MOFC DIgital icd and idd

Version *<1.0>*

*<12/6/2024>*

VERSION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Version #** | **Written**  **By** | **Revision**  **Date** | **Reason** |
| 1.0 | *<Guy izkovksy>* | *<25/05/2022>* | First draft |
|  |  |  |  |
|  |  |  |  |

TABLE OF CONTENTS

[2 Purpose of this documnet 4](#_Toc105594085)

[3 Version 1.0 5](#_Toc105594086)

[3.1 Block diagram 5](#_Toc105594087)

[3.2 Version 1.0 RFSOC content 6](#_Toc105594088)

[3.2.1 RFSOC for version 1.0 Top-level block diagram 6](#_Toc105594089)

[3.2.2 Sampling 7](#_Toc105594090)

[3.2.3 Data-plan processing units 7](#_Toc105594091)

[3.2.4 Testing capabilities 8](#_Toc105594092)

[3.2.5 Configuration support 8](#_Toc105594093)

[3.3 Version 1.0 VERSAL content 9](#_Toc105594094)

[3.3.1 VERSION for version 1.0 Top-level block diagram 9](#_Toc105594095)

[3.3.2 Firmware Data processing 10](#_Toc105594096)

[3.3.3 Testing capabilities 10](#_Toc105594097)

[4 Version 2.0 11](#_Toc105594098)

[4.1 Block diagram 11](#_Toc105594099)

[5 Appendix a versal EVB-version 12](#_Toc105594100)

[5.1 Purpose 12](#_Toc105594101)

[5.2 Block diagram 12](#_Toc105594102)

[5.2.1 VERSAL-EVB operation concept 12](#_Toc105594103)

[5.2.2 DRAM memory structure and formats 13](#_Toc105594104)

[5.2.3 The Playback module 14](#_Toc105594105)

[5.2.4 AXI stream FIFO 17](#_Toc105594106)

[5.2.5 AXI stream interconnect 17](#_Toc105594107)

# Purpose of this documnet

The purpose of this document is to define the Firmware versions for CSPIR development.

The basic idea of the version is that each version supplied will be complete solution from samples to PDWS, which can be demonstrated.

|  |  |
| --- | --- |
| **Version** | **value** |
| 1.0 | Complete Data-plane solution, containing flow of   * Firmware processing: FFT->DET->BM-Packetizer * AI processing: Parameter estimator * Real Time SW processing on ARM - MERGE |
| 2.0 | Version 2.0 will add control plane to support  Simple scan operation, supporting OTD and simple Scan assigments |
| 3.0 | Will add abilities for Cognitive processing with addition of Narrow-Band machines and |

# Version 1.0

## Block diagram



Figure 1CSPIR version 1.0 Block Diagram

## Version 1.0 RFSOC content

### RFSOC for version 1.0 Top-level block diagram

Figure below shows the RFSOC top level block diagram

* ***Sampling***: ADC sample data at FS = FPGA\_SYS\_CLK x 16, FS targeted to 4.6Ghz internal FPGA\_CLK is 287.5[Mhz]
* ***Data-plane processing***: Data-plane processing flow
  + FFT: samples are transform to frequency domain
  + Detector: Detect Pulses and generate pulse envelope per FFT-BIN
  + Buffer-manager: save in Buffer Pulse FFT-BIN data
  + Arbiter: arbitrate between Buffers and send Data in packets to estimator (data is sent through Aurora toward the VERSAL FPGA)
* ***Testing capabilities***:
  + Enable recording input Data both to internal memory and to DRAM (limited BW)
  + Enable Playback of data from DRAM or from internal memory
  + Output data recorded to Internal Debug-memory
* ***Configurability***: all FPGA internal memory and CSR (Configuration Space Registers) can be access from ARM subsystem



### Sampling

* Sampling by ADC tiles with FS = 4.6GHz
* All 4 channels must be aligned at the single sampling level after power-up
* ADC tile should be calibrated to work at the 2nd nyquist zone

### Data-plan processing units

#### FFT

* FFT streaming Full-BW real-time processing
* FFT input is 14bit real,
* include Cheby windowing
* FFT parallel implementation with 16 slices
* Configurable window size of 128 to 4K

#### Detector and Threshold management

* Detector is based on detector IP used in SPIR projects
* Enable mechanism for Dynamic Threshold management
  + With presence of strong signal
  + When ADC is saturated
* 3 modes of Detector operation
  + Common detector interferometer mode: in this mode the detector between 4 channels is common and detect with AND logic the relevant mode
  + Common detector FOV mode: in this mode the detector between 4 channel is common and detect with OR logic (out of scope for this document)
  + Separate channel mode: each channel FFT is processed autonomously not related the other channels

#### Buffer manager

Based on Buffer manager module user in current SPIRs today 2 major operation mode

* Common-mode: input from 4 channels are save to Buffers together
* Independent-channel-mode: each channel is treats separately

#### Arbiter-packetizer

Arbitrate between Buffers, and send Data as AXI-stream packets to RFSOC through Aurora interface

#### Aurora-TX

4 lanes of Aurora TX interface @ 25Ghz

### Testing capabilities

#### Input Data Recording to DRAM

Enable recording of samples toward DRAM support following modes of recording

* Single channel 14bit
* 4 channels 8 MSB
* 4 channels 8 LSB

#### Output Data Recording

Support recording of BM packets Data output

* Data should be stored at internal FPGA memory with storage capacity of at least 1MB

#### Play back from DRAM

Enable playback data from DRAM in the following modes:

* Single-channel: Single channel data will be duplicate 4 times and playback toward FFT
* 4ChannelsMSB: Data from each channel will be zero padded at LSB and playback toward FFT
* 4ChannelsLSB: Data from each channels will be padded at MSB and playback toward the FFT

Enable streaming data from DRAM to processing (use DRAM samples instead of ADC inputs), at this mode all channel 1 Data will be inject as input toward the FFT of all 4 channels

### Configuration support

* Enable read/write operation to all RFSOC memory map including CSR (Configuration Space Registers)
* Enable read/write operation to/from DRAM

## Version 1.0 VERSAL content

### VERSION for version 1.0 Top-level block diagram

Figure below shows the VERSAL top level block diagram

* ***Data-plane processing***: Data-plane processing flow
  + ***FW-AI interface***: input is BM-data-packets arrive from RFSOC through Aurora interface, data is sent toward the AI engine
  + ***Estimator***: the parameter Estimator algorithm is implemented within the AI array and output PDW’s toward the Arm-Subsytem
  + ***Merge:*** The Merge algorithm is implemented within the ARM sub-system inputs are PDWs arrive from the estimator, output is Unite PDWS (concatenating several PDW’s to one PDW after LFM detection)
* ***Testing capabilities***
  + Play-Back of BM-data-packets from DRAM
  + Recording BM-data-packets arrive from RFSOC to DRAM
* ***Configurations support*** 
  + ARM Enable read/write operation to all VERSAL memory map including CSR (Configuration Space Registers)
  + ARM Enable read/write operation to/from DRAM
  + Enable access of AI ENGINES toward DRAM



### Firmware Data processing

#### Aurora TX

4 lanes @25Gbps of Aurora-RX interface

#### FW-AI interface

Deliver the packets arrive form RF-SOC through Aurora-TX toward the AI engine for estimator algorithm processing.

* Enable buffering of 1Mbtye to average rate toward the AI engine

#### Estimator (AI engine processing)

Implementation of parameter estimator module, input are BM packets, output is PDW (Pulse Descriptor word)

#### MERGE (Real-Time ARM processing)

MERGE algorithm inputs are PDWs arrive from the estimator, output is Unite PDWS (concatenating several PDW’s to one PDW after LFM detection)

### Testing capabilities

#### PlayBack

The playback module reads BM-packets stored at DRAM and stream them toward the FW-AI interface module as if they arrive from the Aurora-TX.

Each BM-packets stored at DRAM will have an additional info of TimeStamp, the Playback will refer to this Timestamp and in that manner will stream the BM-data-packets at the desired rate.

#### Input recording

write BM-data-packets toward DRAM, fro BM-data-packet recorded to DRAM an extra TimeStamp field will be added

# Version 2.0

## Block diagram

Block diagram of version 2.0 is shown below

The version add basic SCAN capabilities (control plane)

* VERSAL-RFSOC communication
  + VERSAL-RFSOC communication Master is added at the VERSAL
  + VERSAL-RFSOC communication slave is added at the VERSAL
* Basic Scan-support state machine (Task manager) implemented at VERSAL R5 processor



# Appendix a versal EVB-version

## Purpose

Purpose of this version are:

* Test and evaluate performance of estimator module implemented as stand alone
* The developed content will be design in a way that with minimum effort the design will be embedded within the CSPIR board

## Block diagram

Top-level block diagram of the VERSAL-EVB firmware version illustrated below



### VERSAL-EVB operation concept

The concept and flow of the Versal-EVB is as follow

* ARM load BM-packets from File into DRAM
* ARM configure the PlayBack module to start playing the BM-packtes
* PlayBack module reads data from DRAM and start streaming it toward the AI engine
* Axi-stream FIFO is used to average peak loads
* Axi stream interconnect send the BM-packet data toward the correct Axi-stream interface (there are 8 per PL-interface tile) based on Buffer-ID
* AI engine array process the BM-packets and send PDWs toward the ARM through the NOC tile interface

### DRAM memory structure and formats

The DRAM is used to store stimulus data (simulating BM-packets arrive from BM), before each test the ARM will write stimulus data from file toward the DRAM.

The basic unit of a stimulus data is BM-packet,

#### BM-packet structure

the BM-packet have 2 parts

* Header: fix 32 Bytes
* Payload: Variable length (up to 4K bytes)

The table below shows the BM-packet header fields which are relevant for the design

|  |  |  |
| --- | --- | --- |
| **Bits** | **Field** | **Description** |
| 0-63 | TimeStamp | The Timestamp when this packet arrive (this Timestamp is used by the Playback module to inject the stimulus at the same rate as arrive) |
| 64-67 | BufferID | Indicate the Buffer-ID from where the BM-packets is received, this field will be use to send the data to the correct AXI-stream interface toward the AI-engine |
| 68-79 | Payload size | Indicate the size of the Payload in bytes, maximum payload size is 4KB and it will be always an integer division of 4. |
| 80-255 | Other estimator configurations | Out of scope of this document |

#### BM-packet storage at DRAM

In order to make the Playback module robust and simple ARM will save each BM-packet in 8KB buffer,



### The Playback module

The purpose of the Play-Back module is to read stimulus Data from DRAM forward BM-packets toward the AXI-FIFO

Figure below shows the Payback module top-levele block diagram.



#### PlayBack interfaces

The Playback module include the following interfaces

* Interface to DRAM
* Configuration interface APB/AxiLite
* Axi-stream master interface is used to send the BM-packets toward the AXI stream FIFO
  + Axis\_Tlast is used to indicate the last data of packet
  + Axis\_tuser: will pass the BufferID field (which later will be used inorder to route the packet to the correct AXi-stream interface at the AI input)
  + Axi-Tready indication will be examined before start sending the packet

#### Configuration space register

|  |  |  |  |
| --- | --- | --- | --- |
| **RegAddr** | **RegName** | **DIR** | **Description** |
| 0x0 | DRAM\_Base\_ADDR | WO | Define the DRAM-Base |
| 0x4 | NumberOfBMPackets | WO | Define Number OF BM-Packets store at DRAM |
| 0x8 | Start | WO | Write to this register will start the PlayBack operation |
| 0xc | DropCnt | RO | Indicate number of BM-packets dropped because of AXI-stream FIFO was not ready |
| 0x10 | Status | RO | Status of Plaback State machine IDLE/BUSY |

#### PlayBack operation concept

The operation concept of the PlayBack module is illustrated at the figure below.

* Operation will start when ARM will write to start register (after start the Status register should be updated to BUSY)
* After Start the DRAM\_BaseAddr and the NumberOfBMPackets are stored loaded
* A complete BM-buffre-packet (maximum size is (4K+64) bytes), from header the Payload-size, TimeStamp, and Buffer-ID are extracted
* Wait for Time >=TimeStamp
* Examine Tready
  + If not ready the BM-packets will be dropped (register of Drop-cnt will be accumaletd)
  + If ready stream packet
* Stream-packet: The BM-packetis streaed with Axis-stream interface, the last data will be indicated by tlast and BufferId will be copied too the Tuser
* Decrease N to indicate whether this is the last packet,
  + If last packet: End (status register wil return to IDLE)
  + Otherwise: increase addr and read the next buffer



### AXI stream FIFO

The purpose of the AXi-stream FIFO is to buffer BM-packets which arrive at peak rate and by that enable to sustain an high average rate.

The FIFO is standard Xilinx AXIS-stream FIFO which supports

* Storage of 128Kbyte (1Mbut)
* Work and store and forward packet mode
* Tready behavior
  + Tready will be deasserted when there is no romm at the FIFO fro a complete BM-packer (4K+64) bytes
  + Tready will be deaaserted only after packet ends (TLAST)
* Support Tuser (pass theBufferID)

### AXI stream interconnect

Use standard iP core of Xilinx AXI-interconnect with

* One Axi-stream slave interface
* 8 AXI-stream master interface
* Tdata per AXi-stream master should not exceed 64 bits