SMA Wideband Correlator: Memo 1 Polyphase Filter-bank Utilization

Rurik A. Primiani
Harvard-Smithsonian
Center for Astrophysics

Jonathan Weintroub
Harvard-Smithsonian
Center for Astrophysics

Jonathan deWerd
Harvard University

***Abstract*—We study the applicability of a packetized FX correlator architecture to the Submillimeter Array (SMA) wideband-correlator upgrade. This uses FPGA processors with a packetized corner turner made from a commercial off-the-shelf 10 Gbit/s network switch placed between the F-engines and the X-engines. We estimate quantitatively the resource utilization in the F-engine's polyphase filter-bank under a variety of assumptions. In summary, we find that the correlated bandwidth dominates FPGA multiplier and adder utilization, while spectral resolution drives memory needs.**

I. Introduction

The SMA is expanding its bandwidth to an 18 GHz system with two simultaneous receivers, one for each orthogonal polarization. A new wideband high spectral resolution correlator is needed. The requirements call for uniform spectral resolution of 100 to 200 kHz over the full band. At 28 baselines per receiver and full Stokes capability, the total number of baselines becomes effectively 112.

The enormous computational requirements of such a correlator demand high-bandwidth and highly parallel signal processing engines, of which the Field Programmable Gate Array (FPGA) seems to be the most appropriate. In particular, the printed circuit boards provided by CASPER[1] along with the frequency domain (FX) architecture favored by that collaboration have been chosen as the most viable design model.

A general trend among the community for large correlators involves processing baselines for many antennas, $\gg 8$, the so-called "large-N" regime. The future SMA upgrade, however, with a modest antenna number but large bandwidth can be said to lie in a "large-D", or large-demux, regime (see section III). This is a hitherto unexplored area and this memo (in fact, this memo series) intends to explore the resource requirements of the PFB which is vital to understanding the capabilities of various DSP engines.

We will refer from time to time to the Xilinx Virtex 6 SX475T FPGA. This device is used on the CASPER ROACH 2 processor and is used here as a point of reference. Subsequent memos will detail the expected utilization and processing limits of particular FPGAs including the V6-SX475T.

II. Sampling the IF

Currently no practical 40 GSa/s ADC exists that can sample the full analog bandwidth of 18 GHz in one channel. Thus the IF must be downconverted and filtered into sub-bands, or blocks of bandwidth, with Nyquist bandwidths matching those of the candidate samplers. For example, a 20 GSa/s sampler would sample two 9 GHz bands while a 5 Gsa/s ADC maps to nine 2 GHz bands. Other considered scenarios include three 6 GHz bands (14 GSa/s), and four 4.5 GHz bands (10 GSa/s). Keep in mind that the number of bands must be multiplied by two receivers per antenna, and by eight antennas. The resulting block downconverter becomes large and expensive if there are many bands, and thus the motivation to sample and digitally process data as fast as possible.

III. Demultiplex factor

Though the candidate samplers operate at rates between 5-20 GSa/s (f_{sample}) typical FPGAs are limited to rates of a few hundred MHz (f_{FPGA}). This implies that the proposed correlator will be parallelized by a factor of f_{sample}/f_{FPGA}. This ratio is the net demultiplex (or demux) factor D.

Because the downstream FFT implementation is radix-2, it is strongly preferred that the demux factor also be a power of 2. For example, for 20 GSa/s. a demux factor of 128 (2^7) or for 5 GSa/s a demux factor of 32 (2^5) translate to an FPGA clock rate of 156 MHz. This is readily achievable but it could be argued that such a sedate processing rate under-utilizes the power of the FPGA. A FPGA clock rate of 312 MHz would halve the demux factor.

Experience shows that clocking an FPGA with a complex bitcode at rates approaching or exceeding 300 MHz stretches its capabilities, and those of the design tool-flow, to meet timing. Were 312 MHz achievable, however, our resource calculation will show that a very substantial savings in multiplier and adder resources results. Without constraint it would perhaps be preferred to clock the FPGA at about 250 MHz, however because the demux factors are quantized to radix-2 numbers, it is important to appreciate that stretching to the next demux boundary can yield significant returns in utilization.

IV. PFB utilization

The standard CASPER wideband-correlator approach is a polyphase filter bank (PFB or F-engine) followed by single-lag correlators in each channel (X-engines), with a 10 Gbit/s network switch implementing the corner turn from antenna based F-engine processing to the baselines for the X-engines. A PFB can be constructed using an FIR filter followed by a

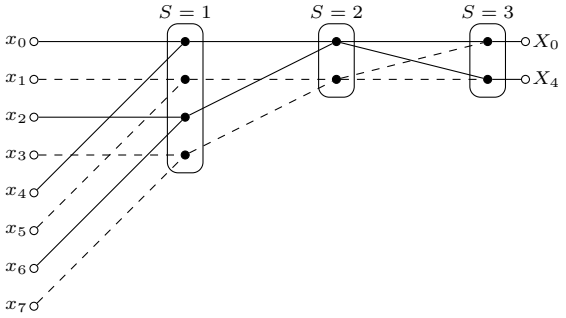
[1]Collaboration for Astronomy Signal Processing and Electronics Research. https://casper.berkeley.edu

DFT which extracts the appropriate subbands. The DFT can be implemented using a FFT algorithm in order to take advantage of the $O(N \log N)$ optimization those algorithms afford. However as bandwidth, and therefore demux, grows, more samples are presented at once which means more multipliers must be instantiated in hardware.

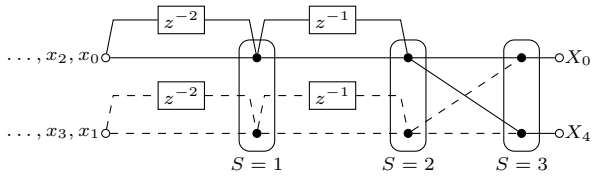
Consider Fig. 1a which shows a fully parallel butterfly diagram where a total of $N = 8$ samples at a time are presented to the FFT. If however only 2 samples can be processed at a time, i.e. $D = 2$, then the butterflies of the first 2 stages must buffer up the samples they need, while the last stage can process on the partial products directly available to it. In general the first $\log_2 \frac{N}{D}$ stages require buffering and can be called "pipelined," while the last $\log_2 D$ stages do not and are thus "direct." Conceptually these sets of stages can be thought of as separate partial FFTs that form a large FFT, there being D FFTs of size N/D followed by a single FFT of size D.

Within the pipelined stages in Fig. 1b the butterfly operators are used only half the time since they must wait for the buffered samples to form the pairs they need, this would result in an under-utilized multiplier. To get around this these butterflies can be multiplexed such that they process two streams at once, fully utilizing the multipliers at all times. The circled butterflies in stages 1 and 2 of Fig. 1b would thus only use a single complex multiplier, or in the case of real inputs, 4 real multipliers. Fig. 2 shows the computation inside a single so-called "biplex" FFT, essentially a fully-utilized butterfly. (See also the discussion in [1].)

The CASPER FFT implementation uses the above approach. It assumes that $N > D$ and that both N and D are powers of 2. It's important to note that the pipelined stages are presented



(a) Parallel butterfly implementation with N=8, showing only the butterflies needed to produce outputs $X(0)$ and $X(4)$.



(b) Pipelined approach for D=2 case, also showing only those butterflies needed for the same two output bins.

Fig. 1. A simplified diagram of a fully parallel and a pipelined FFT. Each dark circle represents a butterfly operation, of which a schematic can be seen in Fig. 2.

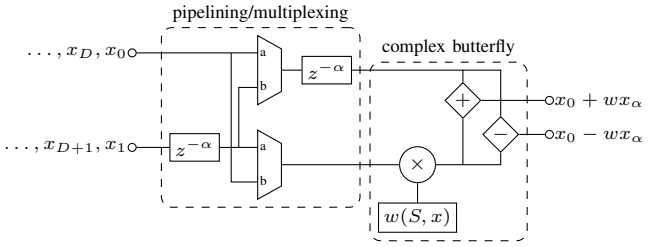


Fig. 2. A simplified diagram of the "biplex" pipelined butterfly FFT, showing the multiplexed multiplier and sample buffering. Here we have $\alpha = \frac{N}{D2^S}$, where S is the stage number using the same convention as Fig. 1. The complex multiplier implements 4 real mutipliers and 2 real adders and each complex adder/subtractor implements 2 real adders, for a total of 4 real multipliers and 6 real adders per butterfly.

only with D samples at once meaning that it takes N/D clock cycles to perform a complete computation. The subsequent FFT however, being of size D, performs a full computation per clock. This means that the full core is "streaming," or that for each input time-domain sample there is exactly one output frequency-domain bin.

A. *Multipliers*

The single stage shown in Fig. 2 can process 4 real, parallel inputs using a single complex multiplier; this amounts to a single real multiplier per parallel input for the pipelined stages. However, since the first and second stage coefficients are simply the first and second roots of unity, 1 and ± 1 respectively, the first two stages require no instantiated multipliers. The total number of instantiated multipliers for the initial $\log_2 \frac{N}{D}$ pipelined stages is thus,

$$M_{\mathrm{FFT}}^{\mathrm{pipelined}} = D \log_2 \frac{N}{D} - 2D \tag{1}$$

Note that these are the number of instantiated multipliers not the amount of multiplications done per computation. Since these stages require N/D clocks to process a full frame, the above number must be multiplied by N/D to get the total multiplications which yields the familiar $N \log N$ form.

The final direct stages are implemented with a standard butterfly structure, see Fig. 1a, with $M_{\mathrm{FFT}}^{\mathrm{direct}} = 4 \times \frac{D}{2} \log_2 D$ multipliers, where we multiply by 4 since these are complex multipliers. In this case this *is* the total multiplications performed per computation. Adding this to the above results in the following total multiplier utilization for the CASPER FFT implementation,

$$\begin{aligned} M_{\mathrm{FFT}} &= D \log_2 \frac{N}{D} + 2D \log_2 D - 2D \\ &= D \log_2 ND - 2D \end{aligned} \tag{2}$$

The FIR filter preceding the FFT uses a single real multiplier per tap, so given T taps (typical numbers are 6 to 8), the full PFB multiplier utilization is shown below with the various components identified,

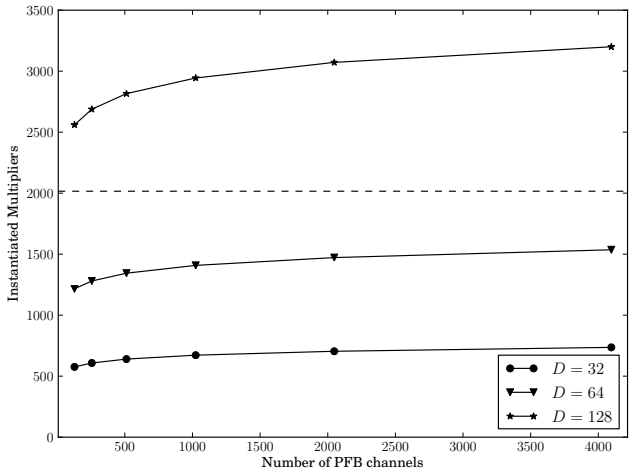


Fig. 3. Number of multipliers versus number of PFB channels for various values of demux. The dashed line is the upper limit if multipliers are implemented using DSP slices on the ROACH 2.

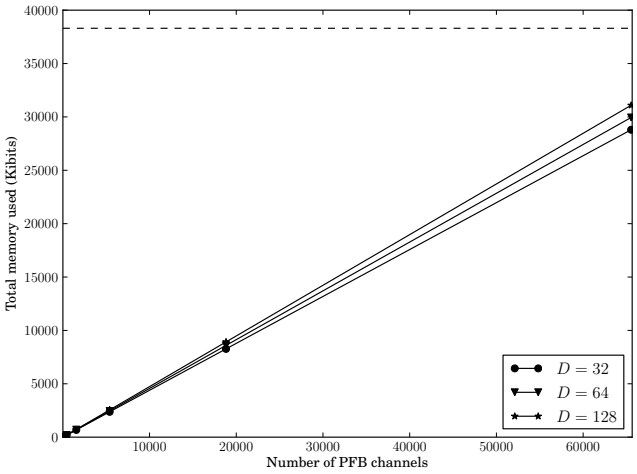


Fig. 4. Kibibits (1024 bits) of memory require by the PFB algorithm for demux of 32, 64, and 128. The ROACH 2 has 38,000 Kb of Block RAM memory and 7,640 Kb of distributed memory.

$$\boxed{M_{\mathrm{PFB}} = \underbrace{D \log_2 ND}_{\mathrm{FFT}} + \underbrace{TD}_{\mathrm{FIR}} - \underbrace{2D}_{\mathrm{optimization}}} \tag{3}$$

The most important thing to note from this equation is that since the pipelined stages divide their computation over N/D clock cycles the total number of multipliers only grows with N as $\log N$. The main contributer to multiplier utilization is instead the demux factor.

Assuming an 8-tap FIR filter, the total number of instantiated multipliers is governed by $D \log_2 DN + 6D$. A plot of this relationship is shown in Fig. 3 for demux of 32, 64, and 128 along with the upper limit for implementing the multipliers in DSP slices, 2016, on the ROACH[2] 2.

B. Adders

To find the total adders we go through a similar calculation but noting that the butterflies have $\frac{3}{2}$ as many adders as multipliers and the FIR only needs to sum all taps and thus performs $D(T-1)$ adds. The total is shown below, again with the contributions pointed out,

$$\boxed{A_{\mathrm{PFB}} = \underbrace{\frac{3}{2} D \log_2 ND}_{\mathrm{FFT}} + \underbrace{D(T-1)}_{FIR} + \underbrace{D}_{\mathrm{reorder}}} \tag{4}$$

The added D results from extra adders in the block needing to do the final reordering of the FFT output. A similar plot to Fig. 3 can be shown for the adders however it would look essentially the same.

C. Memory

The two major consumers of memory will be the delay lines and coefficient storage of the PFB; we will attempt in this section to make a rough estimate of total memory usage by analyzing just these two factors.

The PFB is presented with D samples which proceed through a FIR filter of length NT where T is the number of taps. The delay lines between taps are thus of length N/D of which there are D rows corresponding to each of the D subphases and for every delayed sample there is one coefficient stored. Assuming then that the sample bitwidth is b^s_{FIR} in bits, the stored coefficient bitwidths are b^c_{FIR} in bits, and that these remain constant through the FIR, we see that each of the TD taps in the FIR uses $\frac{N}{D}(b^s_{\mathrm{FIR}} + b^c_{\mathrm{FIR}})$ bits for a total of

$$\begin{aligned} TD \times \frac{N}{D}(b^s_{\mathrm{FIR}} + b^c_{\mathrm{FIR}}) &= TN(b^s_{\mathrm{FIR}} + b^c_{\mathrm{FIR}}) \\ &= TN b^{\Sigma}_{\mathrm{FIR}} \end{aligned} \tag{5}$$

where $b^{\Sigma}_{\mathrm{FIR}}$ is the sum of the sample and coefficient bitwidths. Equation 5 represents the *total memory utilization* of the FIR filter in bits assuming a perfect packing configuration. As we shall see in Memo 2 the reality is that the memory types and sizes available on a specific FPGA, as well as the packing scheme used, significantly affect the utilization.

As we saw in Section IV, the full FFT is split up into $\log_2 \frac{N}{D}$ pipelined stages followed by $\log_2 D$ direct stages (a small-N, small-D example is presented in Fig. 1b). Each pipelined stage requires buffering data by 2α for each parallel input (see Fig. 2), where $\alpha = \frac{N}{D2^S}$ and $S = 1, 2, \ldots$ starting with the first stage moving forwards. Summing over all pipelined stages we get the following expression for the total buffered data (in bits)

[2]https://casper.berkeley.edu/wiki/ROACH2

per partial FFT,

$$b^s_{\mathrm{FFT}} \sum_{S=1}^{n-d} \frac{2N}{D2^S} = b^s_{\mathrm{FFT}} \sum_{S=1}^{n-d} 2^S$$
$$= b^s_{\mathrm{FFT}} \frac{2 - 2^{n-d+1}}{1-2}$$
$$= b^s_{\mathrm{FFT}} \left(2\frac{N}{D} - 2\right) \quad (6)$$

where we have used the relation for a geometric series and defined $n = \log_2 N$, $d = \log_2 D$, and b^s_{FFT} is the bitwidth in bits of samples through the FFT which we assume to remain constant throughout. Note that Equation 6 represents the amount of buffered data *per parallel input*.

The coefficient storage can be found in a similar fashion since the coefficients are simply the roots of unity which number by 2^S per stage. Looking at the w block in Fig. 2 we see that there is a single complex coefficient, 2 real, being multiplied at any particular time. However since this single stage must process the butterfly operations for two streams (due to the multiplexing) it has to store twice as many coefficients. This amounts to needing to store a single set of coefficients per stage per parallel stream, which turns out to be the same amount for the buffered data (but stored in reverse stage order). As an optimization the first 2 stages do not require storage since the first two roots of unity are simply 1 and ± 1 respectively, as we saw in Section IV-A.

Combine this with Equation 6 and we get an expression for the total memory used by the pipelined stages,

$$R^{\mathrm{pipelined}}_{\mathrm{FFT}} = b^s_{\mathrm{FFT}} D \left(2\frac{N}{D} - 2\right) + b^c_{\mathrm{FFT}} D \left(2\frac{N}{D} - 2 - 6\right)$$
$$= 2b^{\Sigma}_{\mathrm{FFT}} (N - D) - \underbrace{6Db^c_{\mathrm{FFT}}}_{\text{optimization}} \quad (7)$$

where b^s, b^c, and b^{Σ} are defined similarly as with the FIR. Please note that this equation does not include storage needed to reorder after every stage.

The direct stages of the FFT by definition do not need to buffer any data, i.e. they operate on only D samples at a time, so the main source of memory usage are the twiddle coefficients. On every clock the direct stages are presented with a different set of partial FFT outputs, $\frac{N}{D}$ for the full window, and thus each butterfly needs to store $\frac{N}{D}$ twiddle coefficients. Since the direct stages are doing the *last* $\log_2 D$ stages of the full FFT there are no trivial coefficients so all must be stored. The total memory used is thus,

$$R^{\mathrm{direct}}_{\mathrm{FFT}} = \underbrace{2b^c_{\mathrm{FFT}} \frac{N}{D}}_{\text{memory per butterfly}} \times \overbrace{\frac{D}{2} d}^{\text{total butterflies}} = b^c_{\mathrm{FFT}} N \log_2 D \quad (8)$$

using similar conventions as previously. Combining Equations 5, 7, and 8 we reach the following expression for total PFB memory utilization,

$$R_{\mathrm{PFB}} = TNb^{\Sigma}_{\mathrm{FIR}} + b^c_{\mathrm{FFT}} N \log_2 D$$
$$+ 2b^{\Sigma}_{\mathrm{FFT}} (N - D) - 6Db^c_{\mathrm{FFT}} \quad (9)$$

the partial FFT term is multiplied by D since there are D partial FFTs. We can assume that $b^s_{\mathrm{FIR}} = b^c_{\mathrm{FIR}} = b^s_{\mathrm{FFT}} = b^c_{\mathrm{FFT}} = b^{\Sigma}/2$ which simplifies Equation 9 to,

$$\boxed{R_{\mathrm{PFB}} = b^{\Sigma} N \left(\frac{1}{2} \log_2 D + T + 2\right) - 5b^{\Sigma} D} \quad (10)$$

This relationship is shown in Fig. 4 for a demux of 32, 64, and 128 with the total Block RAM available with the ROACH 2 as a fit indicator. Again, please note that this is intended as a rough estimate of the total memory and does not include other sources such as reordering or logic implemented in memory.

V. Conclusion

Within a polyphase filter-bank the multiplier and adder utilization appear to grow significantly with the demux factor, namely as $D \log_2 ND$, whereas the amount of required memory depends critically, and linearly, on the total channels, N. Generally this implies that designs with modest bandwidth but requiring significant spectral resolution will be constrained by memory. The SMA wideband correlator, however, has both very large bandwidth and substantial PFB size to achieve a moderately fine spectral resolution. Thus it will be necessary to find the appropriate combination of parameters which meet the requirements of bandwidth and spectral resolution while fitting the logic and memory available in a particular FPGA. This will constrain the width of the processed bandwidth block for a given FPGA, and thus provide an independent constraint (to data rate) on the ADC rate.

The application of these methods and equations to specific FPGAs will be discussed in the next memo in this series. In particular we will look at FPGA-specific optimizations, such as adders folding into DSP slices and packing of memory into Block RAM's of certain widths, in an attempt to narrow down possible PFB implementations.

References

[1] A. Parsons, *A Scalable Correlator Architecture Based on Modular FPGA Hardware and Data Packetization*, http://casper.berkeley.edu, 2008.