INTRODUCTION TO CHANNELIZATION ALGORITHMS IN SDR AND COMPARE THEM

Mehdi naderi soorki : 8605224

Abstract:

In recent years, RF receiver designers focused on replacing analog components with digital ones, trying hard towards the ideal Software Defined Radio (SDR) where all signal processing is done in software. Such an ideal SDR platform may form a flexible and reprogrammable receiver that can deal with many different standards, e.g., IS-95, GSM, UMTS, and especially the various military standards. A wideband receiver has to simultaneously deal with hundreds to few thousands channels, which lay in the same spectrum interval. One of the most computation intensive tasks in such receiver is channelization . A wideband channelizer decomposes its RF input signal into separate outputs, each containing the signal of single channel. The goal of the presented research is to study algorithms for wideband channelization then we compare the introduced algorithms.

1. Software Defined Radio:

The necessity for software defined radio (SDR) appeared from military applications where communication between several different forces (i.e., air-force, ground force, navy, etc.) had to be facilitated while preventing interception by enemy forces. DARPA's SPEAKeasy and JTRS projects are examples for development of SDR, where multiple air-interfaces with different signal processing techniques were integrated into one platform. However, the necessity for SDR also exists in civil applications. Typical example is a cellular phone that is capable of operating within the different existing standards (UMTS, GSM, DCS-1800, IS-95, JDC, and many more).

2. subjects driving the channelization architecture

Two key subjects defining the technical requirements of the channelizer :

2.1.Spectral Content of the Wideband Channel

The frequency allocation plans supported by the SDR drivesThe technical requirements for the channelization approach chosen. At one extreme is cellular communications, where the system architecture typically defines a fixed carrier spacing with a constant RF bandwidth per carrier channel. For example, GSM900 defines an uplink band from 890 to 915 MHz and a downlink band from 935 to 960 MHz (see Figure 1) . Both of these bands contain 124 carrier channels spaced 200 kHz apart. The channelizer supporting this type of network may be able to utilize the redundancy of the channel structure to provide an efficient channelization mechanism.

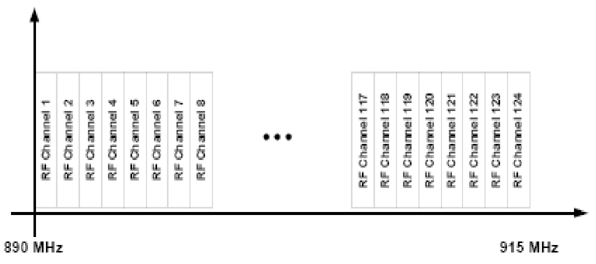

Figure 1: Uplink RF Channel Structure for GSM900

At the other extreme, both the carrier frequency and RF bandwidth per carrier are dynamically assigned. This type of architecture is found in a multi-standard communications system such as a multi-standard satellite Gateway . In these types of systems, subscribers may be assigned a waveform specific to a service offering, or may be assigned an operating mode for a waveform based on pre-defined requirements for quality of service. The carrier frequency, synchronization scheme, and analog bandwidth parameters of each subscriber signal will vary depending upon the specified operating parameters of the assigned waveform (see Figure 2). The channelizer employed in a radio supporting this type of architecture must be flexible enough to accommodate all of the carrier/bandwidth combinations supported by the network architecture, and possibly allow for the dynamic reallocation of channel resources within this architecture during operation

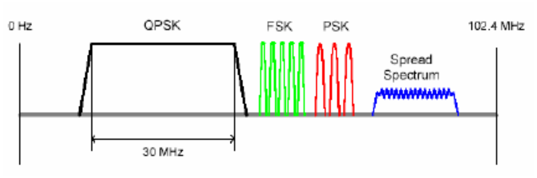

Figure 2: Possible Frequency Plan for Wideband Satellite Radio Link

2.2. Processor Selection for Channelization Processing

In SDR architecture, processing would be limited to general purpose processors (GPPs) communicating with standards based protocols, such as TCP/IP or CORBA. This model allows for maximum reuse of application code across multiple platforms, accelerating time to market and maximizing the return on investment in application software through code reuse and upgradability . In general, however, this type of operating environment is not practical in a field deployable system for two primary reasons:

∗ The power utilization and heat dissipation of GPPs are often prohibitive in many size, weight, and power limited systems. As a result, Digital Signal Processors (DSPs) are often utilized to supplement the processing provided by the GPP to keep the architecture within the specified power budget.

∗ GPPs and DSPs employ a serial processing architecture that does not provide sufficient performance for the processing of wideband signals .As such, the use of Field Programmable Gate Arrays (FPGAs), which gives near ASIC like performance in a programmable device, is often required in the SDR platform.

For these reasons, a heterogeneous processing engine incorporating a combination of FPGAs, DSPs, and GPPs is typically required in the digital transceiver architecture. Front-end channelization processing in this type of platform is typically limited to FPGAs due to performance constraints in dealing with the wideband input, although back-end processing which is preformed on a per channel basis may incorporate DSPs or GPPs.

3. Channelization Algorithms analysis

channelization is a process where single, few, or all channels from a certain frequency band are separated for further processing. The separation of single channel is usually done by down-conversion followed by filtering and optional sample-rate conversion.
The channels of interest may be of equal or different bandwidths and may be uniformly or non-uniformly, continuously or non-continuously distributed over the input frequency band.

4. Channelization Algorithms

This section presents 3 channelization algorithms:

4.1 The per-channel Approach

A straightforward implementation, which is also the traditional implementation of Wide band channelizer, is to simply use a single-channel channelizer for each channel of interest, and connect them all to the input frequency band signal . Figure 3 illustrates such algorithm.

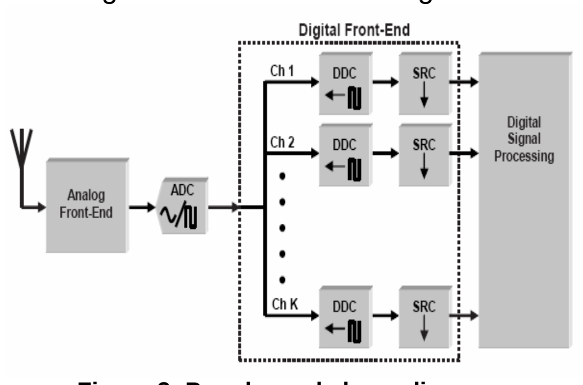

Figure 3: Per-channel channelizer

This approach provides a great deal of flexibility in the choice of channels to be separated . Each single-channel channelizer can be individually designed for BW and frequency choice. Furthermore, the separated channels are not constrained to be of the same bandwidth or to be uniformly distributed over the frequency input band . However, once such channelizer is designed, it is very rigid for alteration. Adapting this channelizer algorithm to different air-interface might require replacement of some or all single-channel channelizers. When a change has to be done only in part of the input frequency band, only the corresponding single-channel channelizers have to be altered or replaced. Another weakness of this algorithm is that for wideband receivers, where many channels are to be separated, silicon costs and power consumption are extremely higher than in other, more advanced wideband channelization techniques introduced in the following sections .

4.2 Pipelined Frequency Transform

The Pipelined Frequency Transform (PFT) algorithm is based on a binary tree of DDCs and SRCs (see Figure 4) where units of DDC followed by SRC are used for dividing their input band into two half-bands with half sampling rate. This algorithm creates a binary tree that splits the input frequency in two half-bands and then splits each half-band again into two half sub-bands and so on, until the last tree level produces the required separated channels. . This algorithm for itself has no advantage over the algorithm presented in the previous section and is actually much more expensive in terms of silicon use, since apart of a

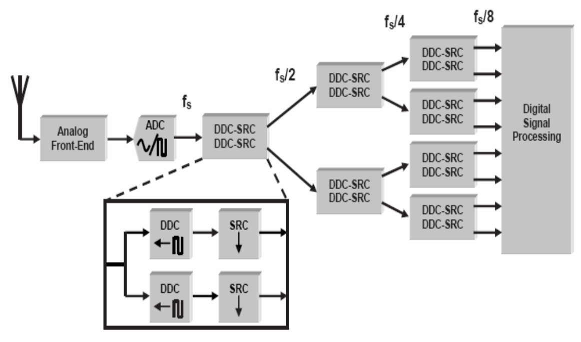

Figure 4: DDC-SRC tree

single-channel channelizer for each channel of interest (as in the perchannel algorithm) in the last stage of the tree, many more are needed in the other stages. Nevertheless, each single-channel channelizer complexity can be reduced dramatically, taking advantage of half band filters symmetry and restricting the output sample rate to be quarter of input sample rate in each single node in the tree. Observing that the components in each stage perform in half of the sampling rate of its former stage components, a considerable optimization can be performed. The actual amount of operations-per-time performed in each level of the tree is equal while distributed over twice components than in its former tree level. Instead of using two components for each component in the former tree level at half sampling rate, one component that performs in the same sampling rate can be used in combination with inter leaver , which distributes the samples accordingly. This is done using complex (IQ) DDC and DUC as illustrated in

Figure 5 (The DDCs and DUCs that are not in the 1st level are of a special interleaved version). The channels however are output serially, and therefore some extra processing is required for distributing them in distinct outputs.

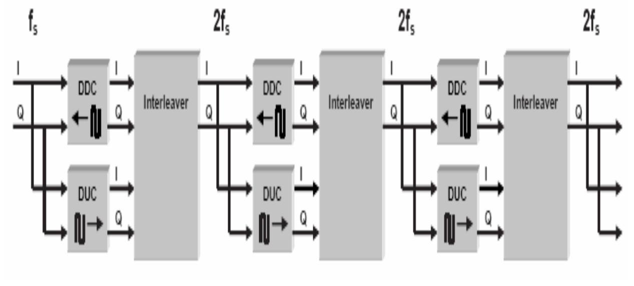

Figure 5: DDC-SRC tree

The PFT algorithm seems to be much more economical in terms of silicon use and power consumption when compared to per-channel channelizers algorithm. Especially when many channels are to be separated from the frequency input band . However, it demonstrates less flexibility, as the separated channels must be of equal bandwidth and uniformly distributed. The Tunable PFT algorithm is an adaptation of the PFT that alleviates this inflexibility by introducing interleavers that provide intermediate outputs from the PFT stages that may be used for fine tuning channelization . This improvement, however, leads to increasing HW costs and is not applicable for wideband channelizers.

4.3 Polyphase FFT

This channelization algorithm is an improvement of FFT channelization using a polyphase filterbank in combination with FFT, taking advantage of the equivalence theorem and noble identities while posing acceptable restriction over the sampling rate.

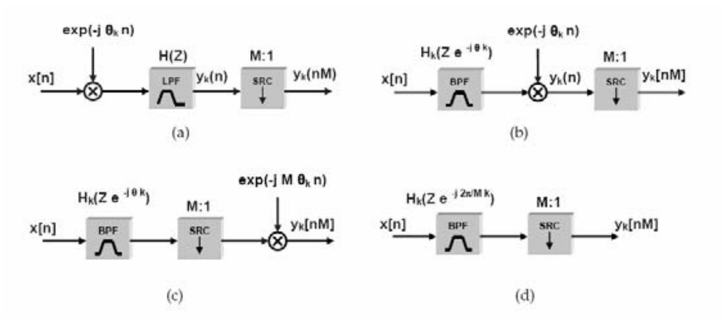

Figure 6: Modifications to the kth single channel channelizer

We consider the kth single (complex) channelizer from the per-channel channelizer in Figure 3 (see Figure 6(a)) and apply series of modifications to it . The expression of the LPF output in Figure 7(a) is a multiplication of the input samples *x*[*n*] with the complex heterodyne and a convolution with the filter coeffcients *h*(*n*), and is given in Equation 4.3.1.

$$\begin{aligned} y_k(n) &= [x(n)e^{-j\theta_k n}] * h(n) \\ &= \sum_{r=1}^{N-1} x(n-r)e^{-j\theta_k(n-r)}h(r) \end{aligned} \qquad (4.3.1)$$

Swapping between the complex multiplier and the prototype LPF alters the LPF to a BPF in accordance with the equivalency theorem (see Figure 6(b)). The corresponding modification to Equation 4.3.1 is shown in Equation 4.3.2.

$$\begin{aligned} y_k(n) &= \sum_{r=1}^{N-1} x(n-r)e^{-j\theta_k(n-r)}h(r) \\ &= \sum_{r=1}^{N-1} x(n-r)e^{-jn\theta_k}h(r)e^{jr\theta_k} \\ &= e^{-jn\theta_k}\sum_{r=1}^{N-1} x(n-r)h(r)e^{jr\theta_k} \end{aligned} \qquad (4.3.2)$$

Observing that only everyMth result of the complex multiplier in Figure 6 (b) is kept after of the SRC, we interchange these two elements while adapting the phase of the complex multiplier (multiply with M) as shown in Figure 6(c). Constraining the center frequency for the kth channel to be an integer multiple of the output sample rate so that $\theta_k = \frac{2\pi k}{M}$ results in aliasing to baseband, since the complex multiplier term becomes $e^{-j2\pi n} = 1 + 0j$. Consequently, the complex multiplier becomes superfluous and can be removed, as shown in Figure 6(d).

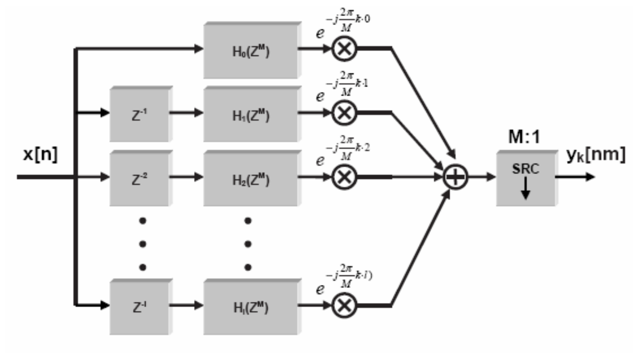
x[n]
Z⁻¹
Z⁻²
Z⁻ˡ
H₀(Zᴹ)
H₁(Zᴹ)
H₂(Zᴹ)
Hₗ(Zᴹ)
$e^{-j\frac{2\pi}{M}k\cdot 0}$
$e^{-j\frac{2\pi}{M}k\cdot 1}$
$e^{-j\frac{2\pi}{M}k\cdot 2}$
$e^{-j\frac{2\pi}{M}k\cdot l)}$
M:1
SRC
yk[nm]

Figure7: *l* branches in the filterbank decomposition of the kth single channelizer

Noting that as before, every Mth output of the BPF in Figure 6(d) is not used due to the SRC, it would be sensible to "shift" the SRC to the left of the BPF. In order to do so, we have to invoke the noble identity . For this purpose we first have to decompose the BPF in the kth single channelizer into a filterbank of *l* (=M) sub-filters. The filterbank decomposition is described in Equation 4.3.3.

$$H(Ze^{-j(\frac{2\pi}{M})k}) = \sum_{r=0}^{M-1} Z^{-r} H_r(Z) e^{j(\frac{2\pi}{M})rk}$$

(4.3.3)

The resulted *l* sub-filters in the filter bank are composed of delay element, sub-filter, and (time invariant) scaling multiplier (see Figure7). Moving the SRC through the scaling multipliers and the *l* sub-filters we invoke the noble identity. The resulted filterbank is depicted in Figure 8. The corresponding output function is shown in Equation 4.3.4.

$$y_k(nM) = \sum_{r=0}^{M-1} y^{(r)}(nM) e^{j(\frac{2\pi}{M})rk}$$

(4.3.4)

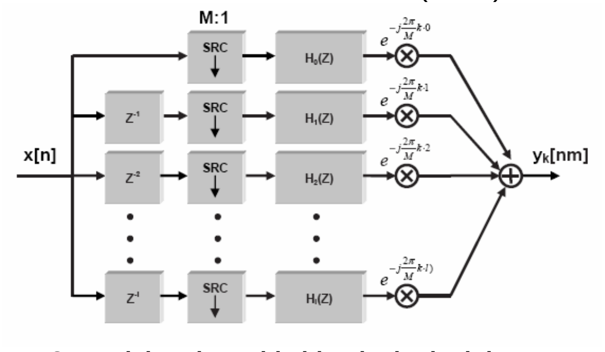
M:1
x[n]
Z⁻¹
Z⁻²
Z⁻ˡ
SRC
SRC
SRC
SRC
H₀(Z)
H₁(Z)
H₂(Z)
Hₗ(Z)
$e^{-j\frac{2\pi}{M}k\cdot 0}$
$e^{-j\frac{2\pi}{M}k\cdot 1}$
$e^{-j\frac{2\pi}{M}k\cdot 2}$
$e^{-j\frac{2\pi}{M}k\cdot l)}$
yk[nm]

Figure8: Applying the noble identity in the kth filterbank

where $y(r)(nM)$ is the *nM*th sample from the *r*th sub-filter.
The delay elements, the SRCs and the sub-filters are similar for all the k filterbanks and therefore only one should be physically implemented. Observing in Equation 4.3.4 that the multipliers and adders practically function as M-points DFT, they can be replaced with FFT for reducing complexity. The final result illustrated in Figure 9 (note that the delay elements are replaced by a chain of one unit delay elements) is known as the polyphase FFT (PFFT) filterbank channelizer algorithm.In comparison with the per-channel channelization algorithm formerly introduced,the PFFT algorithm is much more rigid to changes, and is subject to restrictions imposed over the sampling rate, the number of channels to be extracted, and the number of taps in the prototype filter. however, it seems to show extremely lower silicon costs.

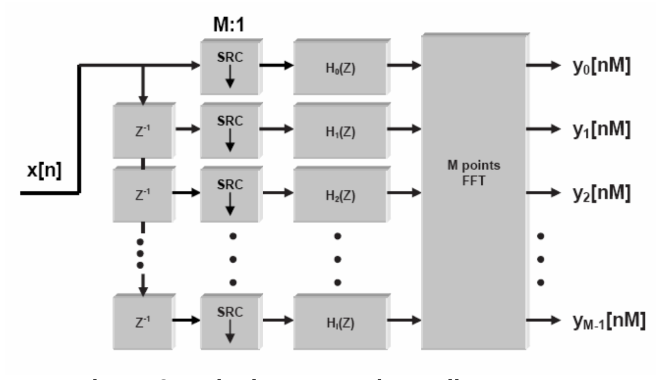

Figure 9: Polyphase FFT channelizer

Measured in terms of number of arithmetic operations per number of separated channels (computational complexity) , it seems that the PFFT outperforms the perchannel algorithm when separating more than 3 channels.

5. Algorithms Comparison

In this section, we present HW-complexity (cost) comparison and a qualitative comparison of introduced channelization algorithms.

5.1 Hardware Complexity Comparison

The following HW complexity comparison is based on data from , which is put herein plots. The first comparison is for LUT utilization. The right plot in Figure 10 show us that the per-channel algorithm (stacked) utilizes far more LUTs than the PFT (binary) and PFFT algorithms and that its tendency is much steeper. The left plot in Figure 10 gives us a clearer comparison between the other two algorithms. We can see that for all given number of channels PFT more than twice LUT resources than the PFFT algorithm does. The second comparison is of memory bits utilization. The right plot in Figure 11 shows us that the per-channel algorithm employs much more memory resources than the PFT and PFFT algorithms for all number of channels. However, comparing the PFT and PFFT algorithms (left plot in Figure 11) we can see that the PFFT algorithm is superior only when channelizing more than 300 channels. Another important property is that the PFT curve's inclination is much steeper than the PFT curve.

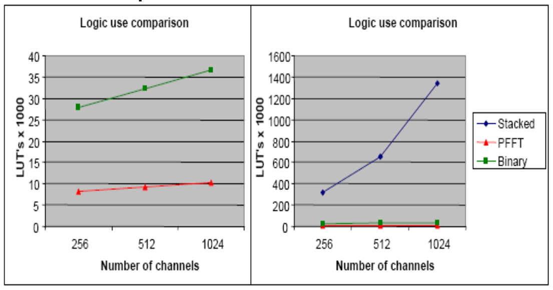

Figure 10: Comparison of LUT utilization

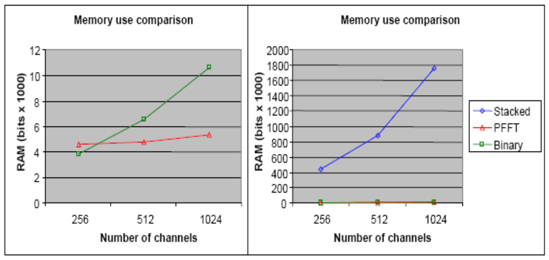

Figure 11: Comparison of memory bits employment

5.2 Qualitative Comparison

The parameters ranges for comparison are wide. Limitation to practical parameters may alleviate this difficulty . Most of newly proposed channelization techniques in literature are compared to the traditional per channel channelizer. The comparisons here are divided to three groups: Computational complexity, size ("silicon costs"), and group delay and flexibility.

5.2.1 Computational complexity

A common comparison parameter is computational complexity, which is usually derived from simulations and software implementations. Such a comparison projects on silicon costs but usually do not take into account memory requirements and control complexity. Previous works show that when 3 or more channels are to be channelized, the PFFT algorithm outperforms the per-channel algorithm. also show that an improvement in the filters of the per-channel algorithm raises this limit to lay between 4 and 20 channels for some scenarios.

5.2.2 Silicon cost

Comparison that is based on actual implementation in FPGA gives a good idea about the HW complexity of the different algorithms. A drawback of such comparison is that it is not platform independent. Different FPGAs contain some dedicated multipliers and built-in memory blocks. Each configuration of distinct algorithm may have trade-offs in FPGA resources that is difficult to measure and compare. Such is the comparison is made between the PFT, PFFT, and per-channel algorithm implementations presented in Subsection 4.1. Based on this comparison, it seems that in terms of memory use, the PFT memory requirement is growing rapidly with the number of channels to be separated. The conclusion drawn in previous works is that up to 256 channels, the HW complexity of PFT and PFFT is comparable and that above 256 channels PFFT outperforms the PFT.

5.2.3 Group delay

Generally, group delay is not a major consideration in the choice of channelization algorithm. It is usually of concern when designing receivers that deal with analysis of short radar pulses. one work shows that the group delays of PFT and PFFT algorithms in different configurations are quite similar. Normally, PFFT group delay is better than in PFT algorithm, but more rigid implementation of the PFT (giving up intermediate outputs) may reach a comparable or better group delay than in the PFFT algorithm. Comparing the PFFT and per-channel algorithm, it seems that the later has a small, advantage due to the FFT stage in the PFFT algorithm.

5.2.4 Flexibility

As this study is aimed toward the mapping of selected algorithm on reconfigurable digitizers, analysis of two flexibility aspects in the different algorithms is essential. The following discussion offers analysis of initial design flexibility and reconfigurability.

5.2.5 Initial design

In this aspect, the per-channel approach is clearly the winner. All the separated channels are independent, may have different bandwidths and may be non-uniformly and non-continually distributed over the input frequency band. The PFT and PFFT algorithms suffer from similar limitations. Namely, producing channels with equal bandwidth that are uniformly and continually distributed over the input frequency band. The PFT suffers from another restriction however. The number of the separated channels has to be an integer power of 2. The PFFT in principle is more flexible in the choice for number of channels to be separated. Nevertheless, the most economical implementations of FFT have integer power of 2 bins, and that may also impose restriction on the implementation of PFFT filterbank. An advantage of the PFT over the PFFT is its possibility to produce intermediate outputs of channels with half of the resolution and twice the bandwidth of the channels in the next level of the PFT tree. The PFFT has a constraint on the number of taps in the prototype filter, which must be
an integer multiply of the number of channels.

5.2.6 Reconfiguration

This is a key concern in the evaluation of the different channelization algorithms . Addition or removal of single or few channels is very easy to implement on the per-channel algorithm, while in most cases, for the PFT and the PFFT algorithms it means a complete reconfiguration of the whole implementation (especially when a change in an integer power of 2 number of channels is required). Adaptation of the filtering performance (channels separation quality) requires modification to the number of taps and the weight for each tap in the filter. In the PFFT algorithm the filtering is implemented in a logically separated block, and therefore its adaptation need not have consequences for the rest of the algorithm implementation. In the PFT and the per-channel algorithms, however, the filters are distributed within different logical blocks, so that adaptation in their performance may have consequences to the rest of the implementation. Table 1 summarizes the qualitative comparison between the different channelization algorithms. Careful examination of the different comparison shows that the per-channel approach wins in many aspects. Conversely, its implementation for high number of channelsis infeasible, and that makes the PFFT algorithm the most suitable for SDR wideband channelizer . Nevertheless, the differences between the PFFT and FFT implementations for medium number of channels (few tens to few hundreds) is not well documented, and investigation in this direction might be subject for further research.

Aspect		Algorithm		
		Per-Channel	PFT	PFFT
Computational Complexity for high number of channels		Poor	Good	**Excellent**
Silicon Cost Efficiency		up to 3-20 channels	up to 128-256 channels	**8-16 channels and above**
Group Delay		Good	Good	Good
Initial Design Flexibility	Independent channels	**Yes**	No	No
	Number of channels	**Selectable**	2^{INT}	Preferably 2^{INT}
	Intermediate outputs	No	**Yes**	No
Flexibility for Reconfiguration	Addition / removal of channels	**Excellent**	Poor	Poor
	Filtering independence	Poor	Poor	**Good**

Table 1: Qualitative Comparison

6. Conclusion

we introduced three different channelization algorithms. Namely, the per-channel, the PFT, and the PFFT algorithms, explaining in details . Consequently, we presented HW comparison between these algorithms for LUT and memory resources utilization. Based on this comparison, the PFFT algorithm appears to be superiorly cost efficient when channelizing few hundreds or more communication channels. Afterwards, we presented a qualitative comparison between these three algorithms that comprises also group delay, initial design flexibility, and reconfigurability. Based on the performed comparisons, we came to the conclusion that despite the fact that the per-channel algorithm has better score in many comparison aspects, its implementation is critically HW inefficient and is infeasible for high number of channels, even on todays largest available FPGAs.

7. REFERENCES

[1] " CHANNELIZATION TECHNIQUES FOR SOFTWARE DEFINED RADIO", Lee Pucker (Spectrum Signal Processing Inc., Burnaby, B.C, Canada;

[2] "FLEXIBLE ARCHITECTURES FOR WIDEBAND SDR CHANNELISATION ",John Lillington (RF Engines, Newport, Isle of Wight, UK; john.lillington@rfel.com) Steve Matthews (RF Engines, Newport, Isle of Wight, UK; steve.matthews@rfel.com)

[3] " EFFICIENT WIDEBAND DIGITAL FRONT-END TRANSCEIVERS FOR SOFTWARE RADIO SYSTEMS ", Approved by: Professor Gordon L. St¨uber, Adviser Professor Mark A. Clements Professor Ye (Geofferey) Li Professor H. Venkateswaran College of Computing Professor John R. Barry Date Approved: April 6, 2004

[4] "Scalable & Reconfigurable Software
Defined Radio Digital Front-End Architecture
FOR Wideband Channelizer
",Gil Savir

[5] "pipelined frequency tansformer (PFT)"

[6] "the ventrix rang polyphase DFT cores "