ALMA

NORTH AMERICAN ARC
ALMA Regional Center

North American
ALMA Science
Center

NRAO

Development Upgrades of the Atacama Large
Millimeter/submillimeter Array (ALMA)

Study Report

EXAMINE IF THE NRC ALMA VERY COARSE CHANNELIZER FPGAS CAN HANDLE 16 GHZ PER SIDEBAND PER POLARIZATION

PRINCIPAL INVESTIGATOR AND PRIMARY AUTHOR: THUSHARA GUNARATNE

CONTRIBUTING AUTHORS: BRENT CARLSON, MIKE PLEASANCE, AND STEPHEN HARRISON

INSTITUTION: THE NATIONAL RESEARCH COUNCIL OF CANADA – HERZBERG ASTRONOMY AND ASTROPHYSICS RESEARCH CENTER

ADDRESS: PO BOX 248, PENTICTON, BC, CANADA V2A 6J9

PI CONTACT INFORMATION: *250-497-2341* *Thushara.Gunaratne@nrc-cnrc.gc.ca*

Prepared By	Name	Thushara Gunaratne	Signature	[illegible] 2022-02-11
	Organization	NRC		
Reviewed By	Name	Brent Carlson	Signature	Carlson, Brent Digitally signed by Carlson, Brent DN: cn=Carlson, Brent, c=CA, o=GC, ou=NRC-CNRC, email=brent.carlson@cnrc-nrc.gc.ca Date: 2022.02.11 14:46:50 -08'00'
	Organization	NRC		
	Name	Mike Pleasance	Signature	Pleasance, Michael Digitally signed by Pleasance, Michael DN: cn=Pleasance, Michael, c=CA, o=GC, ou=NRC-CNRC, email=michael.pleasance@cnrc-nrc.gc.ca Date: 2022.02.11 14:29:42 -08'00'
	Organization	NRC		
	Name	Stephen Harrison	Signature	Harrison, Stephen Digitally signed by Harrison, Stephen Date: 2022.02.11 14:41:48 -08'00'
	Organization	NRC		

National Research Council Canada Conseil national de recherches Canada

Canada

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

LIST OF ACRONYMS AND TERMS

AFSP	ALMA Frequency Slice Processor
ALMA	Atacama Large Millimeter/submillimeter Array
AT.CBF	ALMA TALON Correlator Beamformer
AT.CDP	ALMA TALON Correlator Data Processor
AT.CSP	ALMA TALON Central Signal Processor
AT.HILS	ALMA TALON Hardware In the Loop Simulator
AVCC	ALMA Very Coarse Channelizer
DeTrI	Device Tree Interconnect. A custom interface to interconnect firmware blocks.
DSB	Double Side Band
FEC	Forward Error Correction
FFX	Correlator architecture where two cascading frequency segmentation stages followed by the cross-correlator.
F-FFX	Correlator architecture where the initial frequency segmentation is performed at the antenna followed by two cascading frequency segmentation stages followed by the cross-correlator.
FIFO	First-In-First-Out (memory) buffer
FIR	Finite Impulse Response
FPGA	Field Programmable Gate Array
FS	Frequency-Slice
FSM	Finite State Machine
Gs/s	Giga (i.e. 10^9) samples per second
HAA	Herzberg Astronomy and Astrophysics (Research Center under the NRC, CANADA)
HDL	Hardware Description Language
IDFT	Inverse Discrete Fourier Transform
IFFT	Inverse Fast Fourier Transform
LAB	Laboratoire d'astrophysique de Bordeaux (Université of Bordeaux, FRANCE)
LSB	Lower Side Band
MAC	Media Access Controller (in Transceivers)
Ms/s	Mega (i.e. 10^6) samples per second
NA	North America
NRAO	National Radio Astronomy Observatory (USA)
NRC	National Research Council (CANADA)
NSF	National Science Foundation (USA)
OSPFB	Oversampled Polyphase Filter-Bank
P-IFFT	Parallel Inverse Fast Fourier Transform
RTL	Register Transfer Level
SKA	Square Kilometre Array (www.skatelescope.org)
SLIM	Serial Lightweight Interconnect Mesh
STA	Static Timing Analysis
TDC	Talon Demonstrator Correlator
TDL	Tapped Delay Line
USB	Upper Side Band
VHDL	VLSI (Very Large Scale Integration) Hardware Description Language
WIB	Wideband Input Buffer

ACKNOWLEDGEMENTS

We would like to thank Crystal Brogan for her support in clarifying interface matters between the Digitizer and the Correlator and her comments on the first version of this report.

1 Executive Summary

The 2018 ALMA Development Roadmap [3] identified at least doubling ("2x") the correlated bandwidth of ALMA from 8 GHz to 16 GHz per polarization as its number one near-term priority. This major development initiative has come to be known as the ALMA 2030 Wideband Sensitivity Upgrade (WSU). As originally submitted to the North American ALMA Development Project Call, the NRC's *"2nd Generation ALMA Correlator/Beamformer – the ALMA TALON Central Signal Processor"* (AT.CSP) would only be able to ingest, coarsely channelize into 200 MHz increments called Frequency-Slices (FSs), and correlate 2x BW (i.e., 16 GHz per pol) within the proposal cost-cap, though it would be expandable to larger bandwidth when more funding becomes available in the future. However, recent advancements in digitizer technology have demonstrated the potential of *quadrupling* ("4x") the digitized bandwidth to 32 GHz per polarization [8], i.e., 16 GHz per sideband per pol. Subsequently, the ALMA Project has expressed strong interest in eventually achieving 4x Correlated BW.

Based on design advances that occurred after the initial AT.CSP Project proposal submission [1], this 3-month-long NA Development Study was initiated to investigate if a single ALMA Very Coarse Channelizer (AVCC), each with 2 FPGAs, can be used to ingest and coarsely channelize the full 16 GHz per sideband per pol digitized data stream with the already costed resources. In this scenario, future bandwidth expansion will only require additional hardware for the fine channelization and correlation stage, and in the meantime a user will be able to flexibly select exactly which 16 GHz per pol (from the 32 GHz per pol of digitized data) is sent on for fine channelization and correlation in FSs. Additionally, the resulting detailed design work for the AVCC stage presented here represents a significant jump-start towards one of the initial project milestones of finalizing detailed FPGA designs.

After considerable cross-subsystem discussion and refinement, it has been agreed that a first sub-banding will occur in the antennas in the Back-end subsystem. In the agreed design, the 32 GHz of science bandwidth per polarization contained in the 40 Gs/s sampled sequence will be segmented into sub-bands within the Digitizer subsystem. These sub-bands will be then transferred to the AT.CBF and further segmented into FSs in the AVCCs. One key topic has not yet been resolved, namely the exact nature of the first-F sub-banding, though the NRC and LAB teams agreed to consider two options:

Option 1: Sampled at 2.0 Gs/s containing 1.6 GHz (3.2 GHz for DSB) of bandwidth and

Option 2: Sampled at 2.5 Gs/s containing 2.0 GHz (4.0 GHz for DSB) of bandwidth,

both consisting of complex-valued sample streams at 6 + 6b[1] resolution [10] and resulting in a total of 16 GHz per sideband per polarization of science quality data. However, only one of these options could be selected for the detailed design analysis within the specified time limit. After preliminary assessment of the two first-F sub-banding options, we found that Option 1 is a significantly better match (requiring fewer FPGA multiplier resources and logic), and hence a much lower risk option for achieving the 4x BW goal for the AVCC stage (see Section 11). Thus, Option 1, assuming that the back-end of the Digitizer subsystem produces 10 x 1.6 GHz sub-bands of science quality data per sideband per polarization, was employed for the detailed design study reported here.

[1] Containing 6-bit real-component and 6-bit imaginary component

In the first phase of this study, the design and functional verification of a single instance of a two-stage[2] Oversampled Polyphase Filter-Bank (OSPFB) for segmenting sub-bands into FSs was conducted using Intel's DSP design tool 'DSP Builder' in the model-based design environment MATLAB/SIMULINK, followed by compilation and timing analysis to ensure the design a) functionally performs as expected, and b) performs those functions at the required speed. In the second phase, 20 instances of OSPFBs and the other required firmware blocks to deliver 20 sub-bands to the OSPFBs and FS switching fabric to select and route the resulting 80 FSs x 2 pols to ALMA Frequency Slice Processors (AFSPs), were instantiated. The result of the compilation of this design is that ~65% of the primary logic resources are utilized and F_{max} of 470 MHz is achieved, where 450 MHz is the minimum requirement.

Given this FPGA resource use, the fact that the logic instantiated and compiled is close to the final design—and in fact is the final OSPFB design—and that F_{max} of 470 MHz was achieved, we are at least 95% confident that the final AVCC FPGA design will fit and operate in the required FPGA at 450 MHz.

[2] This is not, functionally, another "F" stage of the AT.CBF FFX design—it is an optimization the "first-F".

2 Background

In the 2018 ALMA Development Roadmap [3], at least doubling the correlated bandwidth of ALMA has been identified as its number one near-term priority. This was based on the vast array of science opportunities that would be facilitated by this upgrade, especially in the key science areas of the *Origins of Galaxies, the Origins of Chemical Complexity, and the Origins of Planets*. In aggregate, this upgrade (including upgrades to the upstream digital system and upgraded receivers) has come to be known as the "*ALMA2030 Wideband Sensitivity Upgrade*" with a goal for completion before the end of this decade (though some receiver bands may be upgraded later).

In response, the Front-End/Digitizer Working Group [4] recommended that the goal should be 4x the current correlated bandwidth because both front-end receiver and digitizer technology were sufficiently mature to make that feasible, in particular that 16 GHz per sideband per polarization should be digitized and transmitted to the correlator. Subsequently, the Correlator Specifications Working Group [5] recommended that at minimum the 2nd Generation ALMA Correlator should be able to produce at least 2x the current bandwidth initially (8 GHz per sideband per polarization) and have a plan to expand it to 4x correlated bandwidth.

In 2020, the correlator design team from Herzberg Astronomy and Astrophysics, National Research Council Canada (HAA-NRC) participated in the ALMA Development Study - Cycle 7 and submitted a report [6] on how to adapt their Correlator and Beamformer (CBF) based on the TALON Frequency Slice Architecture (TALON FSA) for the 2nd Generation ALMA Correlator. The TALON FSA CBF is based on FFX architecture and has been proposed for the SKA Mid.CBF and advanced through multiple review stages including the final design review and is currently being prototyped. Based on this study, in April 2021, HAA-NRC in collaboration with NRAO[3], submitted a proposal to design and build the 2nd Generation ALMA Correlator [7]. After a successful outcome in the NA down-selection and approved by the NSF, this proposal was selected to move forward for a full review and consideration by the ALMA Project and Board. The proposed design, called the ALMA TALON Central Signal Processor (AT.CSP) is comprised of three main elements: the ALMA TALON Correlator Beam Former (AT.CBF), the ALMA TALON Correlator Data Processor (AT.CDP), and the ALMA TALON Hardware in the Loop Simulator (AT.HILS). The AT.CBF has two stages: first, the ALMA Very Coarse Channelizers (AVCCs) where the digitized sequences are segmented into oversampled 'Frequency-Slices' (FSs) containing 200 MHz of usable bandwidth, followed by ALMA Frequency Slice Processors (AFSPs) where the FSs are Resampled into the sample rate of 221.184 Ms/s and corrected for delay/phase and then further segmented into 13.5 kHz wide fine channels and cross-correlated between up to 70 antennas. The original proposal had the limitation that while it could AVCC-channelize and AFSP-correlate 2x bandwidth (16 GHz per pol) and was expandable, it lacked the ability to flexibly select what part of the total 32 GHz per pol expected to be produced by the digitizers (i.e., 16 GHz per pol per sideband), could be correlated.

Subsequent to the NA correlator project down-select, in July 2021, the Digitizer Team at the Laboratoire d'astrophysique de Bordeaux (LAB), University of Bordeaux, France published their final Study report [8] confirming that digitization of 16 GHz per sideband per polarization at a sample rate of 40 GS/s would be feasible. Additionally, the LAB study report indicated the preference to perform a coarse channelization (e.g., 2 GHz of usable bandwidth per sub-band) using oversampled polyphase filter-banks (OSPFBs) to be

[3] for the correlator software development.

instantiated in FPGAs that are needed to interface to the ADCs and facilitate serial data transmission over an optical network. In other words, to perform a "first-F" at the antennas, though preserving the time-series nature of the digitized signal.

Shortly after release of the Digitizer Study Report, cross-subsystem discussions between the LAB and NRC-NRAO teams and a few ESO and NRAO representatives were held in August to October 2021. A number of options for achieving the desired science flexibility between the 4x bandwidth digitizers and 2x bandwidth (initially) AT.CSP were explored. At the same time, the AT.CSP design was evolving to use the innovative "Async-Talon" approach recently developed for the SKA-mid correlator design [9][4]. This approach significantly reduces the resources required in the AVCCs of the AT.CBF because, DDR4 memory in the AVCCs is no longer required for bulk delay compensation, such delay effectively now occurring in the AFSPs' corner-turner where large memory is required anyway. Furthermore, a two-stage implementation (see section 4.1.2) of an oversampling channelizer can further reduce the number of multipliers needed for AVCC implementation. The combination of these two design improvements opened the highly desirable possibility that, within the current capped-budget allocation (which has been capped by the NA Development Program), each AVCC (each with 2 FPGAs) could in fact handle the full 40 Gs/s digitized output (32 GHz of science bandwidth) per polarization from the digitizers compared to just 16 GHz per polarization in the original design. In this scenario, no first-F at the antennas is required, from the AT.CSP point of view, for the purpose of limiting the data transmitted to an (initial) correlation capacity of only 2x BW.

Since the ALMA Management Team had already confirmed that it would be highly desirable to make the path towards eventual expansion to 4x BW correlation a priority, this 3-month NA Development Study was initiated to confirm that with the new design improvements, the AVCC stage could process the raw 40 Gs/s digitized data stream (16 GHz of bandwidth per sideband per polarization) prior to submission of the full proposal to the ALMA Board. If successful, eventual AT.CSP bandwidth expansion would only require additional AFSP hardware. It should be emphasized that the detailed design work for the AVCC stage presented in this study report would have been required in the early stages of the AT.CSP project (if approved) anyway, i.e., it represents a significant jump-start toward completion of the final detailed AT.CSP design.

In the meantime, lack of consent on the need for a first-F at the antennas led to a 'mini-review' by the ALMA Integrated Development Team (IDT) on 13th of October 2021[5]. Alternatives were presented by both teams, and unsurprisingly, the compromise design presented by both teams, was chosen. In the agreed design, the 32 GHz of science bandwidth per polarization in the 40 Gs/s sequence will be segmented into sub-bands within the Back-end subsystem. These sub-bands are then transferred to the AT.CBF and further segmented into FSs in the AVCCs. One key topic was not resolved at the mini-review, namely the exact nature of the first-F sub-banding, though the NRC and LAB teams agreed to consider two options:

Option 1: Sampled at 2.0 Gs/s containing 1.6 GHz (3.2 GHz for DSB) of bandwidth and

[4] The "Async-Talon" [9] approach was developed for the SKA-mid correlator, was subjected to a stringent external review (in process during the original AT.CSP proposal preparation and submission), and was approved.
[5] After the proposal for this study [1] was submitted, but before its start.

Option 2: Sampled at 2.5 Gs/s containing 2.0 GHz (4.0 GHz for DSB) of bandwidth,

both consisting of complex-valued sample streams at 6 + 6b[6] resolution [10] and a total of 16 GHz per sideband per polarization of science quality data. As a result of these decisions, the aforementioned scope of the study was revised to address the first-F at the antenna baseline plan, though with the understanding that only one of the two sub-banding options could be carried through to the detailed AVCC design stage, after a preliminary evaluation. As described in Section 11, this evaluation indicates that Option 1 provides a significantly better match to the AT.CSP architecture with lower multiplier and logic requirements. Therefore, the following study report only considers the detailed design and implementation of over-sampling channelizers to segment sub-bands sampled at 2.0 Gs/s containing 1.6 GHz (3.2 GHz for DSB) of bandwidth, into 200 MHz FSs. That is, we assume that the raw 40 Gs/s digitized data stream will be sub-banded at the antennas into 10 x 2.0 Gs/s per sideband per pol sub-bands, each sub-band containing 1.6 GHz science quality bandwidth for a total of 16 GHz per sideband per pol.

[6] Containing 6-bit real-component and 6-bit imaginary component

3 Study Approach

The purpose of this 3 month long study is to determine the feasibility of handling a total of 32 GHz bandwidth (16 GHz x 2 polarizations) contained in 20 sub-bands with one ALMA Very Coarse Channelizer (AVCC) Stratix-10 1SX280HU2F50E1VG FPGA. The main signal processing firmware blocks in an AVCC FPGA are the 10-channel OSPFBs that segments the 1.6 GHz bandwidth contained in a sub-band at the sample rate 2.0 Gs/s into 8 FSs containing 200 MHz of bandwidth. Hence, it requires 20 OSPFBs to process the total bandwidth of 32 GHz. Further, there are firmware blocks to deconstruct and reconstruct data frames to facilitate data transmission to and from the AVCC. Confirming that all required firmware blocks can be instantiated in the selected FPGA device and configured such that the signal processing functions are performed at the desired rate facilitating the expected data throughput will retire significant risk for the full 2nd Generation Correlator Project Proposal that is to be submitted to the ALMA Board in April 2022.

The step-by-step description of the study is given in the following;

1. Carried out the design and functional verification of the OSPFB using Intel 'DSP Builder' [11] blocks within the model-based design environment provided by Simulink/MATLAB [12]. The OSPFB design was modularized such that key signal processing operations such as the polyphase FIR filter-banks, parallel 10-point IFFT and the array of Half-Band filters (see Section 4.1.2) were separated into different modules along with input and output data arrangement and scaling and re-quantizing.

2. Configured the auto-generation of the HDL code for the OSPFB targeting Intel's Stratix-10 1SX280HU2F50E1VG FPGA. The auto-generated code was verified with RTL simulations using 'ModelSim' with the test-vectors generated by DSP Builder and MATLAB [13]. This is to verify that the generated HDL code for the OSPFB performs the DSP and logic functions as expected.

3. Carried out multiple compilation flows for a single instance of OSPFB in the selected FPGA with the Intel's Quartus Prime Pro design suite [14] and reviewed the key FPGA resource utilization metrics and the Static Timing Analysis (STA) leading to F_{max} estimates. After each run, modifications were made to the DSP Builder design such that F_{max} exceeded the operational clock-rate at 450 MHz with a margin of 150 MHz. This margin was considered to be sufficient to increase the possibility of timing-closure when 20 instances of the OSPFB and other required firmware blocks are added to the design.

4. Modified the HDL code and encapsulated with a HDL 'wrapper' such that it is compatible with the existing framework that have been developed for the firmware development and verification for the Talon Demonstrator Correlator (TDC). This in turn allowed the straight-forward use of key firmware blocks (e.g. 100G Ethernet MAC, Sync-FIFO (single-clock synchronous FIFO) and Circuit Switch firmware blocks) that have already developed.

5. Incrementally integrated up to 20 instances of OSPFB firmware designs and conducted compilation flows. The utilization of primary resources, routing congestion and F_{max} estimates were scrutinized.

6. Incrementally integrated up to six[7] 100G Ethernet MACs, up to six Sync-FIFO firmware blocks to mimic the wideband Input Buffers (WIBs) in the Async-Talon design and two actual 80x80 12-bit wide Circuit Switch firmware blocks that cross-connect the FSs from OSPFB firmware blocks to serial output channels, which ultimately are connected to AFSPs.

[7] All four of configurable high-speed transceivers in the Stratix-10 SX2800 FPGA and two soft-core transceivers. The final implementation will likely only include five 100G transceivers, since 6+6b data fits on this many 100G links (480 Gbps of data), and distributing data received on five links to 20 OSPFBs is much simpler than six.

4 Overview

4.1 The Basics of Oversampled Polyphase Filter-Banks

4.1.1 Single Stage Implementation of Oversampled Polyphase Filter-Banks

The main objective of an oversampled polyphase Filter-Bank is to segment the input bandwidth without gaps and to have no significant spectral leakage in the central part of the channel that is to be further segmented into finer bandwidths. This is achieved at the expense of slightly higher sampling rate for each segmented output. Given that a complex-valued input $x_i(n_i)$ at the rate of B samples per second is corresponding to a critically-sampled sequence of a 'flat' input spectrum $X(e^{j\omega})$, with respect to normalized angular-frequency ω/π, is shown in Figure 4-1-(a). The objective is to segment this bandwidth equally to be carried by N_{ch} channels.

Consider a typical magnitude response of the Finite Impulse Response (FIR) low-pass 'prototype' filter $H_P(e^{j\omega})$ is shown in Figure 4-1-(b). In general, $H_P(e^{j\omega})$ is symmetric and therefore the filter coefficients $h_P(n)$ are *real-valued*. Note that for $\omega/\pi > 0$, nominally the passband and stopband edges of $H_P(e^{j\omega})$ are at $\omega_p/\pi = 1/N_{ch}$ and $\omega_s/\pi = (2O_s - 1)/N_{ch}$ respectively [15](Ch.09). Here, $O_s > 1$ is the over-sampling factor. In practice ω_p is selected to be slightly higher than π/N_{ch} such the magnitude responses of channels of the subsequent channelizer that are at the vicinity of ω_p would not be degraded. Similarly, ω_s would be decreased by the same amount in order to avoid excessive aliasing. Also, it is straightforward to have a linear phase response for $H_P(e^{j\omega})$ by selecting an even-order of filter [15](Ch.03).

A direct although not efficient way of evaluating the output sequence for k^{th} channel $ch[k](n)$, is to first modulate the filter-coefficients $h_P(n)$, by $e^{j\frac{2k\pi}{N_{ch}}n}$, such that the magnitude response $H_P(e^{j\omega})$ is shifted right by $\omega/\pi = 2k/N_{ch}$ and fall between $((2k-1)/N_{ch}, (2k+1)/N_{ch})$.

The output of the filter is evaluated by convolving the input sequence $x_i(n_i)$ with the modulated filter coefficients and then down-sampled by an integer factor of L, where:

$$L = \frac{N_{ch}}{O_s}. \tag{4-1}$$

Therefore, the sample rate for each output channel $ch[k](n)$ is $B/L \equiv (B \cdot O_s)/N_{ch}$. Note that due to placing the passband of the modulated filter coefficients within the Nyquist region, the resulting spectrum after down-sampling may not always center at ω/π =0 [16](Ch.04). Hence, a post down-sampling modulator is required to re-orient the spectrum around ω/π = 0. An example of the resulting magnitude spectrum $Ch[k](e^{j\omega})$ for $ch[k](n)$ is shown in Figure 4-1-(c). It can be shown that the spectral components of $Ch[k](e^{j\omega})$ within the range $(-1/O_s, 1/O_s)$ correspond to the spectral components of the input spectrum $X(e^{j\omega})$ within the range $((2k-1)/N_{ch}, (2k+1)/N_{ch})$, which is identified by 'Ch*k*' in Figure 4-1-(a) scaled with the 'passband-ripple' of $|H_P(e^{j\omega})|$, within the range $(-1/N_{ch}, 1/N_{ch})$. There can be some spectral leakage due to the finite attenuation of $H_P(e^{j\omega})$ in the stopband (i.e. beyond ω_s/π) and aliasing back due to the down-sampling operation. Note that, in theory, $H_P(e^{j\omega})$ can be designed to have an arbitrarily small passband ripple and an arbitrarily higher stopband attenuation at the expense of a filter having higher number of coefficients that leads to a higher implementation complexity, a longer temporal response, and increased power consumption.

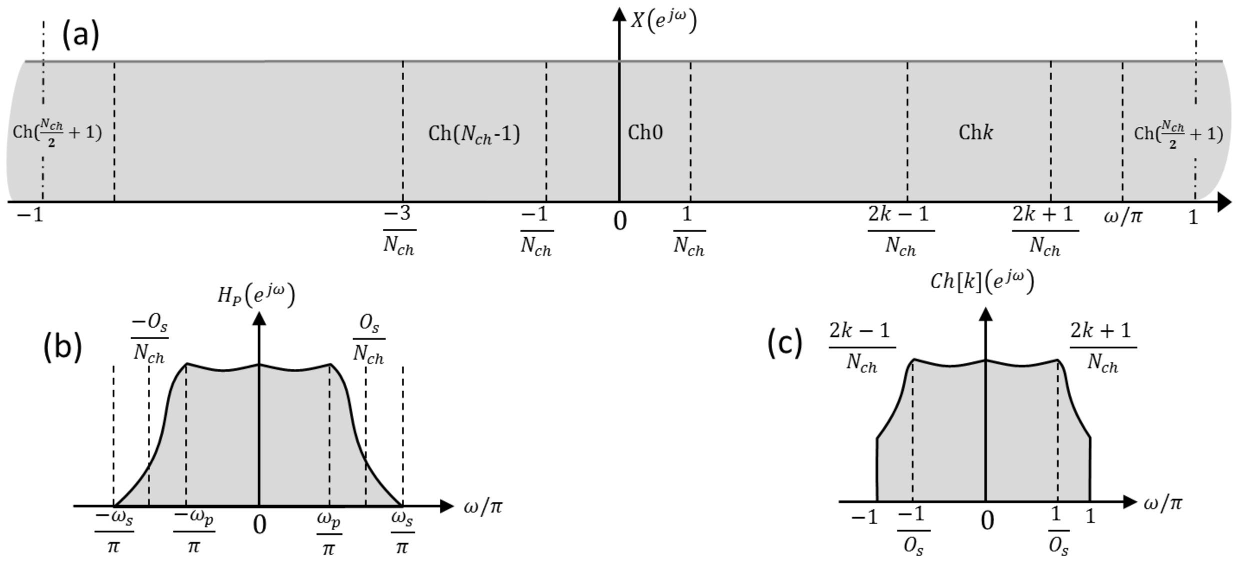

Figure 4-1 Segmentation of the input spectrum by a N_{Ch}-channel over-sampling channelizer having the over-sampling factor O_s (a), the magnitude transfer function of the prototype filter $H_P(z)$ (b), and the output magnitude spectrum of $ch[k](n)$ (c).

A more efficient architecture for realizing the above signal-processing operations is shown in Figure 4-2 [15](Ch.09). Note that the prototype-filter $H_P(z)$ has N_h coefficients. As shown in Figure 4-2, the input sequence $x_i(n)$ is fed into a tapped delay line (TDL) -buffer of length N_h in blocks of L-samples. That is, the n^{th}-block contains the samples $x_i\big((n-1)L+1\big)$ to $x_i(nL)$ in the natural order with the most recent sample at the left. For each such block, a frame of outputs $\{y[k](n)\}$; $k = 0,1,..,N_{ch}-1$ is evaluated such that:

$$y[k](n) = \sum_{l=0}^{\lceil (N_h-1)/N_{ch} \rceil} TDL(k + N_{ch}\cdot l)\cdot h_P(k + N_{ch}\cdot l), \tag{4-2}$$

where $(k + N_{ch}\cdot n)$ refers to the set of elements containing the k^{th} element and every N_{ch}^{th} element until the final element of the sample vector.

Next, the parallel N_{ch} outputs are 'rotated by $(N_c - L)$ for each frame of outputs from the polyphase FIR filters. This operation re-orients the extracted bandwidth around the center of the channel[8]. After the Input Rotation, N_{ch}-point Inverse Fast Fourier transform (IFFT) of $\{y[k](n)\}$ is evaluated such that:

$$ch[k](n) = \sum_{m=0}^{N_{ch}-1} y[m](n)\cdot e^{j\frac{2\pi mk}{N_{ch}}}\text{ ; for } k = 0,1,2,..,N_{ch}-1. \tag{4-3}$$

[8] If this input-rotation is not performed, a parallel complex-modulation with $\left\{e^{j\frac{2\pi Lk}{N_{ch}}n}\right\}$, for $k = (0,1,\ldots,N_c-1)$, then has to be performed on the outputs of the IFFT to re-orient the channelized spectrum [15](Ch.09).

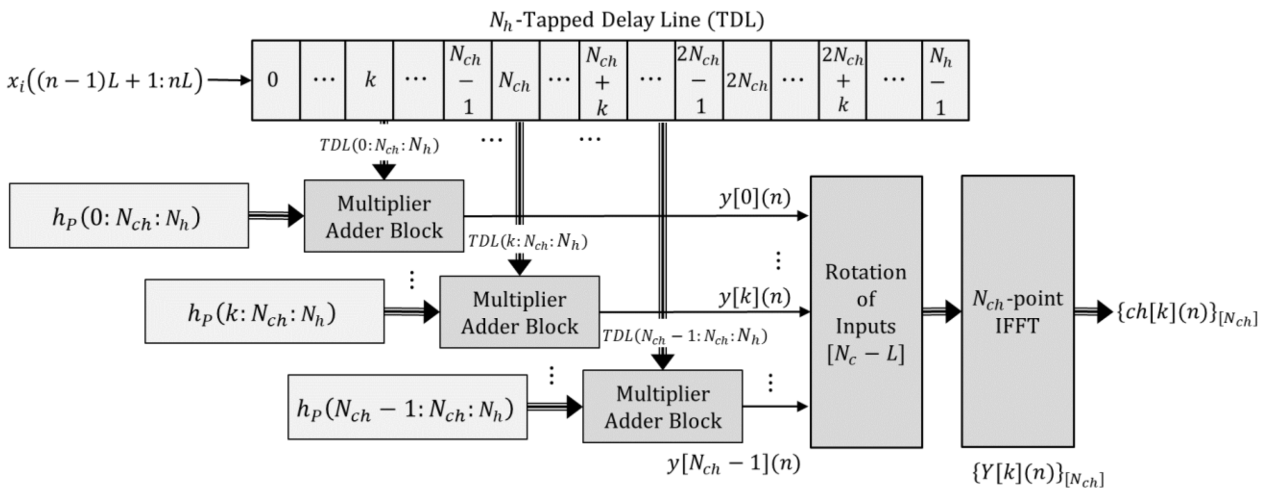

Figure 4-2 Single stage OSPFB architecture.

4.1.2 Two Stage Implementation of Oversampled Polyphase Filter-Banks

In [15](Ch.13), it has been shown that a cascading implementation of a OSPFB and an array of low-pass filters can achieve the required performance with significant computational savings. This is due to the fact that a prototype filter for the OSPFB with a wider transition-band requires fewer coefficients compared to a one with a narrower transition-band [15](Ch.13).

Consider the example shown in Figure 4-3, where the oversampling factor $O_{s1}$of the OSPFB has been selected to be twice the desired oversampling factor O_s (i.e. $O_{s1} = 2O_s$). This in turn increases the transition-band of the prototype filter for the OSPFB by up to 50% and thereby, in general reduce the required number of coefficients by up to a factor of 4 [16](Sec.3.2.4). However, this would produce channels at twice the sample rate than the desired. Hence, before down-sampling by factor of two to achieve the desired sample rate these channels have to be filtered by a low-pass filter to avoid aliasing. As proposed in [15](Ch.13), 'Half-Band' filter would be an efficient way of performing this filtering. For Half-Band filters, almost half of the coefficients are zeros and the rest of coefficients are symmetric and therefore, only a half of the multipliers are needed to implement a Half-Band filter compared to a general low-pass filter with symmetric coefficients. The typical magnitude responses of the prototype filter of the OSPFB for stage-1 and the Half-Band filter for the two-stage implementation of the OSPFB is shown in Figure 4-4.

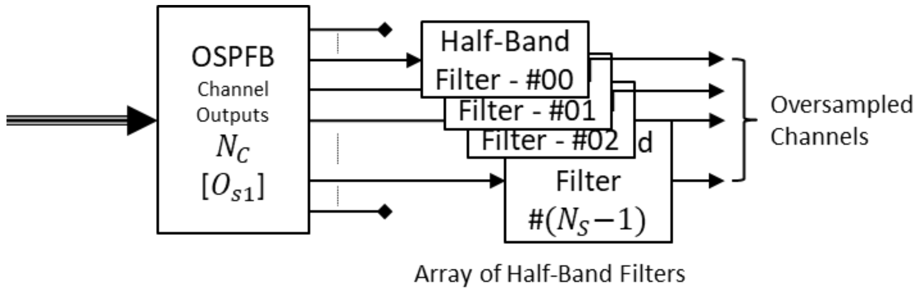

Figure 4-3 System view of a two stage OSPFB.

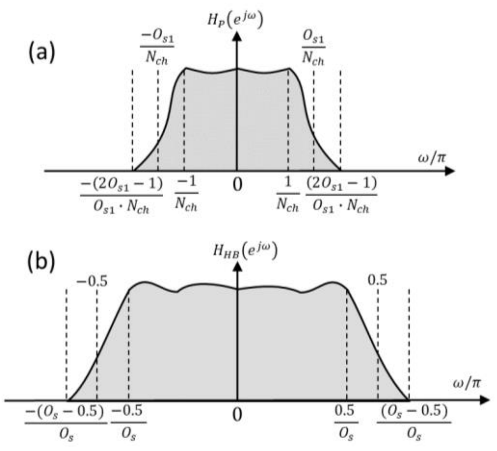

Figure 4-4 Typical magnitude responses of the prototype filter of the OSPFB for stage-1 (a) and the Half-Band filter (b) for the above two-stage implementation of the functionality of OSPFB.

For the two stage implementation of an OSPFB, an efficient implementation of a Half-Band filter that minimizes the required number of multipliers is shown in Figure 4-5. In this example the order of the Half-Band filter is $(N_{HB}-1)$, which must be an even integer but not a multiple of 4. Note that due to the down-sampling by factor of two, only the input samples with even-indices are fed into the 'tapped delay line' (TDL) shown in top of Figure 4-5. On the other hand the input samples with odd-indices are fed into a delay of $(N_{HB}+1)/4$ samples. The value of 'single' filter coefficients that convolve with samples with odd-indices is 0.5 and with binary 2's complement arithmetic's can be impended trivially with shifting the bits of input sample to the left by one position and therefore, does not need a dedicated multiplier. Further, coefficients that convolves with samples with even-indices are symmetric and therefore the corresponding two samples can be added before being multiplied as shown in Figure 4-5. Hence, just $(N_{HB}+1)/4$ dedicated multipliers are needed to implement this Half-Band filter. However, this comes with restriction in the control of stopband attenuation and passband ripple. Overall, the effective impulse response of the two stage OSPFB is *slightly*[9] longer than that of a single stage OSPFB.

[9]This is an inconsequential effect for ALMA since having a short enough impulse response for timing pulsars using VLBI beam data is required, and such is dominated by further downstream processing in the AFSPs.

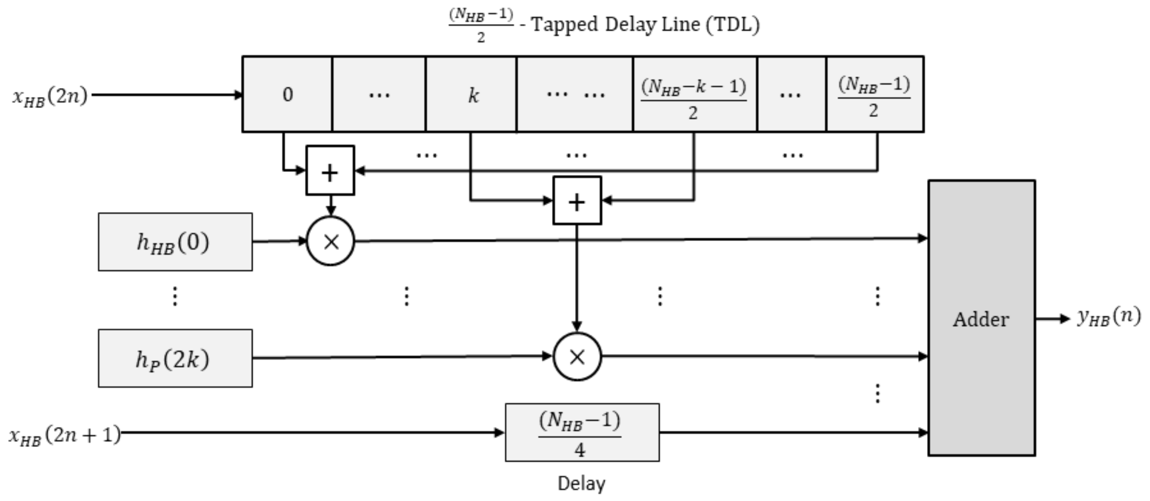

Figure 4-5 An efficient implementation of a Half-Band filter.

4.2 The Proposed Architecture of the Two-Stage Oversampled Polyphase Filter-Bank for Sub-Band Processing

The two-stage OSPFB architecture [15](Ch.13) that has been proposed for extracting FSs from the sub-bands in AVCC FPGAs is shown in Figure 4-6. The main objective is to reduce the number of total multipliers needed such that 20 OSPFBs running at 450 MHz clock rate can be instantiated in a single Stratix-10 1SX280HU2F50E1VG FPGA to process a total of 32 GHz of bandwidth. The input to each OSPFB is complex-valued samples arranged in frames containing $N_p = 5$ contiguous complex-valued samples and producing $N_s = 8$ complex-valued FSs. Note that for the input sample frames both the real and imaginary components are conveyed to the OSPFB simultaneously whereas the output FSs the real and imaginary components are time interleaved in one stream.

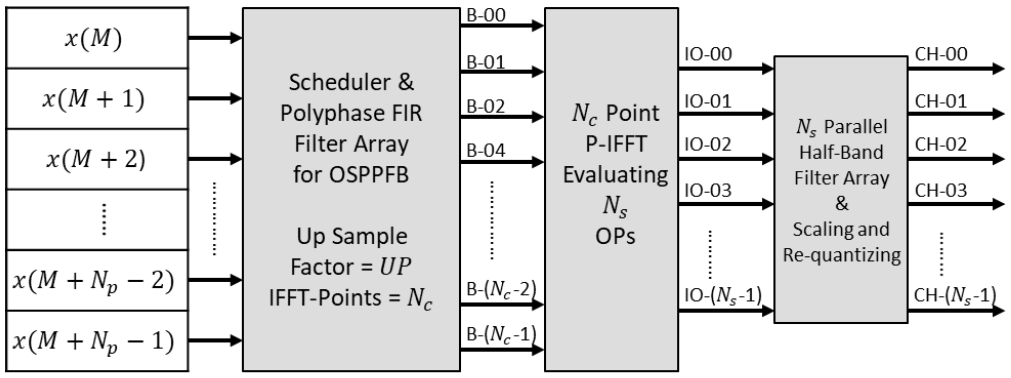

Figure 4-6 Proposed architecture of the OSPFB for extracting FSs from the sub-bands.

The key parameters associated with the selected option are listed in Table 4-1. Note that in order to support the two stage design for the OSPFB with the given input sample rates while achieving the specified output bandwidth, the input samples are nominally up-sampled by a factor of two with insertions zero valued samples. This effectively duplicates the spectrum within the principal Nyquist range [15](Ch.2) and therefore more than one-half of the output channels can be dropped. This further reduces the computations required to perform in the Parallel-Inverse Fast Fourier Transform (P-IFFT).

Table 4-1 The key parameters associated with the OSPFBs.

Input Sample Rate (Gs/s)	**Obs BW (GHz)**	**Number of samples in an Input Frame (N_p)**	**Frames per second**	**Real / Complex Valued Data**	**Up Sampling Factor**	**Over Sampling Factor**	**IFFT Points**	**Selected FSs per OSPBF (N_s)**	**FS Sample Rate (Ms/s)**
2.0	1.6	5	400E6	Complex	2	10/9 ≡ 1.111..	10	8	222.22..

The scheduler (see Figure 4-6) arranges the input samples to be processed by polyphase FIR filters. The flow of input data is controlled by a First-In-First-Out (FIFO) buffer in the scheduler. Due to zero insertion up-sampling, one-half of the samples are zero and therefore, only one-half of the coefficients are required to be multiplied with the corresponding samples. Depending on the state, an array of 2:1 Multiplexers are used to direct the input samples and the corresponding coefficients to the specific 'multiplier adder blocks' (see Figure 4-2). The outputs of these multiplier adder blocks are then subjected to input rotation. If the number of IFFT-points are even, then an array of 2:1 Multiplexers are used to direct the rotated samples to one of two sub-IFFT blocks further reducing the number of resources.

The selected outputs $N_s = 10$, of the P-IFFT block yields a sample rate twice the required rate and the selected outputs are further processed by an array of N_s identical Half-Band filters where the down-sample by a factor of two operation has already incorporated into their architecture (see Figure 4-5). The outputs of these Half-Band filters are arranged such that the real and imaginary components are time multiplexed. The data paths from the input to and outputs of the Half-Band filters are gradually expanded such that to avoid signal saturation. Hence, in order to maintain the desired signal level and reduce the size of the 80x80 FS circuit switch in the FPGA, and data transmission rate to the AFSPs, the outputs of these Half-Band filters are scaled and re-quantized to 8+8b resolution, introducing an inconsequential sensitivity loss.

4.3 An Example Design of the Two-Stage Oversampled Polyphase Filter-Bank for Sub-Band Processing

Following the specifications for passband and stopband edges given in the Section 4.1.2, the prototype filter for the stage-1 OSPFB and the corresponding Half-Band filters are designed to achieve combined stopband attenuation of 41.76 dB[10] and that maintains combined passband ripple within ± 0.1 dB using built in functions in MATLAB.

The prototype filter for the stage-1 OSPFB that meets required specifications is of order 54 and its magnitude response is shown in Figure 4-7. The top pane shows the magnitude response in linear scale illustrating the extended transition-band. As shown in the middle pane the passband ripple is constrained within ± 0.01 dB and in the bottom pane the stopband attenuation is several dBs better

[10] i.e. 1:15,000 instantaneous dynamic range.

than the minimum required 41.76 dB threshold. Note that the passband ripple and stopband attenuation can be constrained with better control for the prototype filter of the stage-1 OSPFB compared to the Half-Band filter.

To meet the required specifications, the order of the Half-Band filter should be higher than 46. The magnitude response of the Half-Band filter of order 46 is shown in Figure 4-8. The stopband attenuation, shown in the left pane of Figure 4-8, is only a couple of dBs better than the required 41.76 dB threshold and the passband ripple, shown in the right pane of Figure 4-8, is within ± 0.06 dB.

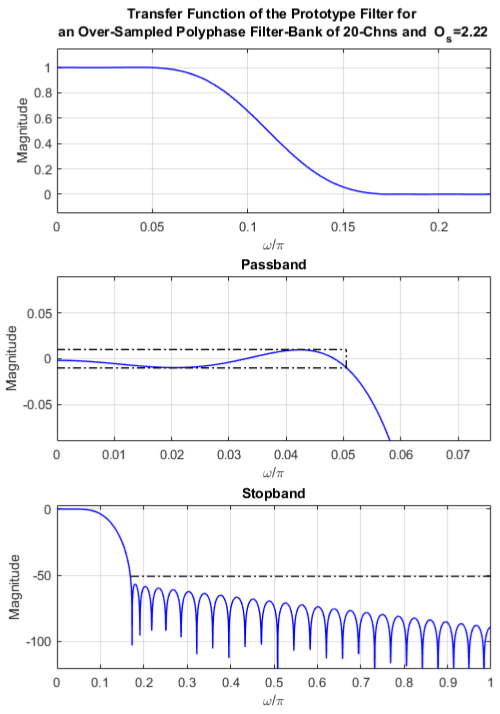

Figure 4-7 Magnitude response of the prototype filter for the stage-1 OSPFB.

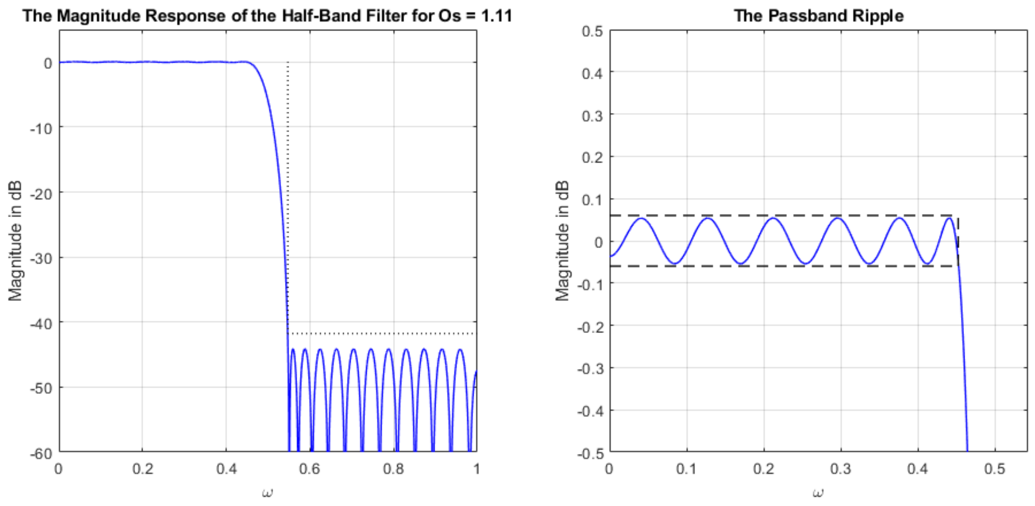

Figure 4-8 Magnitude response of the Half-Band filter of order 46.

The combined magnitude response of the prototype filter of the stage-1 OSPFB and the Half-Band filter is shown in Figure 4-9. The four panes confirm that the requirements are met.

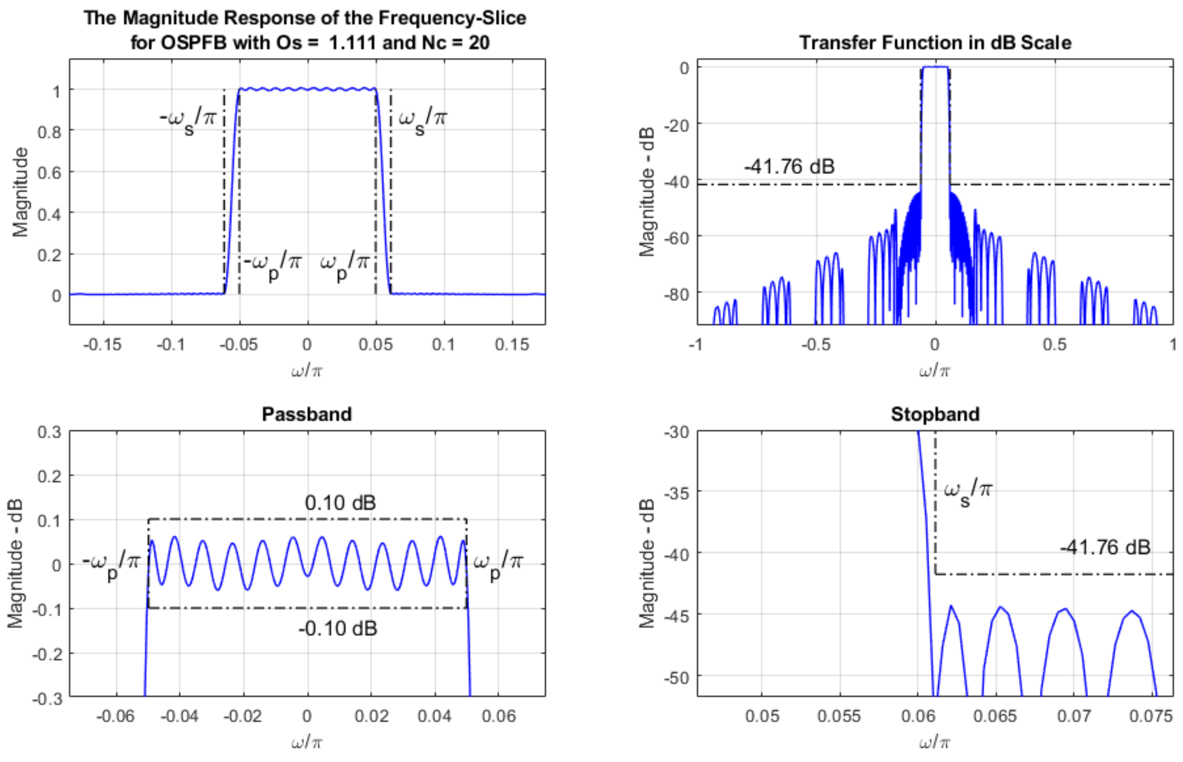

Figure 4-9 Combined magnitude response of the prototype filter of the stage-1 OSPFB and the Half-Band filter.

The initial functional verification of the two-stage OSPFB has been achieved through simulations with test vectors synthesized with collections of sinusoids with known magnitude and phase. FFT methods have been exploited to efficiently evaluate these test vectors. In the following example, the input spectra is selected to consist of uniformly spaced sinusoids with unit magnitude and uniformly distributed phase as shown in Figure 4-10. The corresponding magnitude and phase spectral and the segmented inputs and the corresponding output channels after gain correction compensating for the passband ripples due to combined responses of the stage-1 OSPFB and the Half-Band filter are shown in Figure 4-11 (top). The differences in magnitude and phase between the gain corrected outputs and segmented inputs are shown in Figure 4-11 (bottom).

The impulse response of the OSPFB has been studied using impulsive inputs with unit magnitude placed at different time-slots of the commutator frame. The real and imaginary -components of the responses for each FS for a selected impulsive input are shown in (top) and (bottom) of Figure 4-12. It has been observed that these outputs span maximum of 27 samples (i.e. 121.5 ns). Hence, a flagged input sample[11] would affect 27 output samples at maximum. This fact has been used in the firmware design in

[11] Such is normally only for RFI flagging purposes. The AT.CSP proposal is compliant to requirement 6.2.11 of [5]—no RFI flagging on the real-time data stream is planned, although such could be done if necessary. In the FPGA design done for this study, the mechanism for carrying RFI flags is included, but the logic to generate flags is not.

flag-extension logic. Further, this also shows that the OSPFB has an algorithmic propagation delay of 13 samples. Note that the actual latency through the FPGA implementation of the OSPFB is higher.

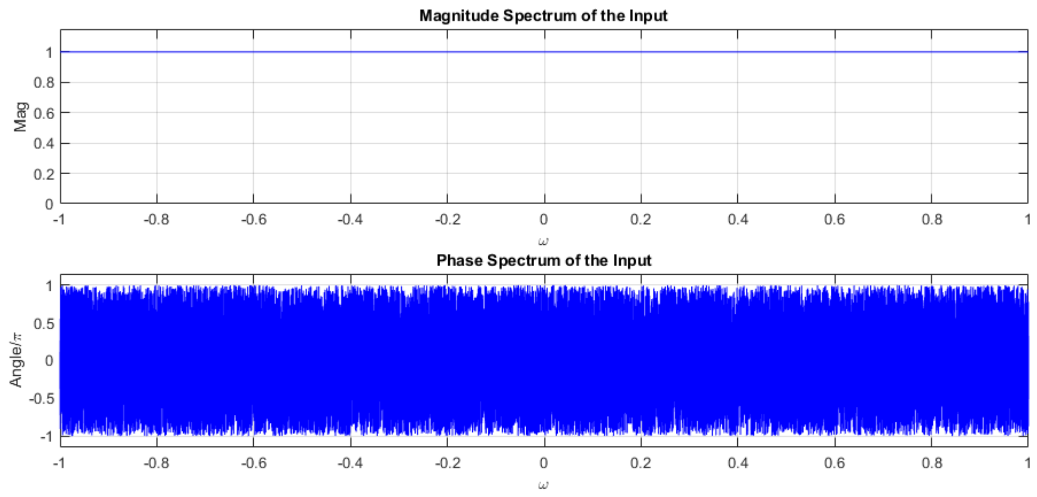

Figure 4-10 Magnitude (top) and phase (bottom) spectra of the input test vector to the OSPFB.

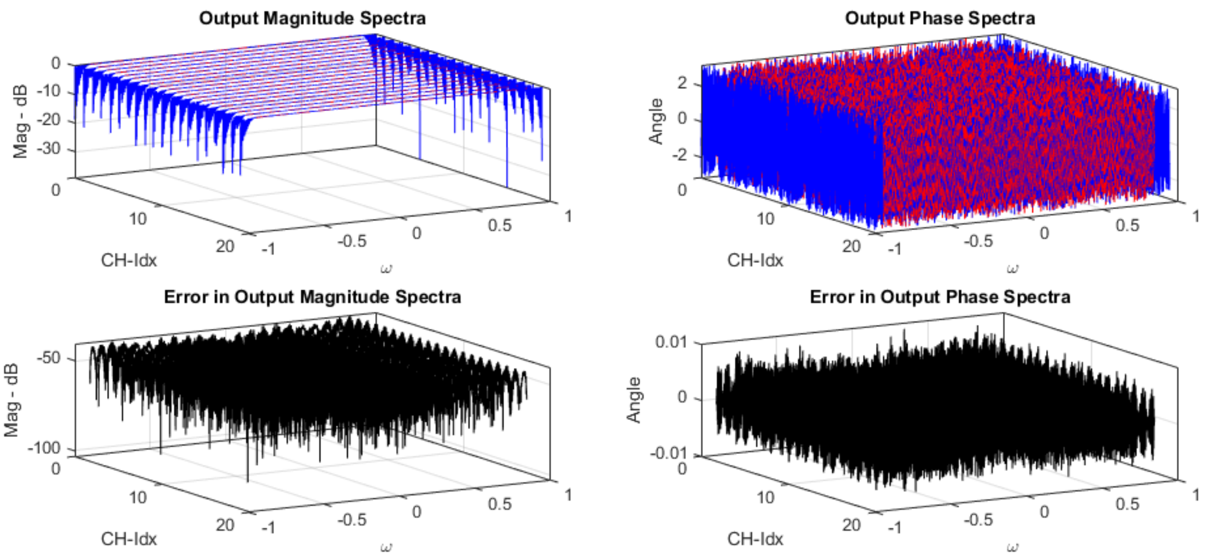

Figure 4-11 Top-Row : Magnitude (left) in dB-scale and phase (right) spectra of the output channels (blue) after gain correction and the corresponding segmented spectra of the input test vector (red) to the OSPFB. Bottom-row: Differences between segmented input and output magnitude (left) in dB-scale and phase (right) spectra.

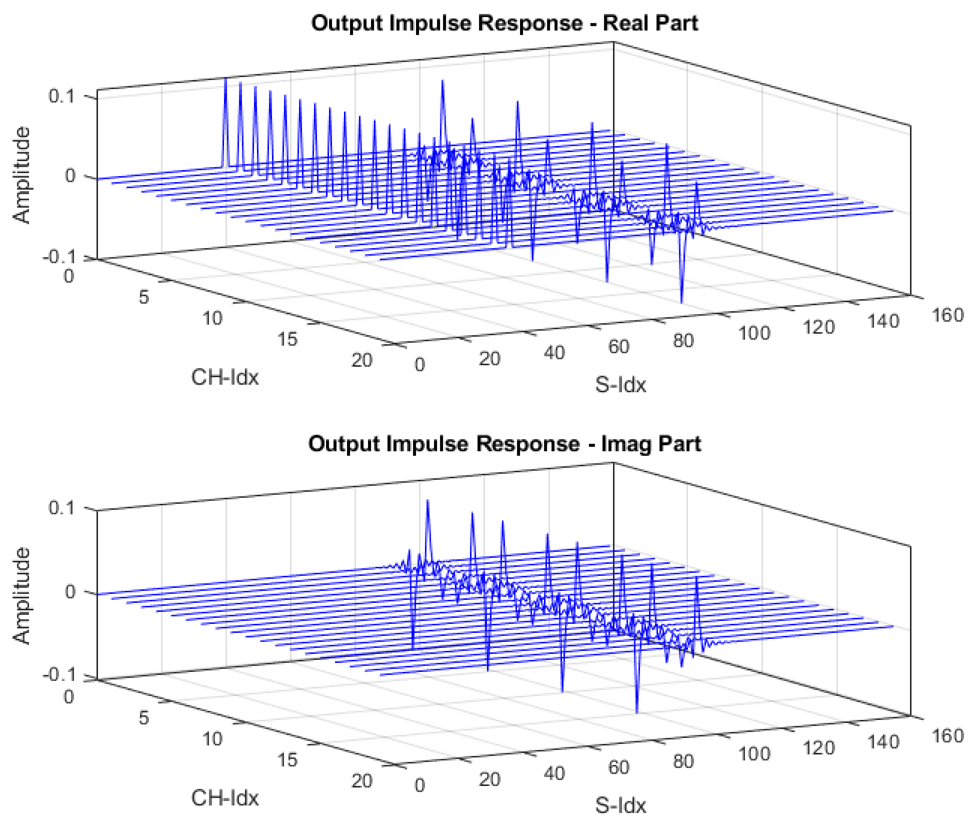

Figure 4-12 Impulse responses of the OSPFB: real-part (top) and imaginary-part (bottom). The impulse response spans a maximum of 27 output samples.

5 Key Study Work Elements

1. Design an OSPFB using Intel's DSP Builder to process a sub-band at the sample rate 2.0 Gs/s and yield 8 FSs containing 200 MHz of observation bandwidth targeting Intel Stratix-10 1SX280HU2F50E1VG FPGA.
2. Generate the HDL code for the OSPFB that is ready for implementation with the selected Intel Stratix-10 1SX280HU2F50E1VG FPGA.
3. Verify the functionality of the HDL implementation of the OSPFB using ModelSim with the stimuli generated using MATLAB/SIMULINK.
4. Configure an AVCC FPGA design that contains 20 instances of OSPBFs, six 100G Ethernet MACs to receive the sub-band data from the Digitizer subsystem, six Sync-FIFO buffers (mimicking WIBs), two 80×80 Circuit Switches to direct the 160 FSs (i.e. 80 FS x 2 polarizations) generated by the 20 OSPFBs to different AFSPs for downstream processing and Device Tree Interface (DeTrI) interconnect linking up the configurable registers associated with aforementioned firmware blocks.
5. Make incremental changes to the firmware blocks, in particular the OSPFB design such that 20 instances can operate at the FPGA clock rate of 450 MHz[12].
6. Compile the entire design, analyze timing, and make design changes/iterate until timing closure is achieved for the required processing clock rates, which is normally 450 MHz, but may be higher or lower in some cases.

[12]This operational clock rate is required to process a total of 32 GHz bandwidth containing in sub-bands resulting a total of 160 FSs each containing 200 MHz.

6 Design Details

6.1 Interfaces

The interface to the ALMA OSPFB firmware block is illustrated in Figure 6-1. The input and output signals to and from the firmware block are arranged into a record array (i.e. 't_CPLX_STRM_a') of the VHDL record 't_ CPLX_STRM' that delivers the real and imaginary components of the associated sample and other control signals. The components of the record 't_CPLX_STRM' are listed in Table 6-1.

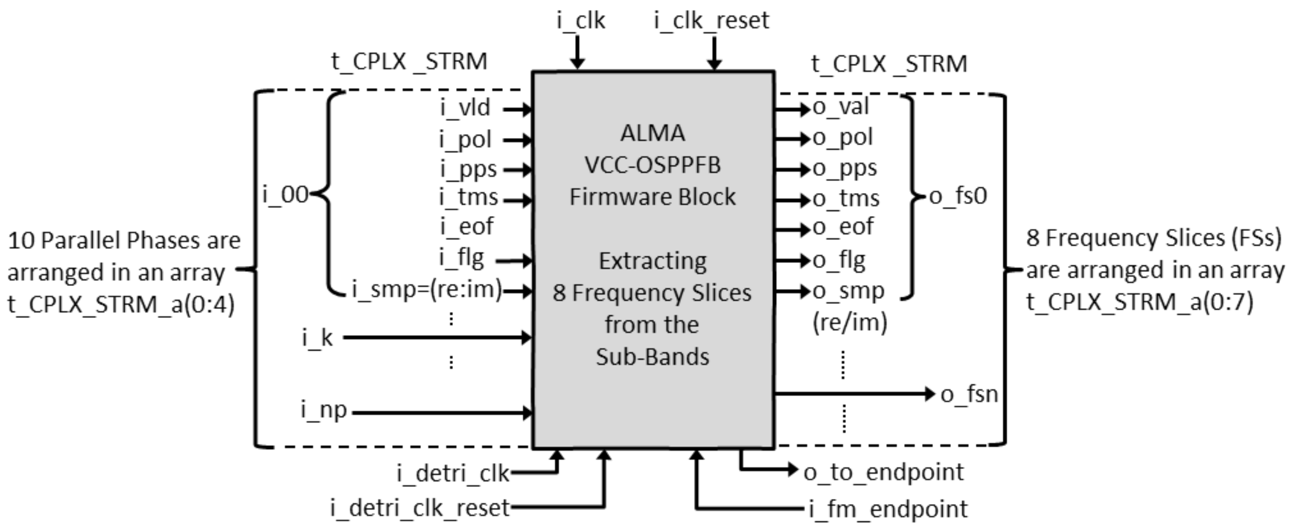

Figure 6-1 Input/output interfaces for the ALMA OSPFB segmenting the sub-bands and generating FSs.

Table 6-1 Components of the VHDL record 't_CPLX_STRM'.

Record Component	Description
_val	The valid marker that indicates the associated sample is valid or not.
_pol	In the input, this signal doesn't represent any significance. In the output, this marks the real (=0) or imaginary (=1) component of the time-interleaved _smp output
_pps	The 'Time-Epoch' marker. In the input this marks 0.048 s time period and in the output marks 0.144 s period.
_tms	The 64-bit 'Time-Code' that is aligned with the '_pps' marker.
_eof	The 'End of Frame' marker that indicate the end of frame for serial 'framed' data. (Not used in the input of the OSPFB)
_flg	The 'Flag' is asserted to mark either saturated or contaminated samples and several other scenarios given in the following.
_smp.re _smp.im	The real/imaginary components of the sample expressed in 'sfixed' format containing up to 18 -bits. The binary point is right of the most significant bit and therefore, the signal amplitude is in the range [-1,+1]. Note that in the output only the real part is non zero

From the remaining signals, 'i_clk' and 'i_clk_reset' signals drive the firmware block and resets the data-path, respectively. The 'i_detri_clk' signal drives the set of registers that can be either written or read by a processor that has access to the FPGA fabric. The 'i_detri_clk_reset' resets the content in these registers. The 'i_fm_endpoint' and 'o_to_endpoint' signals collectively represent the handshake signals, address-bus and data-bus for the 'control' interface with the processor facilitated by the custom 'DeTrI' interface [17].

In the input, five consecutive complex-valued samples of a sub-band sample stream are packed into to an array of records 't_CPLX_STRM_a(0:4)' that consisted of 5 't_CPLX_STRM' records where the real and imaginary parts of each of the five associated samples are assigned to i_smp.re and i_smp.im, respectively. This is illustrated in Figure 6-2. Note that the oldest sample in the sample array is in the top associated with the 0th record 't_CPLX_STRM_a(0)'. Also, the samples corresponding to the 48 ms markers from the digitizer are expected always to be in the 0th record 't_CPLX_STRM_a(0)' with i_pps='1'. Note that the input complex-valued samples are in the format (6+6b) and therefore, the 12 least significant bits of i_smp.re and i_smp.im are set to zero (i.e. i_smp.re/im(-6:-17) = '0').

t_CPLX_STRM_a(0)	i_smp.re = re[x(5·n)]
	i_smp.im = im[x(5·n)]
		i_smp.re = re[x(5·(n+1))]
		i_smp.im = im[x(5·(n+1))]
			....	i_smp.re = re[x(5·(n+m))]
				i_smp.im = im[x(5·(n+m))]

t_CPLX_STRM_a(1)	i_smp.re = re[x(5·n+1)]
	i_smp.im = im[x(5·n+1)]
		i_smp.re = re[x(5·(n+1)+1)]
		i_smp.im = im[x(5·(n+1)+1)]
			....	i_smp.re = re[x(5·(n+m)+1)]
				i_smp.im = re[x(5·(n+m)+1)]

t_CPLX_STRM_a(2)	i_smp.re = re[x(5·n+2)]
	i_smp.im = im[x(5·n+2)]
		i_smp.re = re[x(5·(n+1)+2)]
		i_smp.im = im[x(5·(n+1)+2)]
			....	i_smp.re = re[x(5·(n+m)+2)]
				i_smp.im = re[x(5·(n+m)+2)]

t_CPLX_STRM_a(3)	i_smp.re = re[x(5·n+3)]
	i_smp.im = im[x(5·n+3)]
		i_smp.re = re[x(5·(n+1)+3)]
		i_smp.im = im[x(5·(n+1)+3)]
			....	i_smp.re = re[x(5·(n+m)+3)]
				i_smp.im = re[x(5·(n+m)+3)]

t_CPLX_STRM_a(4)	i_smp.re = re[x(5·n+4)]
	i_smp.im = im[x(5·n+4)]
		i_smp.re = re[x(5·(n+1)+4)]
		i_smp.im = im[x(5·(n+1)+4)]
			....	i_smp.re = re[x(5·(n+m)+4)]
				i_smp.im = re[x(5·(n+m)+4)]

Figure 6-2 Arrangement of the real and complex components of the input samples of the sub-band in the record array 't_CPLX_STRM_a(0:4)'.

In the output, the eight FSs are arranged into an array of records 't_CPLX_STRM_a(0:7)'. Note that o_pps marks the sample corresponding to every third i_pps corresponding to an interval of 144 ms[13]. Also, the 64-bit time-code o_tms is only valid when o_pps is asserted. In between o_pps marks, o_tms to remain unchanged. Further, o_eof, signal is asserted to indicate the last valid sample before the subsequent o_te marker. The output samples are of format (8+8b) and as opposed to inputs the real and imaginary components are time interleaved and marked accordingly with o_pol = '0' for real component and o_pol = '1' for imaginary component, respectively. Hence, o_smp.re(-8:-17) = '0' and o_smp.im(0:-17) = '0'.

[13]In an FS, at the sampling rate of 222.222.. Ms/s there are 10,666,666.6667.. samples in every 48 msec. In order to mark o_tms exactly at the corresponding sample, the marking interval has been increased to 144 msec such that there are exactly 32,000,000 samples between the two o_te marks.

National Research Council Canada Conseil national de recherches Canada Canada

6.2 Monitor and Control

The registers for monitor and control of the ALMA OSPFB firmware block along with some description are listed in Table 6-2. Each register is 32-bits wide and can be accessed in 8-bit (byte-wide) segments.

Table 6-2 List of monitor and control registers for the ALMA OSPFB firmware block.

Offset Address	Read / Write	Register Name	Description
0	Some are read – only (RO) some are read – write (WR)	Flag Status	Bit 00 (RO): i_pps marker has not arrived. Auto reset to '0' when the first i_pps arrives. Bit 01 (RO): The expected number of valid samples has not received between two consecutive i_pps markers. Auto reset to '0' when the expected number of valid samples are received between two i_pps markers. Bit 02 (RO): The FIFO in the data scheduler has been overflown. Fatal condition! Need reset to resolve this. Bit 03 – 04 (RW): Selection of eight consecutive channels from the ten possible outputs of the OSPFB. 00 – Selects Ch#1 : Ch#8 as Fs-00 : FS-07 01 – Selects Ch#2 : Ch#9 as Fs-00 : FS-07 10 – Selects Ch#0 : Ch#7 as Fs-00 : FS-07 Bit 05 – 31 : Reserved
1	R/W	Sft_Scl_00	The shift* (bits 19-16) and scale† (bits 15-0) of the gain factor for FS-00
2	R/W	Sft_Scl_01	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-01
3	R/W	Sft_Scl_02	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-02
4	R/W	Sft_Scl_03	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-03
5	R/W	Sft_Scl_04	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-04
6	R/W	Sft_Scl_05	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-05
7	R/W	Sft_Scl_06	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-06
8	R/W	Sft_Scl_07	The shift (bits 19-16) and scale (bits 15-0) of the gain factor for FS-07

* 'Shift', a 4-bit unsinged integer derives the shift of the binary point of the samples of the particular FS to the left.

† 'Scale', a 16-bit unsinged normalized fractional number, scales the samples of the particular FS, before being quantized to 8-bits. Note that for addresses 1 – 8, bits 20 – 31 are reserved.

Criteria for Assertion of Flags

1. o_flg is asserted for valid outputs until the first i_pps marker arrives. The 00-bit of the read register at the offset address 0 is also asserted to match with this.
2. o_flg is asserted for until the data pipelines of the OSPFB are filled (i.e. for 27 valid output samples) with valid inputs since the first i_pps marker.
3. If i_flg is asserted in an input frame, o_flg is asserted for the subsequent 27 valid output samples indicating contaminated outputs.
4. If the firmware block doesn't receive the expected number of valid samples between two consecutive i_pps markers after the initial i_pps has arrived, o_flg is asserted until the expected number of samples are received between consecutive i_pps markers. The 01-bit of the read register at the offset address 0 is also asserted/deasserted to match with this.
5. If the FIFO in the scheduler (see Figure 4-3) has overflown o_flg is asserted until the firmware block is reset. The 02-bit of the read register at the offset address 0 is also asserted to match with this.
6. o_flg is asserted for individual samples, if the outputs saturate when re-quantize to (8+8b)-words after shifting/scaling.

6.3 Internal Architecture

A modularized design has been adopted for the implementation of the two-stage OSPFB firmware block. The main modules of the two-stage OSPFB firmware block are shown in Figure 6-3. Brief descriptions of the included functionalities and specific interfaces are given the in the following sub-sections.

6.3.1 Data-Condition PPS Slip/Miss Detect Module

As shown in Figure 6-3, input data enters the OSPFB firmware block through Input Data-Condition PPS Slip/Miss Detect module. The mapping of input data and control signals carried in the record-array 't_CPLX_STRM_a(0:4)' to these inputs are specified in Table 6-3.

Table 6-3 The mapping of the 't_CPLX_STRM_a(0:4)' to the inputs of the Input Data-Condition PPS Slip/Miss Detect module.

Record Entry	Input Signal for the OSPFB_IP	Signal Type
t_CPLX_STRM_a(0).vld	IN_V	std_logic
t_CPLX_STRM_a(0).pol	IN_Pol(0); IN_Pol(7 downto 1) →'0'	std_logic_vector(7 downto 0)
t_CPLX_STRM_a(0).pps	IN_PPS	std_logic
t_CPLX_STRM_a(0).flg	IN_Flg	std_logic
t_CPLX_STRM_a(0).tms	IN_TC	std_logic_vector(63 downto 0)
t_CPLX_STRM_a(0:4).smp.re	IN_D_(0:4)re	std_logic_vector(5 downto 0)
t_CPLX_STRM_a(0:4).smp.im	IN_D_(0:4)im	std_logic_vector(5 downto 0)

Note: t_CPLX_STRM_a(0:4).smp.re and t_CPLX_STRM_a(0:4).smp.im need to be converted to std_logic_vector from 'sfixed'

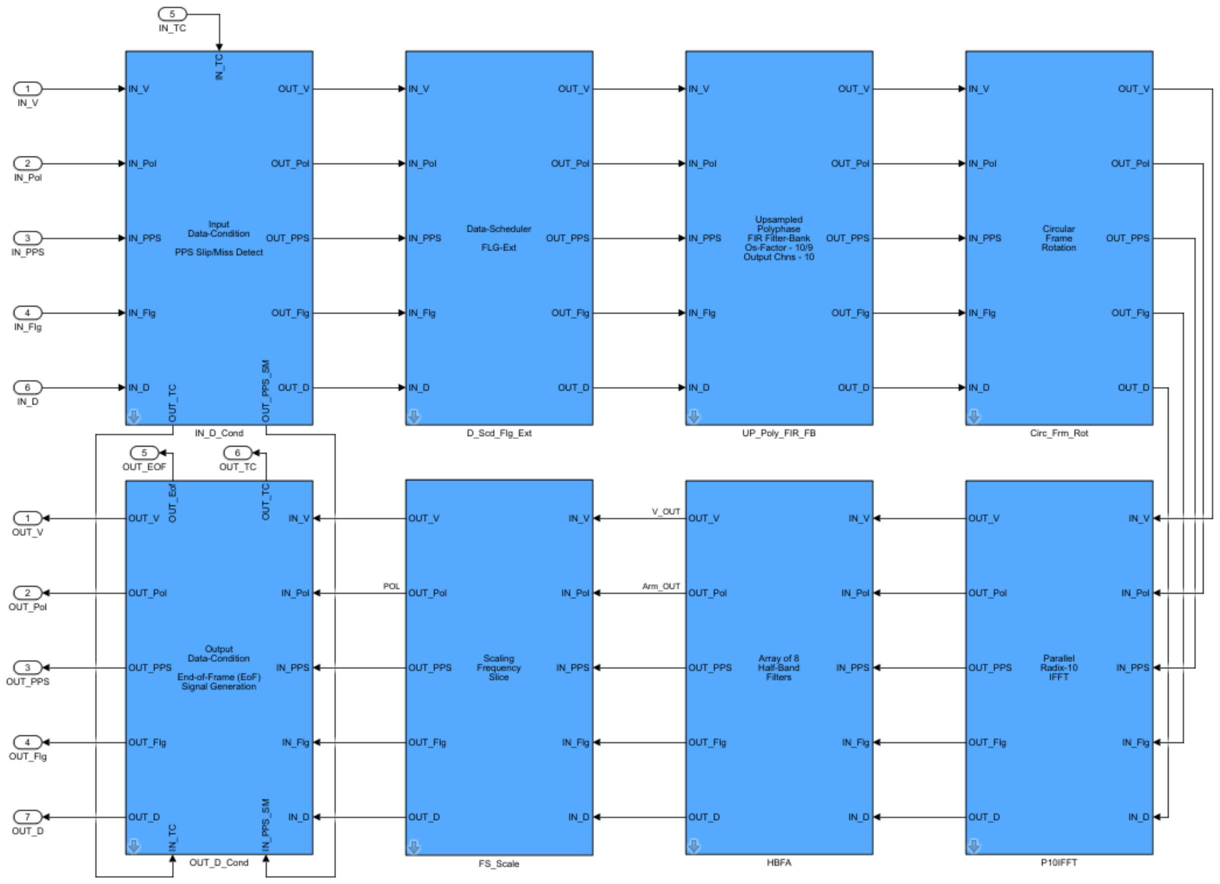

Figure 6-3 Modularized architecture of the two-stage OSPFB_IP firmware-block.

In the Input Data-Condition PPS Slip/Miss Detect module, the input data vector is rearranged as shown in Figure 6-4, (i.e. the latest value is in the top of the rearranged data-frame 'OUT_D' as opposed to the input data frame shown in Figure 6-2). In addition to that this module contains the logic to detect the first arrival of time-epoch marker carried on IN_PPS and to check the expected number (i.e. 19,200,000) of valid input frames between two time-epoch markers separated by 0.048 s. Different number of sample frames between two time-epoch markers indicates a PPS slip or a PPS miss. Accordingly, the processor readable bits #0 and #1 of the register at the offset address 0 (see Table 6-2) are either set or reset indicating that the first time-epoch marker has or has not arrived and a PPS slip/miss has or has not occurred, respectively.

OUT_D_0	re[x(5·n)] + j·im[x(5·n)]	re[x(5·(n+1)] + j·im[x(5·(n+1)]	....
OUT_D_1	re[x(5·n-1)] + j·im[x(5·n-1)]	re[x(5·(n+1)-1)] + j·im[x(5··(n+1)-1)]	....
OUT_D_2	re[x(5·n-2)] + j·im[x(5·n-2)]	re[x(5·(n+1)-2)] + j·im[x(5··(n+1)-2)]	....
OUT_D_3	re[x(5·n-3)] + j·im[x(5·n-3)]	re[x(5·(n+1)-3)] + j·im[x(5··(n+1)-3)]	....
OUT_D_4	re[x(5·n-4)] + j·im[x(5·n-4)]	re[x(5·(n+1)-4)] + j·im[x(5··(n+1)-4)]	....

Figure 6-4 Re-arranged input data frame in Input Data-Condition PPS Slip/Miss Detect module.

In order to minimize the data-paths through the firmware block the input time-code 'IN_TC' is latched to a register in Data-Condition PPS Slip/Miss Detect module and the output 'OUT_TC' then latched by another register in Output Data-Condition module (see Section 6.3.8) with the output time-epoch marker 'O_PPS'. Similarly, the flag indicating PPS slip/miss is also directly transferred to the Output Data-Condition module to be combined with the output flag.

6.3.2 Data-Scheduler and Flag-Extension Module

The outputs 'OUT_V, OUT_Pol, OUT_PPS, OUT_Flg and OUT_D' from Data-Condition PPS Slip/Miss Detect module are the inputs 'IN_V, IN _Pol, IN _PPS, IN_Flg and IN _D' for the Data-Scheduler and Flag-Extension module. These five signals are carried from module to module until the penultimate module Frequency-Slice Shift & Scale module. The Data-Scheduler and Flag-Extension module contains the logic for the Finite State Machine (FSM) that arranges the input data frame 'OUT _D' for the following Upsampled Polyphase FIR Filter-Bank module. There are two alternating states where the first state a 5 element sample-frame contains 5 new input samples and the second state only 4 new input samples in the 5 element sample-frame. To aligned with this FSM also set 'OUT_Pol' signal of the Data-Scheduler and Flag-Extension module toggles the between '00000000' and '00000001' to indicate the two set of coefficients to be used for the Upsampled Polyphase FIR Filter-Bank module.

Also, there is logic to extend an input flag 'IN_Flg' associated with a sample (or samples) in the 'IN_D' sample-frame for the corresponding 27 samples in the outputs. There is a FIFO buffer in Data-Scheduler and Flag-Extension module to absorb the incoming frames while the FSM arranges the input to be processed by the polyphase FIR filter-mask. The depth of this FIFO has been selected considering the nominal sample rates of the input assuming uniformly distributed clock cycles with invalid data (i.e. IN_V ='0'). These cycles are used in the FSM for generating the oversampled outputs. However, if this FIFO buffer is overflown then the timing integrity of the OSPFB firmware block is lost and is unrecoverable without resetting. In such case, the bit #2 of the register at the offset address 0 is set '1' indicating the fatal error has occurred.

6.3.3 Upsampled Polyphase FIR Filter-Bank Module

The Upsampled Polyphase FIR Filter-Bank module contains the DSP resources (e.g. 18x19 bit multipliers and adder-trees) and control-logit to efficiently implement the polyphase FIR filter-mask to an input sample stream as if it is upsampled by factor of two by inserting 0 valued samples in between two input samples. Here, the required number of multipliers are cut in half by using arrays multiplexers to feed the coefficients corresponding to non-zero samples into the multipliers according to the IN_Pol signal.

6.3.4 Circular Frame Rotation Module

This module performs the circular rotation of the output of the Upsampling Polyphase FIR Filter-Bank module such that the spectra of the outputs are centered at 0 Hz [15](Ch.09).

6.3.5 Parallel Radix-10 IFFT Module

In the Parallel Radix-10 IFFT module, the parallel IFFT operation is implemented using two parallel 5-Pont IFFT modules implemented using the Winograd approach [18] and combined with Twiddle-factor multiplications. The module also contains logic to select 8 contiguous outputs out of the 10. The default selection is channels (1 : 8) such that the central 1.6 GHz of bandwidth of the sub-band is segmented into the 8 FSs. The configuration bits [3,4] of the register at offset address 0 can be set as given in Table 6-2 select the channels (1 : 8) or (2 : 9) or (0 : 7) as the output FSs, respectively.

6.3.6 Half-Band Filter Array Module

This module contains 8 parallel Half-Band FIR filters of order 46. As mentioned in Section 4.1.2, each filter contains just 12 coefficients that requires non-trivial multiplications. Due to the down sampling by factor of two the filter outputs are arranged to alternate real and imaginary components. Therefore, it only requires 96 (= 8 × 12) multiplies are required to implement the 8 parallel Half-Band FIR filters shown in Figure 4-5.

6.3.7 Frequency-Slice Shift & Scale Module

The outputs of the Half-Band filters are shifted and scaled before being quantized to 8-bits in this module. The magnitudes can be shifted up by integer factors ¼, ½, 2, 4, 8 and 16. Also, finer adjustment of the scaling is possible with scaling factors ranging within (0.5, 1). The shifting and scaling factors can be configured through processor accessible registers with offset addresses 1-8 as specified in Table 6-2. Also, this module contains logic to detect saturation and or over-flow of the samples due to shifting for each individual FS and the saturated over flown samples are marked with flag for that particular FS.

6.3.8 Output Data Condition Module

This module contains the logic to delay the time-epoch marker OUT_PPS to represent the propagation delay through the polyphase FIR filter and the Half-Band filters and generate the O_EOF signal to mark the real- and imaginary- components of the valid sample before the time-epoch marker. Further, this module contains the registers to latch the time code to 'OUT_TC' to align with 'OUT_PPS' and combine the flags to mark PSS Slip/Miss with the flags due to saturation/overflow. The mapping of output data from the Output Data Conversion module to the 8 't_CPLX_STRM_a(0:7)' records corresponding to the data and control signals are given in Table 6-4.

Table 6-4 The mapping of the outputs of the Output Data-Condition module and the eight records 't_CPLX_STRM_a(0:7)' correspond to the FSs.

Record Entry (n = 0..7)	Input Signal for the OSPFB	Signal Type
t_CPLX_STRM_a(n).vld	OUT_V	std_logic
t_CPLX_STRM_a(n).pol	OUT _Pol(0);	std_logic
t_CPLX_STRM_a(n).pps	OUT_PPS	std_logic
t_CPLX_STRM_a(n).flg	OUT_Flg_n	std_logic
t_CPLX_STRM_a(n).tms	OUT_TC	std_logic_vector(63 downto 0)
t_CPLX_STRM_a(n).smp.re	OUT_D_n	std_logic_vector(7 downto 0)
t_CPLX_STRM_a(n).smp.im	unassigned	N/A

7 Test and Verification

7.1 Functional Verification of a Single OSPFB

During the design process key signal processing and signal arrangement operations were modeled in MATLAB and the results were compared against the algorithmic implementation of a two-stage OSPFB (see Sections 4.1.1 and 4.1.2) considering the re-quantization of outputs in each stage. After the full integration, the functional verification of a single instance of the 10-channel OSPFB has been achieved through first by running selected test stimuli with the SIMULINK model and second by simulating the HDL code generated by the DSP Builder in ModelSim with the same test stimuli generated in MATLAB/SIMULINK.

7.1.1 Gaussian Distributed Test Vectors

In general, the amplitudes of the samples of the sub-bands processed by the two-stage OSPFBs in AVCC are Gaussian distributed. Therefore, the signal-quality achievable with this particular implementation of the OSPFB is preferably evaluated using Gaussian distributed test-vectors. Hence for this test, the real and imaginary sequences are generated using the built in MATLAB function ‘randn’ and each sequence is scaled by 0.2[14] resulting a complex-valued sample sequence corresponding to the combined standard deviation of 0.2828. Next, these sequences are quantized using two ideal quantizers with uniformly distributed quantization-levels (-31, 30, ... , -1, 0, 1, ... , 30, 31)/32 such that these levels can be uniquely expressed using 6 bits, each. Similarly, the coefficients for both the stage-1 OSPFB and the Half-Band filters specified in Section 4.3 are ‘normalized’ and then quantized using an ideal quantizer with uniformly distributed quantization-levels (-131071, -131070, … , -1, 0, 1, … , 131070, 131071)/ 131072 such that these levels can be uniquely expressed using 18 bits. For this particular configuration where the combined standard deviation of the Gaussian distributed complex-sequence is 0.2828, the shift-factor and scale-factor (see Section 6.3.7) for each of the FSs have been selected as 2 and 65535/65536, respectively.

The companion signals IN_V, IN_Pol, IN_PPS, IN_Flg and IN_TC are required to correctly drive the SIMULINK model for the OSPFB. By design IN_Pol = ‘0’ for all times. For this test it had been selected IN_Flg = ‘0’ for all times. For the series of tests, the input time-epoch marker is asserted IN_PSS = ‘1’ for every 240th valid input frames[15]. Finally and most importantly an invalid cycle (i.e. where IN_V = ‘0’ and other signals holding their previous value) is inserted after every 9th input frame resulting uniformly spaced invalid-cycles that accounted to 11.11..% of the valid input-frames. Also, randomly spaced invalid-cycles were inserted that accounted to 1.39..% of the valid input-frames. These percentages correspond to the nominal rates of operation for the AVCC FPGA. In addition to that a few invalid cycles were added in the beginning and end of the simulation to expose any initialization issues and to accommodate the latency of the firmware block. The input stimuli for the SIMULINK model of the OSPFB, except IN_TC, are shown in Figure 7-1.

The corresponding outputs from the SIMULINK model of the OSPFB, except OUT_TC, are shown in Figure 7-2. Note that in the output 11.11..% of the uniformly spaced invalid cycles were *absorbed* by the OSPFB to generate oversampled FSs and only the 1.39..% invalid cycles remain. Further, the OUT_PPS marks

[14] Such that the probability of clipping in either real or imaginary component is 5.7×10^{-7}.

[15] In the normal operation IN_PPS = ‘1’ for every 19 200 000th valid input frames.

the time-epoch marker for every 3rd IN_PPS and delayed by 13 valid samples to account for the algorithmic propagation delay through the two-stage OSPFB. Also, a new signal OUT_EOF is generated to mark the real and imaginary components of the sample just before the sample corresponding to OUT_PPS. For the 8 FSs, OUT_D:1, .. , OUT_D:8, carry the time-interleaved real- and imaginary-components that are marked by OUT_Pol = '0' for real-component and OUT_Pol='1' for imaginary-component, respectively. Note that output flags for the 8 FSs set until the first IN_PPS arrives (i.e. OUT_Flg:1, .. , OUT_Flg:8='1'). Once it arrives, the output flags were maintained for the first 27 valid samples to indicate that these output samples correspond to a partially filled filter-mask. Beyond this, a flag in OUT_Flg:1, .. , OUT_Flg:8='1' indicate the instances where either the real- or imaginary-component of OUT_D:1, .. , OUT_D:8, has exceeded the range after shifting and scaling.

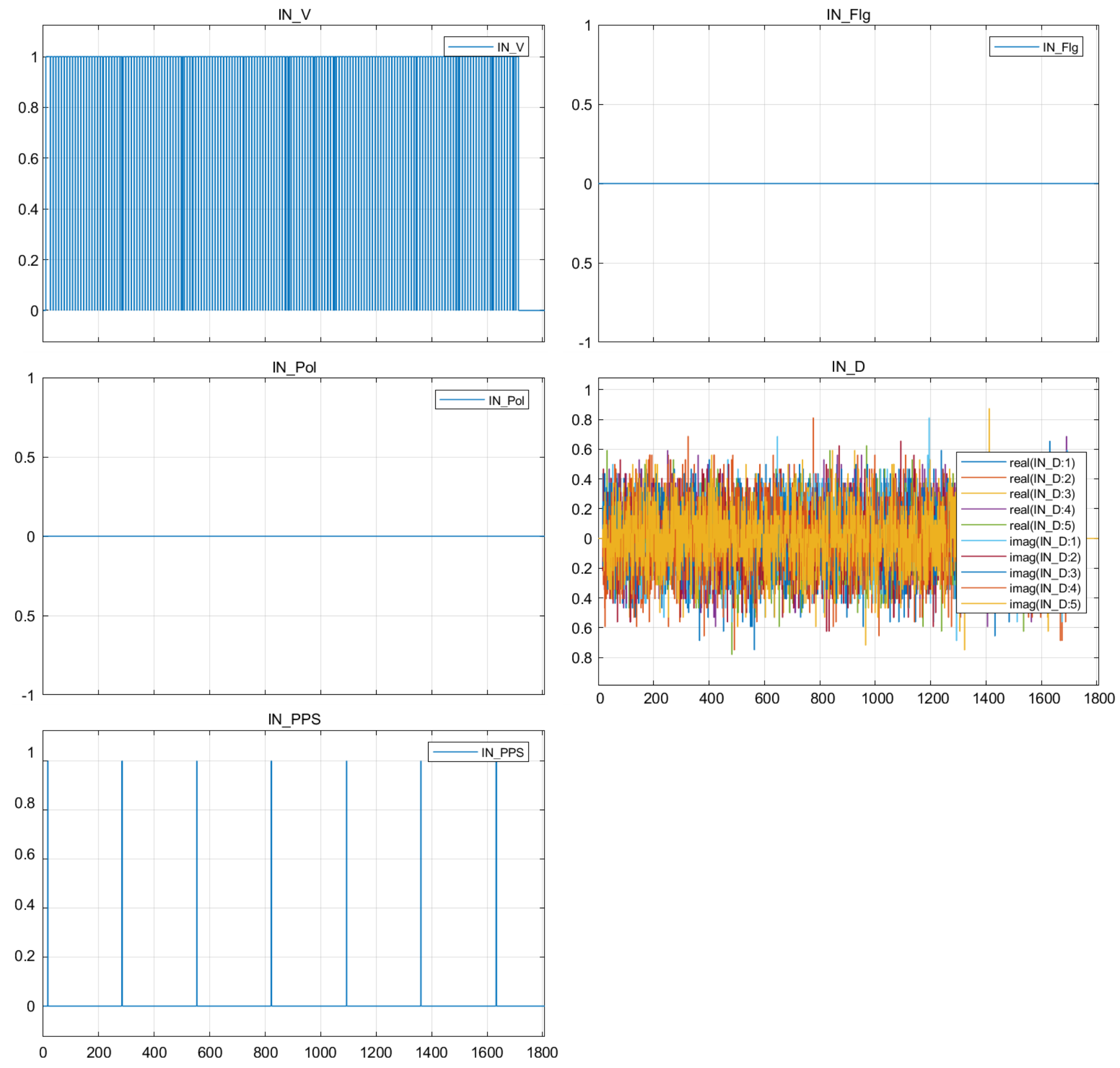

Figure 7-1 Stimuli (Except IN_TC) containing Gaussian distributed test vectors fed to the SIMULINK model of the OSPFB.

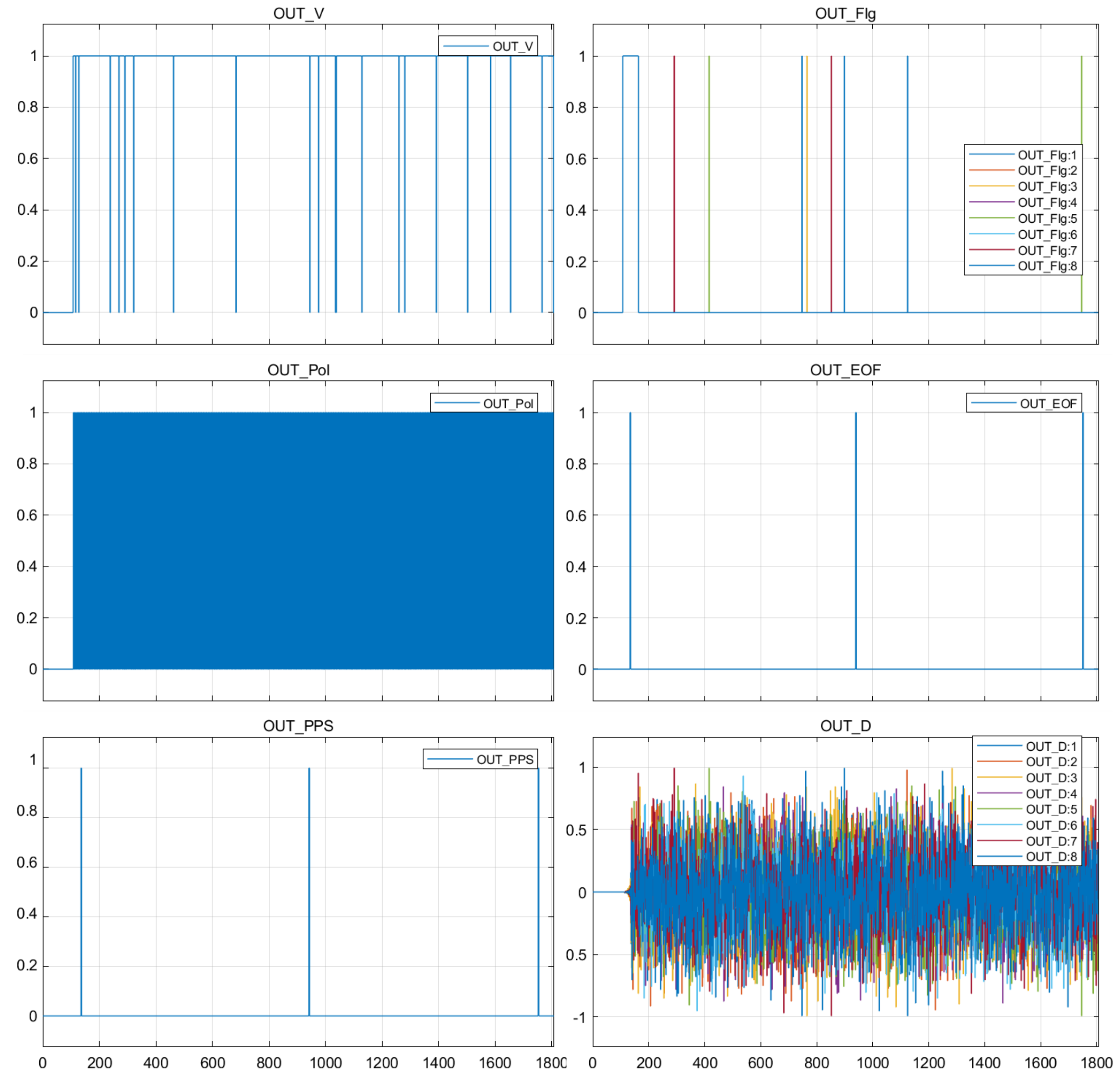

Figure 7-2 Responses (Except OUT_TC) from the SIMULINK model of the OSPFB in response to stimuli shown in Figure 7-1.

The comparison between the valid outputs (i.e. OUT_D:1, .. , OUT_D:8 for OUT_V='1') from the SIMULINK model and the FS outputs from the MATLAB implementation of the two-stage OSPFB using the same quantized inputs, same quantized coefficients, same shifting and scaling factors and final requantization are shown in Figure 7-3. Note that the MATLAB model uses floating point arithmetic in its implementation of multiplications and accumulations (i.e. no internal quantization). As shown in Figure 7-3, the maximum error between the FSs evaluated with MATLAB and SIMULNK models is ± 0.0078125 (i.e. $\pm 2^{-(8-1)}$)[16]. This error is due to the intermediate quantization stages implemented within the data path of the SIMULINK model. It has been observed that the means of errors between the FSs generated with MATLAB and SIMULINK models are essentially 0 (i.e. no bias in the quantization noise).

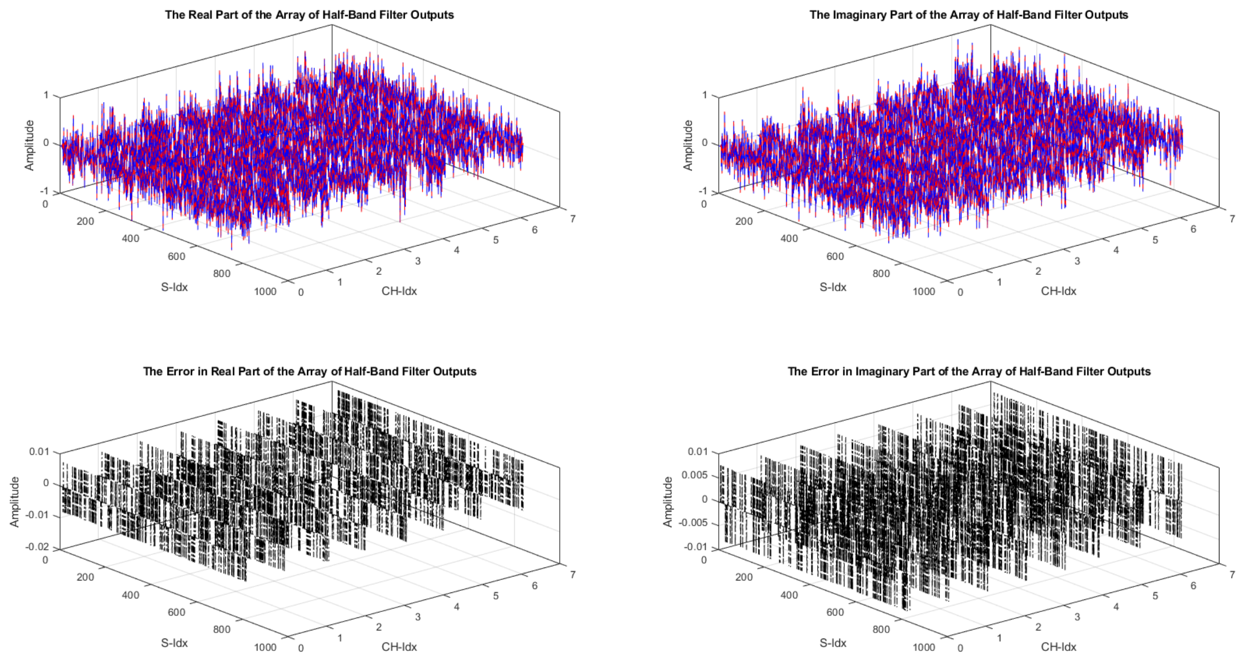

Figure 7-3 Comparison between re-quantized FS outputs from MATLAB implementation and SIMULINK implementation of the two-stage OSPFB for Gaussian distributed inputs.

The next step of verification is to compare the outputs between the SIMULNK model and the ModelSim simulation of the HDL implementation generated with the HDL files auto generated with DSP Builder for the same input stimuli. The DSP Builder provides the option of generating automatic test-benches and a straightforward way of executing the ModelSim simulations. The stimuli to the ModelSim simulation that contains the Gaussian-distributed inputs are shown in Figure 7-4. The expected outputs from the SIMULINK model and the signals generated by the ModelSim with the HDL code are compared in Figure 7-5. The cycle-by-cycle comparison shows that the outputs from the SIMULINKI model and the ModelSim simulation of the HDL implementation do perfectly agree.

[16] This is the quantization step for the ideal output quantizer with quantization levels (-127, -126, ... , -1, 0, 1, ... , 126, 127)/128 such that these levels can be uniquely expressed using 8 bits

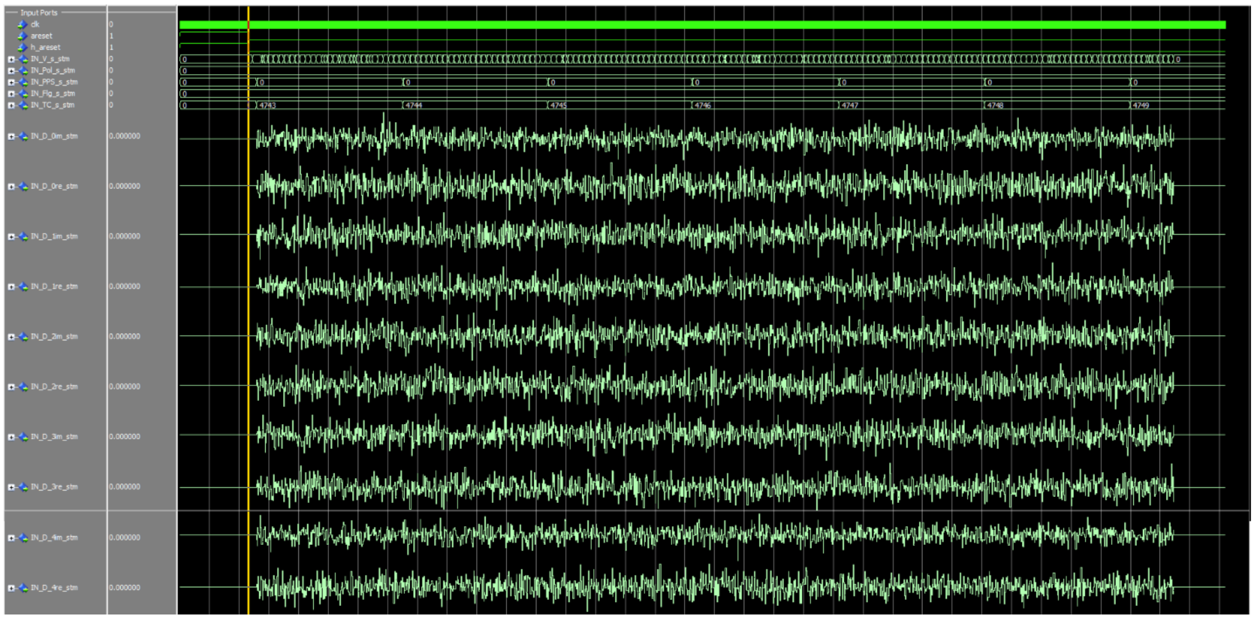

Figure 7-4 Stimuli for the ModelSim simulation – with Gaussian-distributed inputs.

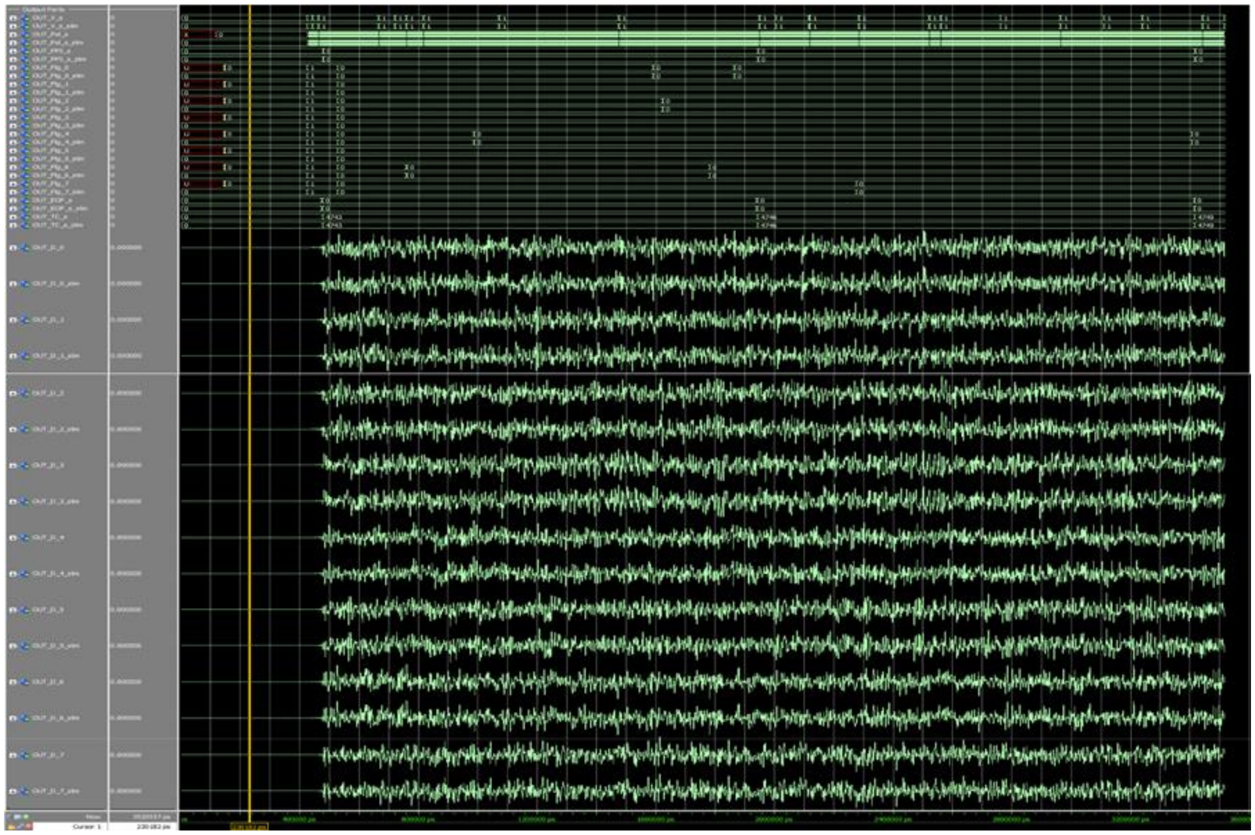

Figure 7-5 Comparison of responses from ModelSim simulation and the SIMULINK model for Gaussian-distributed inputs.

7.1.2 Impulsive Inputs with Flags

The objective of this test is to verify the time-integrity and flagging logic of the OSPFB implementation. Thus, four sparse impulse inputs were used as markers of time in IN_D. In particular two of these impulsive inputs are placed to coincide with the first and fourth PPS markers, respectively. The other two were randomly placed. In order to test the input flag-extension logic, the input flags were set (IN_Flg='1') to align with the impulsive inputs. The other input stimuli IN_V, IN_Pol, IN_PPS and IN_TC were generates as in the previous test. Also, the same configuration of coefficients and the scale factors (i.e. 65535/65536) for the FSs were maintained in the previous test (see Section 7.1.1). However, the shifting factor for all FSs have been selected as 3. The input stimuli for the SIMULINK model, except IN_TC, are shown in Figure 7-6

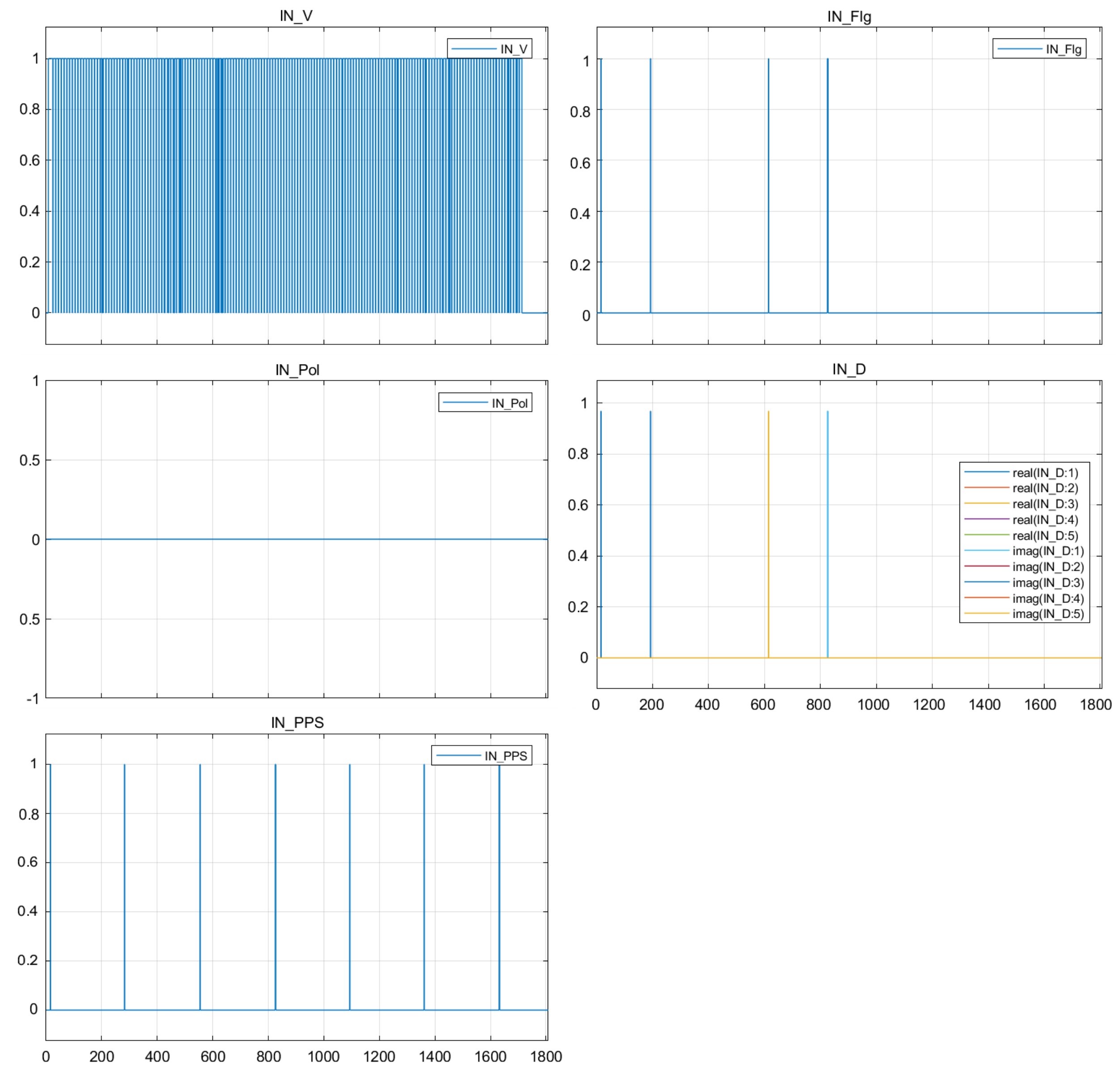

Figure 7-6 Stimuli (Except IN_TC) containing impulses fed to the SIMULINK model of the OSPFB.

The corresponding outputs from the SIMULINK model are shown in Figure 7-7. By observation, it has been confirmed that the peak of the outputs corresponding to the impulse inputs that coincide with the first and fourth PPS coincide with the output PPSs and thereby confirms the time-integrity of the OSPFB implementation. Also, it has been confirmed by observation that for all FSs the output flag remained OUT_Flg:1:8='1' for the 27 valid output cycles that encloses the non-zero data outputs (OUT_D:1-8) corresponding to the input impulses. These data values agree with the outputs evaluated with the MATLAB implementation within the quantization limit as shown in Figure 7-8. The responses from the HDL implementation for the same stimuli (see Figure 7-9) were investigated through the ModelSim simulation and are shown in Figure 7-10. These responses too found to be identical to the corresponding signals of the SIMULINK model.

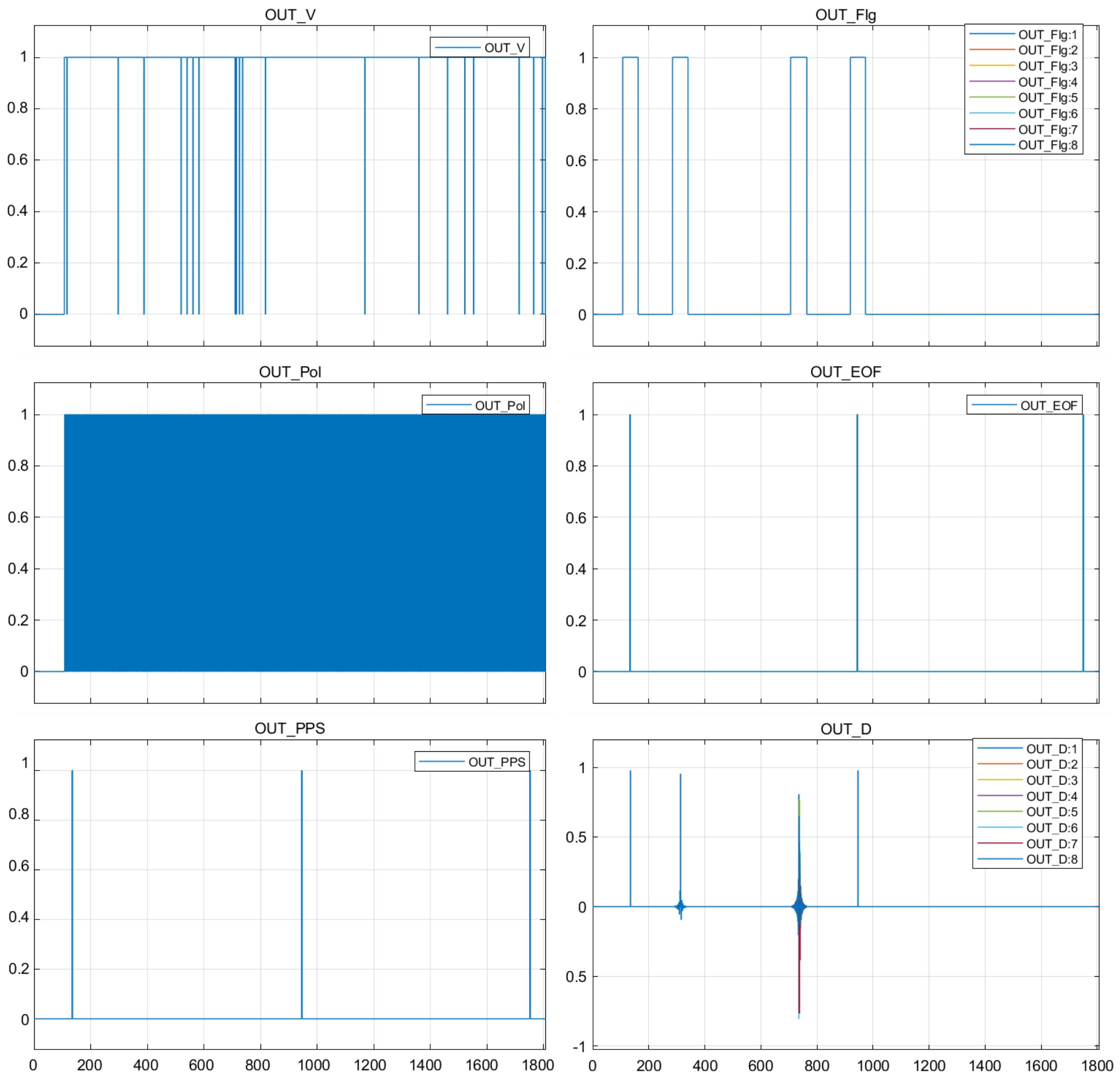

Figure 7-7 Responses (Except OUT_TC) from the SIMULINK model of the OSPFB in response to stimuli shown in Figure 7-6.

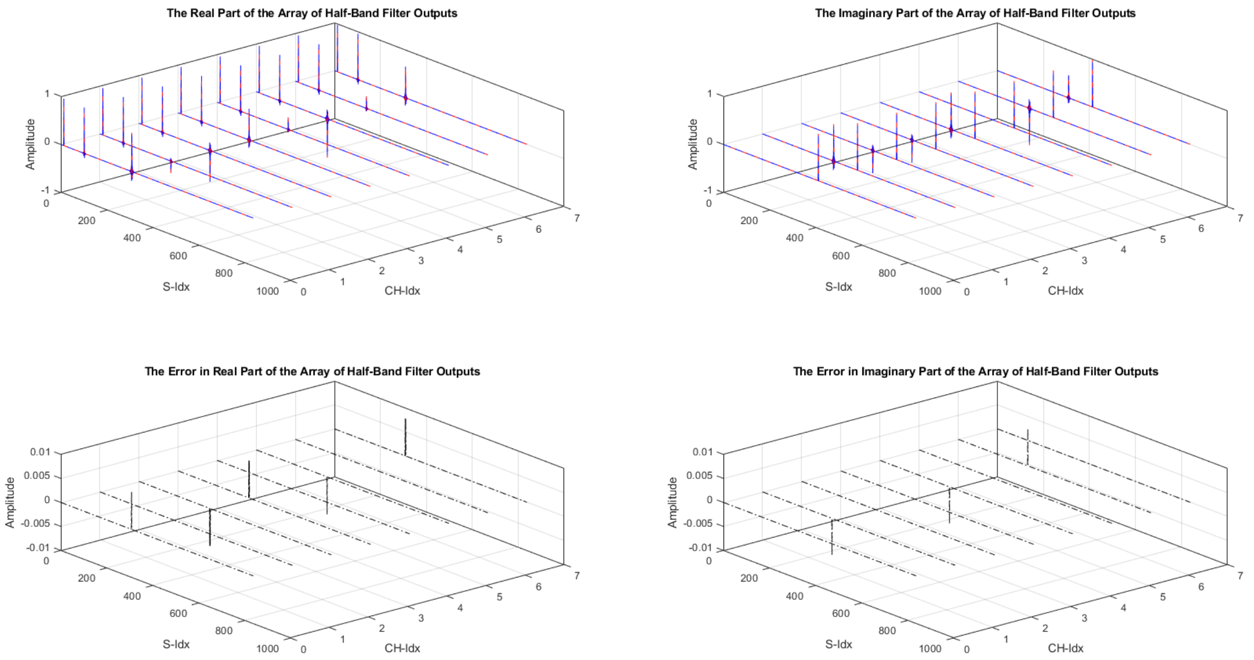

Figure 7-8 Comparison between re-quantized FS outputs from MATLAB implementation and SIMULINK implementation of the two-stage OSPFB for impulsive inputs.

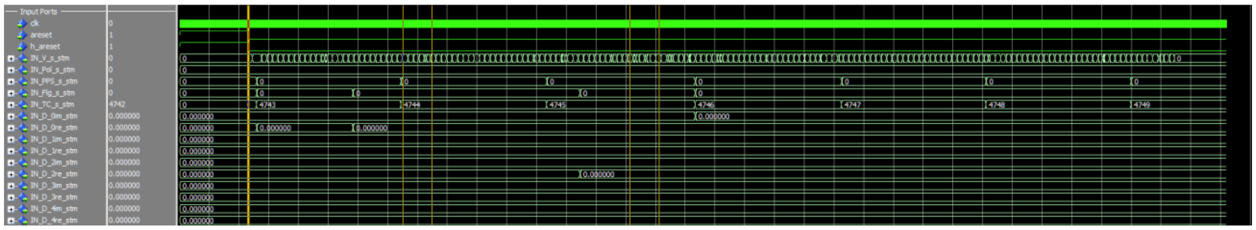

Figure 7-9 Stimuli for ModelSim simulation – with impulsive inputs.

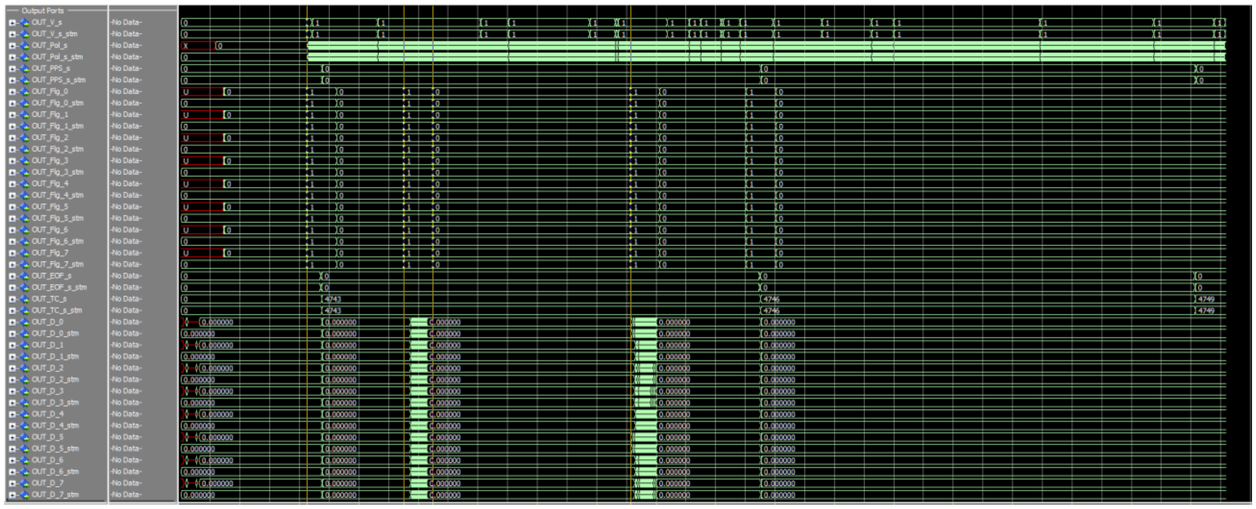

Figure 7-10 Comparison of responses from ModelSim simulation and the SIMULINK model - for impulsive inputs.

7.1.3 Sum of Sinusoids with Different Magnitudes

The objective of this last verification test is to confirm that the 'shifting' and 'scaling' factors are applied properly to the FSs before the final re-quantization step. Here, the combination of 8 complex-valued sinusoids with magnitudes and frequencies as specified in Table 7-1 have been selected to be the input stimulus. Note that the frequencies are randomly selected such that only one sinusoid falls within each of the FSs and the magnitudes have been randomly selected such that the amplitude of the real- and imaginary components of the combined signal remain within (-1, 1) (see IN_D panel of Figure 7-11). The selected shifting factors and the scaling factors for the FSs of the OSPFB are specified in Table 7-1. The other configurations and stimuli generation were made as for the test described in Section 7.1.1. The corresponding responses from the SIMULINK model are shown in Figure 7-12.

Table 7-1 Properties of the 8 sinusoids used in combination as the input stimulus.

OSPFB CH#	Relative Frequency†	Magnitude of the Sinusoid	Shifting Factor	Scaling Factor
1	0.082111621353251	0.124382648120286	2	0.951583862304688
2	0.237655026213343	0.099048965283659	3	0.801895141601563
3	0.297961560819441	0.139384578661599	2	0.815170288085938
4	0.371235319422669	0.196146024474188	2	0.803344726562500
5	0.481733387551645	0.156195908852844	2	0.827102661132813
6	0.629636952628889	0.088388347648318	3	0.962005615234375
7	0.710631159361382	0.110995371955572	2	0.906890869140625
8	0.814723958736016	0.175034872412951	2	0.829406738281250

†Frequency of the sinusoid / 2 GHz (the equivalent of the sample rate)

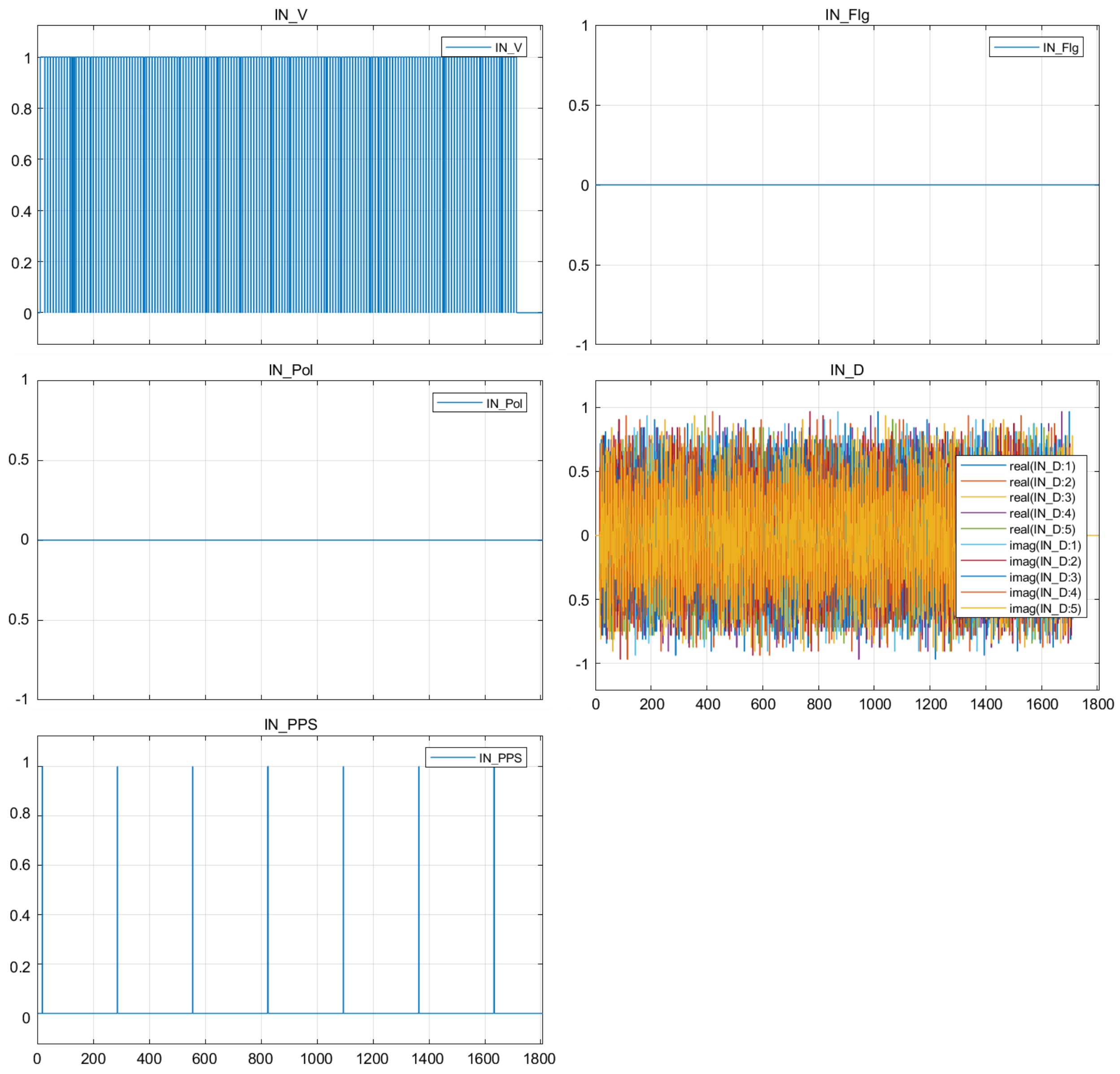

Figure 7-11 Stimuli (Except IN_TC) containing sum of 8 sinusoids fed to the SIMULINK model of the OSPFB.

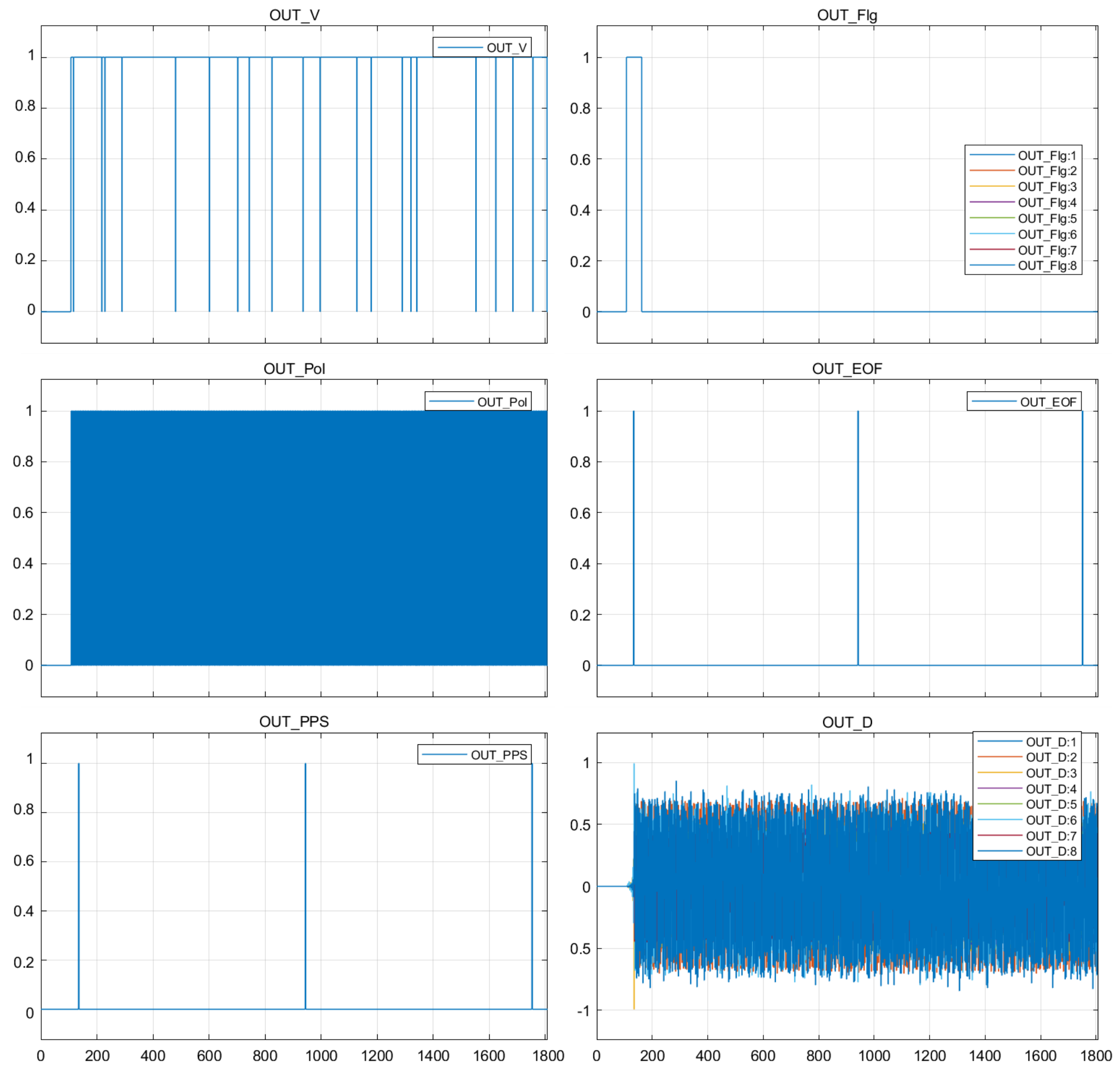

Figure 7-12 Responses (Except OUT_TC) from the SIMULINK model of the OSPFB in response to stimuli shown in Figure 7-11.

Responses from the MATLAB implementation and the SIMULINK model agree within the quantization error as shown in Figure 7-13. Also, responses from the HDL implementation simulated with ModelSim for the same stimuli shown in Figure 7-14 agree perfectly with the corresponding outputs of the SIMULINK model as shown in Figure 7-15.

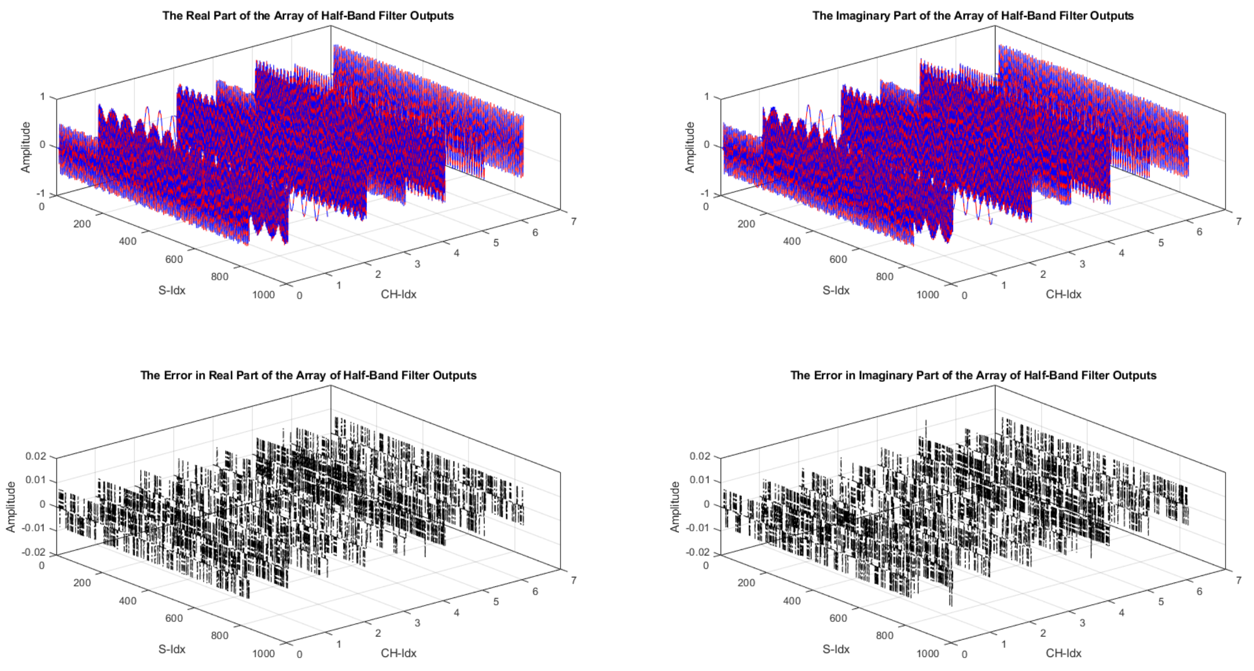

Figure 7-13 Comparison between re-quantized MATLAB implementation and SIMULINK implementation of the two-stage OSPFB for sum of sinusoids inputs.

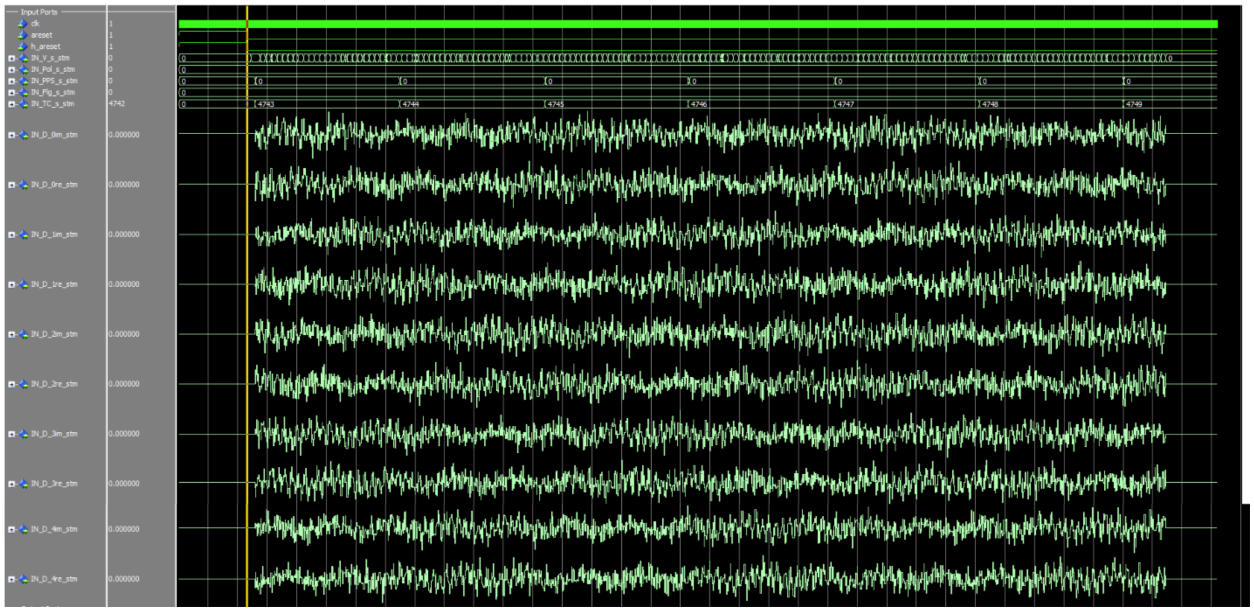

Figure 7-14 Stimuli for ModelSim simulation - with sum of sinusoids.

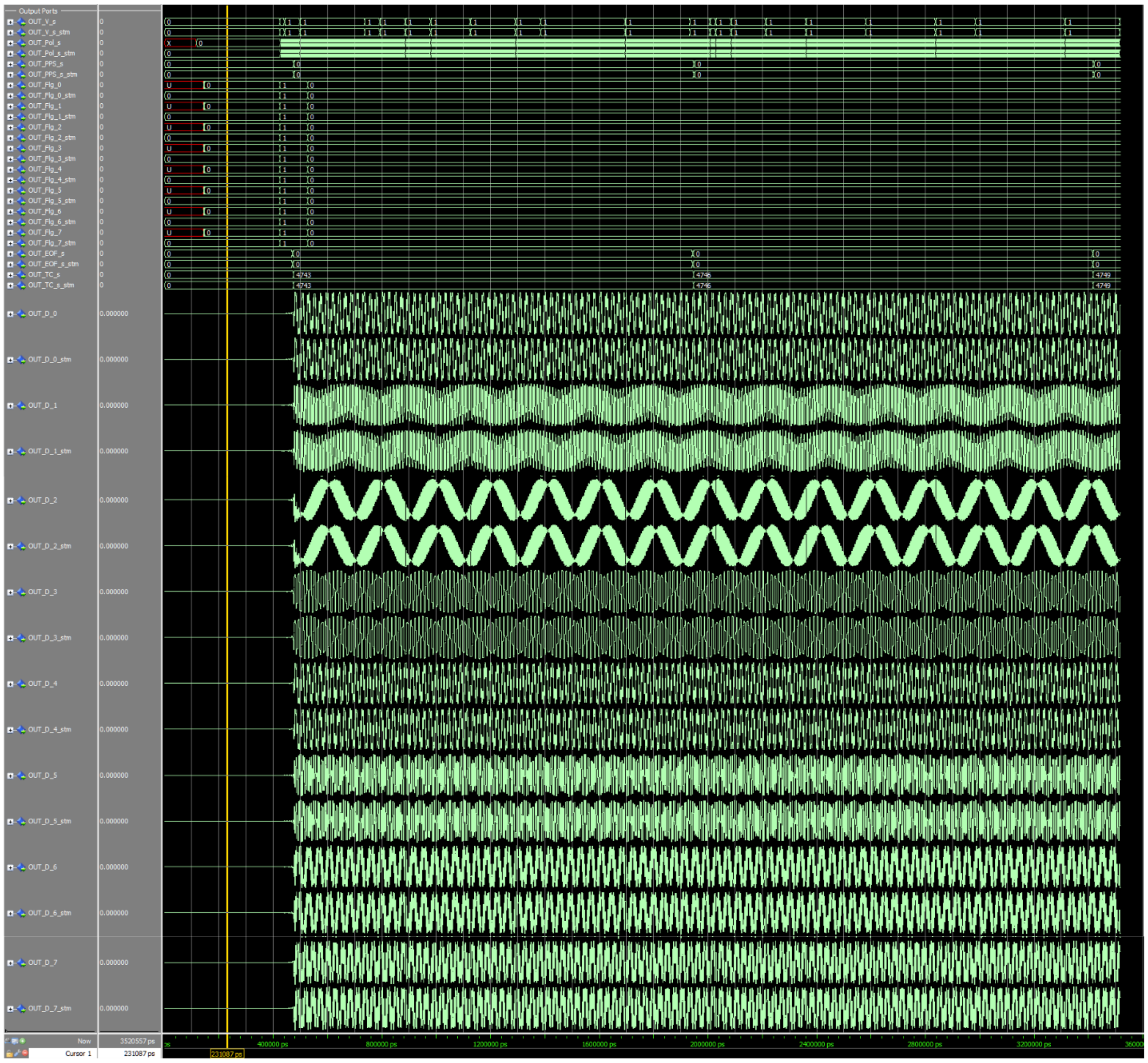

Figure 7-15 Comparison of responses from ModelSim simulation and the SIMULINK model – for sum of sinusoids.

7.2 Synthesis of 20 Instances of OSPFBs and other Key Firmware Blocks

The main objective of this study is to confirm that the proposed AVCC FPGA is capable of processing a total of 32 GHz of bandwidth contained within 20 sub-bands. The ideal way of achieving this is to complete a full FPGA design and verification for the proposed AVCC FPGA. However, it is highly improbable to complete such a design within this 3 month study requiring the custom design of the OSPFB firmware block and at least one of other supporting firmware blocks (i.e. Wideband Input Buffer (WIB) firmware block). Therefore, a ‘strawman’ design of the AVCC FPGA that consisted of almost all the required logic that is expected to be in the full FPGA design of the AVCC.

The architecture of the strawman AVCC FPGA is shown in Figure 7-16. As shown, there are six 100 Gigabit Ethernet MACs configured without forward error correction (FEC), six Intel’s ‘Sync-FIFO’ firmware blocks to imitate the WIB firmware blocks, 20 OSPFB firmware blocks, two 80×80 Circuit Switch with 12-bit wide bus and some glue-logic interconnecting the Sync-FIFOs with the 20 OSPFBs. Forty Serial Lightweight Interconnect Mesh (SLIM) blocks, required to package the 160 FSs (80 FS x 2 polarizations) to be sent to the AFSP FPGAs, are not included in this strawman design since logic utilization of SLIM blocks is very low and it was judged that the benefit of instantiating and testing them within the timeframe of the study wasn’t warranted.

Out of six 100 Gigabit Ethernet MACs, four were instantiated with hard MACs and the remaining two have soft MACs. Each of the Sync-FIFOs is 512-bits wide and 8192-words deep. The custom-developed 80x80 Circuit Switch firmware block is based on a cross-bar interconnect. Even though not shown in Figure 7-16, the DeTrI interconnect has been instantiated and configured to connect with the control registers of the 100 Gigabit Ethernet MAC. The received data from the six 100 Gigabit Ethernet MACs are carried by a bus of width 512-bits to the six Intel Sync-FIFOs. Some simple logic functions were used to separate 6x512 Sync-FIFO outputs in to 4x180-bit (3 streams) + 2x240-bit (4 streams) and then concatenated into a 1200-bit bus that feeds the input data-frames to the 20 OSPFBs. These logic functions also generated the control signals to the OSPFBs. The 160 FSs and control signals from the 20 OSPFB have been divided into two groups of 80 FSs and fed to the two Circuit Switches. The unconnected inputs of the six 100 Gigabit Ethernet MACs, the outputs of the Circuit Switches and the DeTrI interface are assigned to virtual pins to avoid being synthesized away.

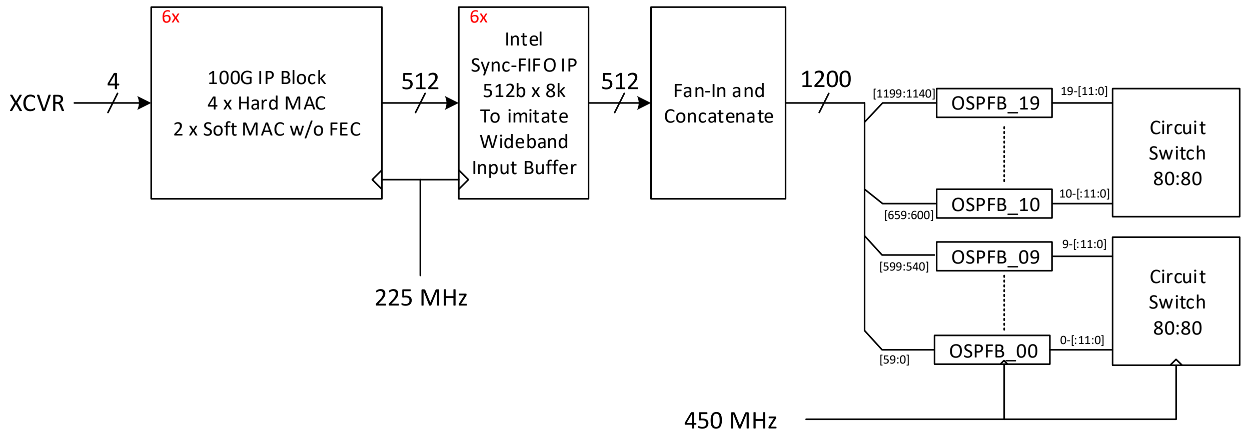

Figure 7-16 Strawman architecture for the AVCC FPGA.

As shown in Figure 7-16, the six 100 Gigabit Ethernet MACs and the Sync-FIFOs operate at the clock rate at 225 MHz whereas the 20 OSPFBs and the two Circuit Switches operate at the clock rate of 450 MHz. On the other hand the DeTrI interface operate at the clock rate of 250 MHz. Note that there are other clocks generated inside the six 100 Gigabit Ethernet MACs to facilitate the internal/external signal-transfer. Paths from other clocks to the DeTrI clock are 'false' paths and therefore timing analysis need not to be conducted. The timing paths between the Sync-FIFOs and the 20 OSPFBs are configured as synchronous multi-cycle paths for the purposes of timing evaluation. In the actual design the Sync-FIFOs will likely operate in a 200 MHz clock domain (contrary to that stated in Section 7.2.3), asynchronous to the 450 MHz OSPFB clock domain, requiring short simple dual-port dual-clock FIFOs implemented in ALMs between them. Synthesis indicates that ~5k ALMs are required for these FIFOs.

The synthesis for the strawman design of the AVCC FPGA was conducted using the Intel Quartus compilation flow [14] with the above mentioned configurations targeting the Intel Stratix-10 1SX280HU2F50E1VG FPGA. Key estimates of resource utilization and timing analysis are given in the following sub-sections.

7.2.1 Resource Utilization Summary

Table 7-2 Summary for resource usage from the Fitter of the Intel Quartus compilation flow.

Resource	Usage	%
Logic utilization (ALMs needed / total ALMs on device)	466,446 / 933,120	50 %
ALMs needed [=A-B+C]	466,446	
[A] ALMs used in final placement [=a+b+c+d]	588,050 / 933,120	63 %
[a] ALMs used for LUT logic and register	174,975	
[b] ALMs used for LUT logic	48,716	
[c] ALMs used for register circuitry	350,179	
[d] ALMs used for memory (up to half of total ALMs)	14,180	
[B] Estimate of ALMs recoverable by dense packing	134,591 / 933,120	14 %
[C] Estimate of ALMs unavailable [=a+b+c+d]	12,987 / 933,120	1 %
[a] Due to location constrained logic	0	
[b] Due to LAB-wide signal conflicts	402	
[c] Due to LAB input limits	819	
[d] Due to virtual I/Os	11,766	
Difficulty packing design	High	
Total LABs: partially or completely used	70,759 / 93,312	76 %
-- Logic LABs	69,341	
-- Memory LABs (up to half of total LABs)	1,418	
Combinational ALUT usage for logic	393,736	
-- 8 input functions	2,032	
-- 7 input functions	1,510	
-- 6 input functions	33,305	
-- 5 input functions	108,603	
-- 4 input functions	26,205	
-- <=3 input functions	222,081	
Combinational ALUT usage for route-throughs	468,271	
Memory ALUT usage;	21,314	

-- 64-address deep	0	
-- 32-address deep	21,314	
Dedicated logic registers	1,330,849	
-- By type:		
-- LAB logic registers:		
-- Primary logic registers	1,050,307 / 1,866,240	56 %
-- Secondary logic registers	258,164 / 1,866,240	14 %
-- Hyper-Registers:	22,378	
Register control circuitry for power estimation	0	
ALMs adjustment for power estimation	31,669	
I/O pins	111 / 1,152	10 %
-- Clock pins	9 / 88	10 %
-- Dedicated input pins	54 / 198	27 %
M20K blocks	1,150 / 11,721	10 %
Total MLAB memory bits	530,384	
Total block memory bits	19,844,288/240,046,080	8 %
Total block memory implementation bits	23,552,000/240,046,080	10 %
DSP Blocks Needed [=A+B-C]	1,800 / 5,760	31 %
[A] Total Fixed Point DSP Blocks	2080	
[B] Total Floating Point DSP Blocks	0	
[C] Estimate of DSP Blocks recoverable by dense merging	280	
IOPLLs	0 / 24	0 %
Global signals	34	
Impedance control blocks	0 / 24	0 %
Maximum fan-out	1155131	
Highest non-global fan-out	233186	
Total fan-out	5370259	
Average fan-out	3.00	

Utilization of DSP Blocks are as expected, however utilization of the M20K memory blocks are lower than expected. Note that the total resource usage is at comfortable 50% level.

The placement of the logic resources for key firmware blocks are shown in four panes of the Figure 7-17: (A) the six 100G Ethernet MACs; (B) the six Sync- FIFOs (C) 20 OSPFB firmware blocks and (D) two 80x80 Circuit Switch firmware blocks.

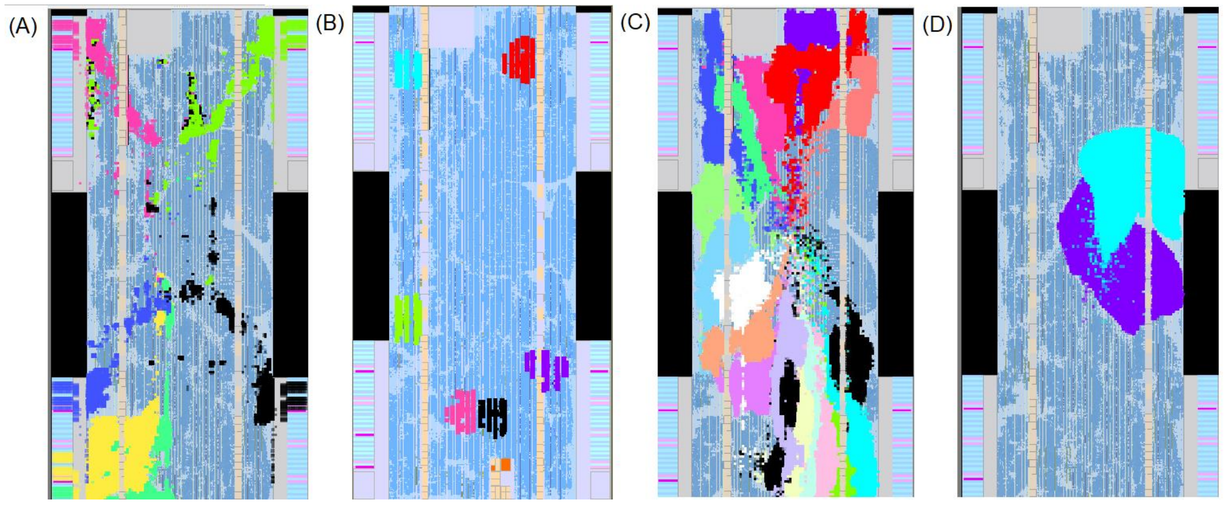

Figure 7-17 Multiple views showing placement of logic resources on the chip for the key firmware blocks. (A) the six 100G Ethernet MACs; (B) the six Sync- FIFOs (C) 20 OSPFB firmware blocks and (D) two 80x80 Circuit Switch firmware blocks.

7.2.2 Routing Congestion

Another key aspect of the AVCC FPGA implementation is the utilization of routing resources connecting logic resources. The 'long' wires are primarily used in timing-critical paths as opposed to 'short' wires that are used to connect non timing-critical paths. For this particular synthesis of the AVCC FPGA the utilization of long and short wires are shown inFigure 7-18 left and right panes, respectively. If unavailable, long wires can be replaced with short wires but with obvious penalty in meeting the timing closure, requiring additional pipelining to resolve.

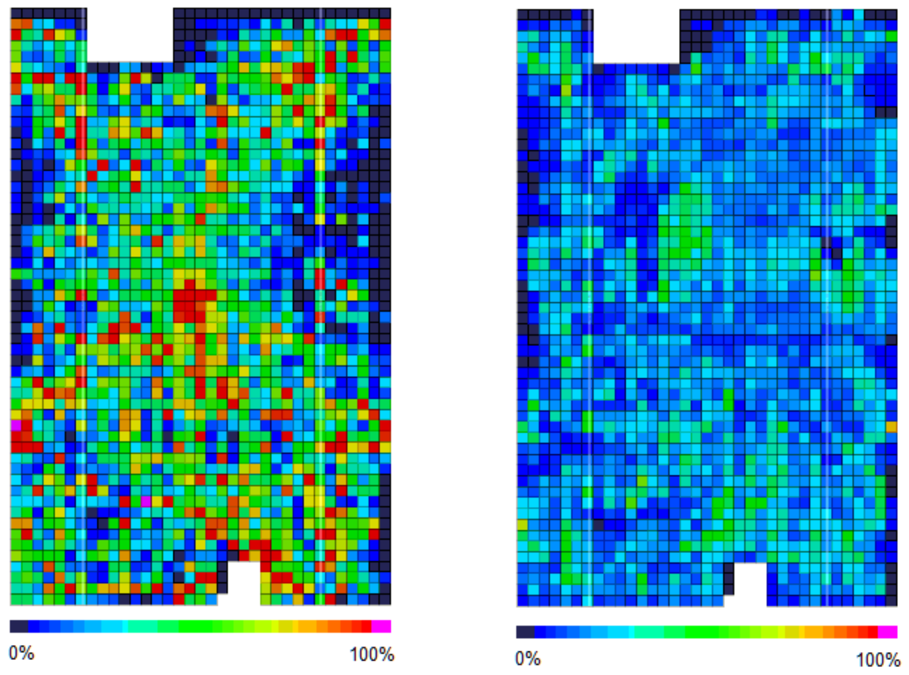

Figure 7-18 Utilization of 'long' (left) and 'short' (right) wires for routing in the AVCC FPGA.

7.2.3 Static Timing Analysis and F_{max} Estimates

The AVCC FPGA design requires 4 main clock signals to drive the internal firmware blocks. Those clocks and their expected rates are given in the following.

1. 'clk' – 450 MHz : The clock that drive the OSPFB and the Circuit Switch firmware blocks.
2. 'fifo_clk' – 225 MHz : The clock that drive the Sync-FIFOs mimicking the WIBs.
3. 'detri_clk' – 250 MHz : The clock that drives the DeTrI interconnect.
4. 'divided_osc_clk' – 125 MHz: The clock to drive the 100G Ethernet MACs.

The rising clock-edge of 'fifo_clk' is synchronized with every other rising clock edge of 'clk'. Also, the 100G Ethernet MACs generates number of additional clocks internally from 'divided_osc_clk'. The static timing analysis conducted for a particular synthesis of logic implementation on an FPGA estimates the maximum possible clock rate (F_{max}) that the associated firmware blocks can operate at. When the change of the output of a register is not expected to be captured by a destination register within the given clock-interval the connection between those two registers it is called a 'false-path' and therefore, need not be considered for timing analysis. Declaration of false-paths not only allows a clear interpretation of timing analysis reports but also better routing options. In the implementation of the AVCC FPGA a number of false-paths have been identified. Particularly those are the paths from the registers associated with the DeTrI interconnect, which update the configuration registers firmware blocks. For this particular synthesis of the AVCC FPGA, the main clock signals, the required clock-rates and estimated F_{max} and the worst-case operating condition that corresponds the F_{max} estimate are listed in Table 7-3. As given there, the corresponding F_{max} estimates for the main clocks of the AVCC FPGA comfortably exceeds the required clock rates. The same parameters for the derived clocks for 100G Ethernet MACs are listed in Table 12-1. According to that the corresponding F_{max} estimates derived clocks also comfortably exceeds the required clock-rates.

Table 7-3 The main clocks needed for AVCC FPGA, required clock rates, F_{max} estimates and worst-case operating condition.

Clock Name	Required Clock Rate (MHz)	F_{max} Estimate (MHz)	Worst-Case Operating Conditions
fifo_clk	225	246.97	1 Slow vid1 100C Model
clk	450	470.15	1 Slow vid1 100C Model
divided_osc_clk (ALTERA_INSERTED_INTOSC_FOR_TRS)	125	163.8	Slow 900mV 100C Model

8 Risks

We estimate that there is less than 1% risk that the instantiation of 40 SLIM firmware blocks increase the routing congestion to where the required clock rates not be achieved.

9 Conclusions

This 3 months long study leads us to conclude with ~95% confidence that the proposed AVCC FPGA can process 32 GHz of input bandwidth carried by 20 sub-bands in six 100G Ethernet links.

This study was conducted in two phases. In the first phase, a custom 10-channel OSPFB to segment a 2.0 Gs/s - 1.6 GHz sub-band into eight 200 MHz FSs has been designed and implemented. Here, the design and functional verification of the 10-channel OSPFB firmware block was carried out using Intel 'DSP Builder' [11] blocks within the model-based design environment provided by Simulink/MATLAB [12]. Multiple runs of the compilation-flow in Quartus Prime Pro edition [14] were also conducted with just a single instantiation of the 10-channel OSPFB firmware block. Iterative design changes were made until the block was able to run at ~150 MHz over the required operational clock rate of 450 MHz.

The second phase of the study was dedicated to determine whether the AVCC FPGA can run at the required speed with required firmware blocks to process 32 GHz of bandwidth. Here, 20 OSPFBs, six 100G Ethernet MACs, six Sync FIFOs (mimicking WIBs) and two 80×80 Circuit Switch firmware blocks were instantiated in a Stratix-10 1SX280HU2F50E1VG FPGA. The compilation flow in Quartus Prime Pro edition [14] was conducted with timing constraints; the design was iteratively modified until timing closure was achieved with a conformable margin.

10 References

[1] T. Gunaratne and B. Carlson, "Examine if the NRC ALMA Very Coarse Channelizer FPGAs Can Handle 16 GHz per sideband per polarization", ALMA development Cycle 9 Study Proposal, 2021-10-05.

[2] G. H. Tan, T. Kojima, T. Hunter and T-C. Shen, Status Review of Baseline Proposals for the ALMA Wideband Sensitivity Upgrade Panel Report (ALMA-05-00.00.00-0055 -A-REP), Version A, 2021-10-15.

[3] J. Carpenter, et. al., "The ALMA Development Roadmap", ALMA Memo 612, AEDM 2018-017-O.

[4] S. Asayama, G. H. Tan, K. Saini, J. Carpenter, G. Siringo, T. Hunter, N. Phillips, H. Nagai, "Report of the ALMA Front-end & Digitizer Requirements Upgrade Working Group", ALMA-05.00.00.00.0048-A-REP, Ver. B, 16 Dec. 2020

[5] C. Brogan et al. "Specifications for a Second-Generation ALMA Correlator", ALMA-05.00.00.00-0049-A-SPE, [Online] https://science.nrao.edu/facilities/alma/science_sustainability/Specifications2ndGenCorrelatorV2.pdf

[6] B. Carlson, M. Pleasance, T. Gunaratne and S. Vrcic, "Study Report: NRC TALON Frequency Slice Architecture Correlator Beamformer (AT.CBF) for ALMA", ALMA Memo 617, 2020/09/21; https://library.nrao.edu/public/memos/alma/main/memo617a.pdf.

[7] B. Carlson, "TALON Frequency Slice Architecture Correlator/Beamformer for ALMA", Cycle 9 NA ALMA Development Project Proposal, 2021-04-09.

[8] B. Quertier, S. Gauffre, A. Randriamanantena and M. Studniarek, "Upgrading the ALMA Digital System, from Digitization to Correlation Final Report," ALMA Development Studies 2019 CFP/ESO/19/25417/HNE, July 16th, 2021.

[9] M. Pleasance, TALON Asynchronous – Mid.CBF Timing and Synchronization without Central Timing Reference, TALON SKA1 MID.CBF Memo 0017, 2021-03-12.

[10] Proto-Interface Control Document (P-ICD) between the Updated LAB ALMA Digitizer and TALON ALMA Correlator, 2021-11-02.

[11] DSP Builder for Intel® FPGAs, [Online] https://www.intel.ca/content/www/ca/en/programmable/support/support-resources/intellectual-property/dsp/dsp-builder/ips-dsp-builder.html

[12] Simulink is for Model-Based Design, [Online] https://www.mathworks.com/products/imulink.html

[13] ModelSim, [Online] https://eda.sw.siemens.com/en-US/ic/modelsim/

[14] Intel® Quartus® Prime Software Suite, [Online] https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/overview.html

[15] Fredric j. Harris, Multirate Signal Processing for Communication Systems, Ed 2 , S.l.: River Publishers, 2021.

[16] P. P. Vaidyanathan, Multirate Systems and Filter Banks, Englewood Cliffs, N.J: Prentice Hall, 1992.

[17] Device Tree Interconnect (DeTrI). [Online] https://gitlab.drao.nrc.ca/SKA/util/DeTrI.

[18] R. E. Blahut, Fast Algorithms for Signal Processing. Cambridge University Press, 2010.

11 Appendix—Preference between Sub-Bands of 1.6 GHz and 2.0 GHz of Bandwidth

As stated in Section 2 - Background, initially, the NRC and LAB teams agreed to consider the following two formats for the sub-bands, each resulting in 16 GHz of usable bandwidth per sideband per polarization:

Option 1: Each sub-band sampled at 2.0 Gs/s containing 1.6 GHz (3.2 GHz DSB) of usable bandwidth or

Option 2: Each sub-band sampled at 2.5 Gs/s containing 2.0 GHz (4.0 GHz DSB) of usable bandwidth,

both consisting of complex-valued sample streams at 6 + 6b resolution [10]. In order to extract FSs containing 200 MHz of bandwidth and to have an oversampling factor of oversampling = 10/9, it requires either:

Option 1: A 10-Channel OSPFB with up-sampling factor of 2, processing the sub-bands at 2 Gs/s or

Option 2: A 25-channel OSPFB with up-sampling factor of 8, processing the sub-bands at 2.5. Gs/s.

Though the scope and funding for this development study was designed to facilitate only one complete AT.CSP AVCC-stage design, we allocated the first few weeks of the study to examining the basic aspects of these two options. **We found that based on three important considerations, that Option (1) 2.0 Gs/s containing 1.6 GHz (3.2 GHz DSB) of bandwidth is advantageous compared to Option (2) 2.5 Gs/s containing 2.0 GHz (4.0 GHz DSB) of bandwidth.**

11.1 Over-Sampled Polyphase Filter Bank Resources

The different design configurations for the two-stage OSPFBs for the two options and the estimates for the required number of 18×18-bit multipliers are listed in Table 11-1. We estimate that the two-stage OSPFB for the 2.5 Gs/s option (2) requires 12.3% more multipliers than the 2.0 Gs/s option (1).

Table 11-1 The design configurations and estimates of required multiplier for the two-stage OSPFBs to extract FSs from sub-bands sampled at 2.0 Gs/s and 2.5 Gs/s.

Design Configuration	**#Mults for FIR Filters of an OSPPFB**	**#Mults for the Parallel IFFT of an OSPPFB**	**#Mults for the Half-Band Filters**	**#Instances to Process 32 GHz**	**#Mults for Scaling 160 FSs**	**Est: Total #Mults**
10-Channel OSPFB Sample Rate = 2.0 Gs/s Up-Sampling Factor = 2 IFFT-Points = 10	60	16	8x12	20	320	3760†
25-Channel OSPFB Sample Rate = 2.5 Gs/s Up-Sampling Factor = 8 IFFT-Points = 25	76	48	10x12	16	320	4224

† These 3760 18×18-bit multipliers are packed in 2080 DSP Blocks in the implementation (see Table 7-2).

Further, it is expected that more logic resources would also be needed to implement the 8:1 multiplexing needed to feed the coefficients into the multipliers in the polyphase FIR filters in the 25-channel OSPFB as opposed to the 2:1 multiplexing (see Section 4.1.2) required for the 10-channel OSPFB, though the factor cannot be easily estimated without a full design study.

11.2 Logic and Mapping for AVCC Data Ingest

Whilst this study has included analysis of the FPGA resources for 6 x 100GbE ingest ports, 6+6b, the requisite ingest data rate for 16 GHz x 2 polarizations (20 x 2 Gsps x 6+6b = 480 Gbps) will fit on 5 x 100GbE ports at 480/500 = 96% utilization (with further IEEE 802.3 and other framing overhead the utilization would be 98.6%.)

For Option 1, there are 10 x 2.0 Gs/s sub-bands per polarization. Five x 100GbE ports maps easily to 20 sub-band channelizers; each 100GbE port contains the packets for 4 sub-band channelizers (2 sub-bands x 2 pol's each). What this means in practice is that, at the digitizer end, each IEEE 802.3 Ethernet packet contains a contiguous time-series of data for one dual-pol sub-band (or a single pol sub-band if desired), for a particular period of time, with destination MAC addressing so that it ends up at a single AVCC 100GbE port. Packets for 2 dual-pol sub-bands contain the same destination MAC address, so that one AVCC 100GbE port only gets packets destined for 4 specific sub-band channelizers.

Inside the AVCC FPGA each 100GbE "FPGA fabric" port is typically 512 bits wide at a ~200 MHz clock rate[17]. At this 512-bit wide 100GbE receiver FPGA fabric port feeding 4 x sub-band channelizers, for each packet, routing of packet data to the correct sub-band channelizer is simply a matter of a) fanning out the 512-bit bus to all sub-band channelizer ingest ports (each having an ingest FIFO) and b) controlling the data valid signal so that only the target channelizer ingest port "gets" the packet data. This is a very simple FPGA implementation.

If, on the other hand, there are 16 x 2 GHz sub-bands (Option 2), this means that either 5 or 6 100GbE ports must map to 16 sub-band ingest ports, i.e., a mapping of 5 or 6 to 16, neither of which is as simple as described above. Although this more complex mapping could be handled in FPGA logic, it is not clear at this time how much logic would be needed and if the required logic pushes the FPGA resource utilization into uncomfortable territory (beyond the 12.3% already described in Section 11.1). Thus, the more complex mapping for AVCC data ingest for Option 2 represents an unquantified risk and certainly additional firmware costs to implement compared to Option 1.

11.3 Distribution of 200 MHz AVCC Frequency Slices

The observation science bandwidth of the sub-bands sampled at either 2.0 Gs/s or 2.5 Gs/s would be distributed symmetrically between ± 800 MHz or ± 1000 MHz as shown in Figure 11-1 (top). However, this distribution is sub-optimal for segmenting into FSs containing 200 MHz of bandwidth as the direct

[17] This is because IEEE 802.3 packets are based on byte octets, each octet forming a 64-bit word, with typically 8 x 64-bit words forming a 512-bit "frame" and multiple such frames forming an IEEE 802.3 packet. Packing of 6+6b (i.e. 12b) samples within in a 512-bit frame is not perfect, for example in the SKA1 Mid dish-to-correlator protocol, each 512-bit frame contains 21 dual-pol 12-bit samples, for an efficiency of 504/512~=98.4%, but nevertheless with simple and exact mapping of 12b samples to 512-bit wires.)

channelization with an OSPFB results in two FSs at the opposite ends containing just 100 MHz of bandwidth. Hence, the spectrum of the sub-bands need to be 'shifted' by ±100 MHz as shown in Figure 11-1 (bottom) such that the edges of the observation bandwidth are at odd-multiples of 100 MHz. This shift may require a time-demultiplexed digital modulator that consumes a non-trivial amount of DSP and memory resources in the AVCC FPGA. Therefore, the NRC team have asked the LAB team to consider implementing this frequency-shift as a part of their sub-band extraction channelizer and is currently under consideration by them.

Note that, if the sub-band sample rate is 2 Gs/s, there is a simple way to perform this shift. The complex-valued output sequence modulating with the sequence of 1, j, -1, -j, 1,… that requires no actual multipliers produces an equivalent frequency shift of 500 MHz. Subsequently, FSs generated in the two-stage OSPFB are selected and labeled accordingly. Note that this method will only work if the sub-band sample rate is 2 Gs/s and the science bandwidth contained within it spans between -800 MHz to +800 MHz. Therefore, even if the LAB team is unable to implement the desired frequency shifts in the sub-bands, having sub-bands at the rate of 2.0 Gs/s will allow the NRC team to implement the required logic to modulate the signal with 1, j, -1, -j, 1,… in the AVCC FPGA.

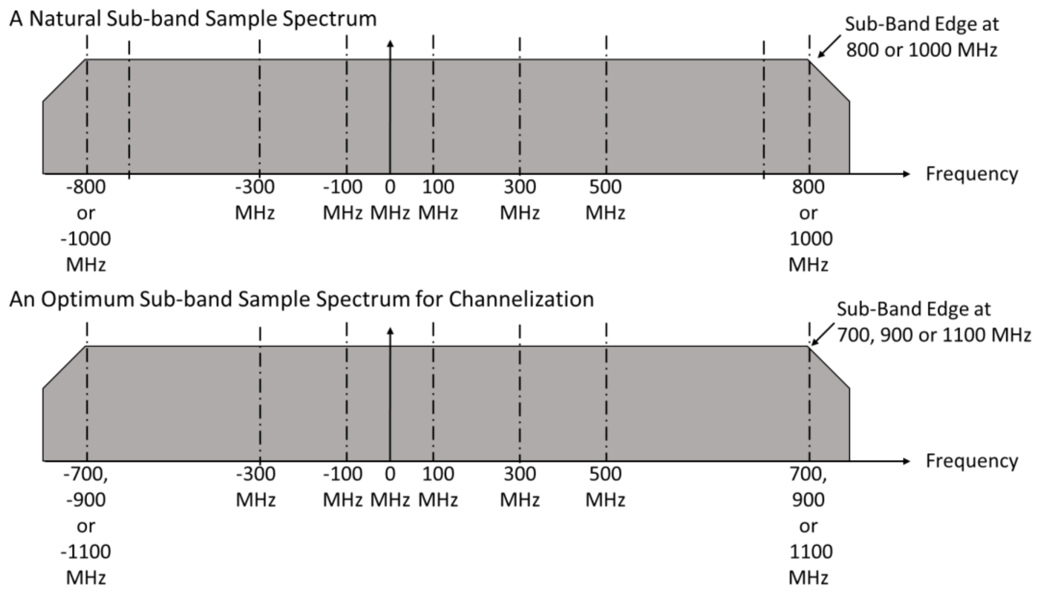

Figure 11-1 The natural distribution of bandwidth for sub-bands sampled at 2.0 Gs/s and 2.5 Gs/s containing 1.6 GHz and 2.0 GHz of bandwidth, respectively (top) and the optimum bandwidth distribution for segmentation into FSs containing 200 MHz of bandwidth (bottom).

Lower computational complexity and the ability to implement the required frequency shift with minimum addition of logic make the sub-bands at 2.0 Gs/s to be preferred over the sub-bands at 2.5 Gs/s.

12 Appendix—F$_{max}$ Estimates for the Internally Generated Clocks in 100G Ethernet MACs

Table 12-1 The derived clocks needed for the 100G Ethernet MACs in the AVCC FPGA, required clock rates, F$_{max}$ estimates and worst-case operating condition.

Derived clocks needed for the 100G Ethernet MACs	Req'd F (MHz)	F$_{max}$ (MHz)	Op condition
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch3	15.625	360.36	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[3].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	376.22	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch1	15.625	376.36	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[1].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	378.79	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	381.24	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[1].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	383.44	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch2	15.625	389.56	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	390.47	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[1].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	396.51	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[2].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	397.3	Slow 900mV 100C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[3].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	398.57	1 Slow vid1 0C Model

E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch0	15.625	403.23	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch3	15.625	403.39	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[3].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	407.17	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[3].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	411.35	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[2].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	413.56	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[2].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	415.45	Slow 900mV 100C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout|ch0	402.83	418.24	Slow 900mV 0C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	419.11	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch1	15.625	421.23	Slow 900mV 100C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|tx_clkout2|ch1	390.62	424.63	Slow 900mV 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout2|ch1	390.62	424.81	1 Slow vid1 100C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch2	15.625	424.99	Slow 900mV 100C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout|ch0	402.83	426.08	Slow 900mV 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|caui4_xcvr|recal_clk_div8|ch0 (aka s10_xcvr_native_phy|recal_clk_div8|ch0)	15.625	429.18	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[2].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	431.59	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|tx_clkout2|ch1	390.62	432.15	Slow 900mV 100C Model

E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout|ch0	402.83	436.87	Slow 900mV 0C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[1].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	438.98	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout|ch0	402.83	443.07	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout2|ch1	390.62	452.9	1 Slow vid1 100C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|s10_xcvr_native_phy|recal_clk_div8|ch0	15.625	463.61	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch3	402.83	773.4	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch1	402.83	805.8	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch0	402.83	820.34	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch1	402.83	828.5	Slow 900mV 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch3	402.83	833.33	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch2	402.83	835.42	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[5].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch2	402.83	838.93	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[4].E_EN100g|G_RTL_NO_FEC.E_ETH_MAC|alt_e100s10_0|xcvr|rx_clkout|ch0	402.83	892.06	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout2|ch0	390.62	1000.0	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout|ch0	402.83	1000.0	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout|ch0	402.83	1000.0	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout|ch0	402.83	1000.0	1 Slow vid1 0C Model
E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout2|ch0	390.62	1000.0	1 Slow vid1 0C Model

E_WRAPPED|Gen_EN100g_Array[1].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[0].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout|ch0	402.83	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[3].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|rx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model
E_WRAPPED|Gen_EN100g_Array[2].E_EN100g|G_HARD_MAC.E_ETH_MAC|alt_ehipc2_0|alt_ehipc2_hard_inst|altera_xcvr_native_inst|g_native_phy_inst[0].s10_xcvr_native_inst|tx_clkout2|ch0	390.62	1000.0	1 Slow vid10C Model