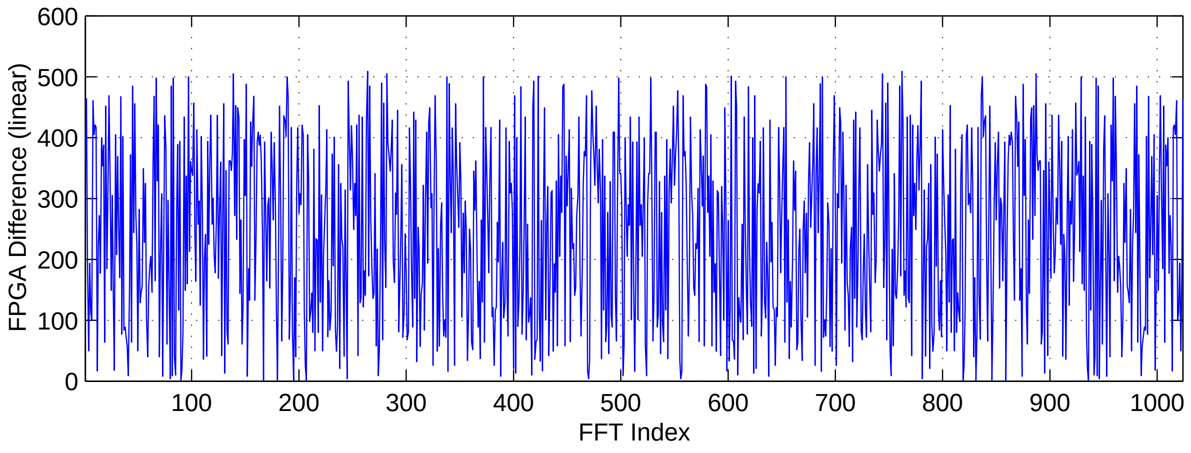
The PFPE was also simulated in Quartus to verify correct operation. Figure 3 illustrates the results of this simulation. The stimuli for this simulation is the FFT of two tones plus noise. The FFT was calculated in Matlab and quantised to 12-bits. Figure 3(a&b) show the results of applying Equation 1 in Matlab and in a FPGA. Due to truncation of the FPGA output the result is 512 times smaller than the Matlab result. Since the input width is 12-bit, the total output width is 25-bits, which is truncated 9-bits ($2^9 = 512$) to a 16-bit output. The difference of the two after scaling is within 0 to 512 indicating correct operation.



(a) Matlab Simulation



(b) FPGA Simulation



(c) Difference (after scaling FPGA result by $2^9 = 512$)

Figure 3: Simulations of the PFPE in (a) Matlab and in (b) Quartus. The simulations are based on using 12-bit inputs. Only 16-bits of the possible 25-bit output is used which introduces a scale factor, which is 9-bits. After taking this scale factor into account the remaining error is less than 2^9, as shown in (c).

Summary and Conclusions

This report has presented a possible 100 MSPS Altera FPGA Post-FFT Power Estimator processor implementation. The processor can be built from free standard Altera library functions. Simulations of the processor indicate that clock speeds in excess of 100MHz are achievable. The number of required FPGA logic elements is within bounds and will occupy less than 25% of a $140 FPGA (EP20K100EQC208-1).

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] *LPM (Library of Parametised Modules) Quick Reference Guide*, Altera Corporation, December 1996. http://www.altera.com/literature/catalogs/lpm.pdf.

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

Appendix A: AHDL Source Code

```
-- I^2 + Q^2 power estimator for FFT
-- Grant Hampson 16 April 2002

INCLUDE "lpm_mult.inc";
INCLUDE "lpm_add_sub.inc";
PARAMETERS (N = 12);   -- input data width (connect to MSB's)

SUBDESIGN i2q2proc
(
   clock,           -- clock/control bits
   real[15..0],     -- Imaginary data bus in (i)
   imag[15..0]      -- Real data bus in (q)
      :INPUT;

   abspower[15..0]  -- i^2+q^2 output bus
      :OUTPUT;
)

VARIABLE
   imult, qmult : lpm_mult WITH(LPM_WIDTHA = N,
                                LPM_WIDTHB = N,
                                LPM_WIDTHP = N+N+1,
                                LPM_WIDTHS = N+N+1,
                                LPM_REPRESENTATION = "SIGNED",
                                LPM_PIPELINE = 1);

   iqadder : lpm_add_sub WITH(LPM_WIDTH = N+N+1,
                              LPM_REPRESENTATION = "SIGNED",
                              LPM_DIRECTION = "ADD",
                              LPM_PIPELINE = 1);

   imult_out[(N+N)..0], qmult_out[(N+N)..0], iqadd[(N+N)..0] : DFF;

BEGIN
   imult.dataa[(N-1)..0] = real[15..(16-N)];  -- Connect MSBs of input real
   imult.datab[(N-1)..0] = real[15..(16-N)];
   imult.clock = clock;
   imult_out[].d = imult.result[];            -- result is i^2
   imult_out[].clk = clock;

   qmult.dataa[(N-1)..0] = imag[15..(16-N)];  -- Connect MSBs of input imag
   qmult.datab[(N-1)..0] = imag[15..(16-N)];
   qmult.clock = clock;
   qmult_out[].d = qmult.result[];            -- result is q^2
   qmult_out[].clk = clock;

   iqadder.dataa[] = imult_out[];             -- connection of adder
   iqadder.datab[] = qmult_out[];
   iqadder.clock = clock;
   iqadd[] = iqadder.result[];                -- result is i^2 + q^2
   iqadd[].clk = clock;

   abspower[15..0] = iqadd[(N+N)..(N+N-15)];  -- take 16 MSB's of result
END;
```