DVB S2 Modulator IP Core in VHDL

IP Core Serial Number

Angelia DVB S2 Modulator No 001 v1.0

IP Types

Soft IP

Standards

DVB S2 System from ETSI EN 302 307-1 V1.4.1 (2014-11)
Digital Video Broadcasting (DVB);
Second generation framing structure, channel coding and modulation systems for Broadcasting, Interactive Services, News Gathering and other broadband satellite applications;
Part 1: DVB-S2

Languages

VHDL

Development Environment

The VHDL software was developed using the following development environment for VHDL synthesis and VHDL simulation.
Xilinx Vivado 2015.2

Maturity / Status

Pre-Silicon

Overview

The aim of this IP core is to implement RTL components for DVB-S2 Modulator on a SoC (System on Chip).

The IP core will match exactly what GNU Radio produces for every combination of parameters described on the DVB-S2 base spec (no extensions yet). This means components will handle

- Frame types: Normal and short
- Constellations: 8 PSK, 16 APSK and 32 APSK
- Code rates: 1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10,

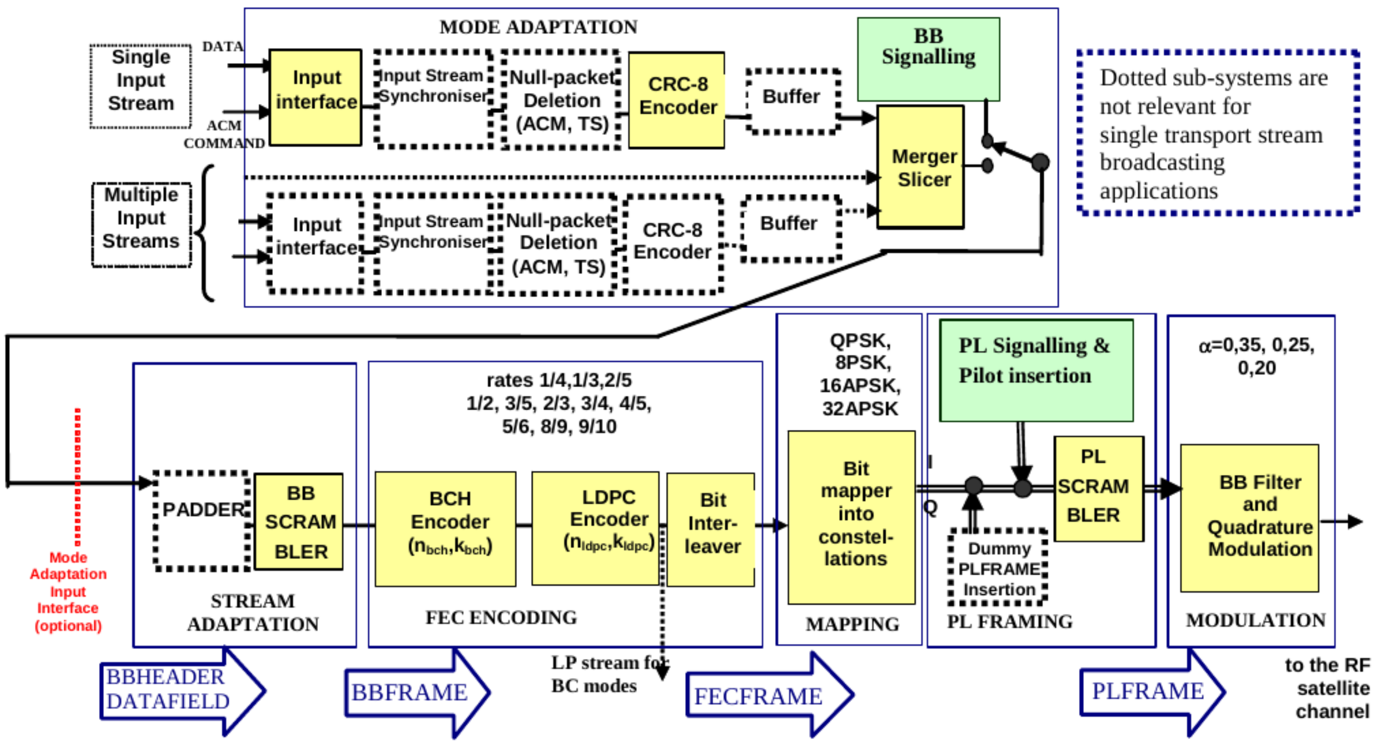
Components will also handle parameters changing on every frame, that is, they will handle frame with config A then a frame with config B immediately afterwards without requiring reset or wait cycles

Use AXI-Stream interfaces

Deliverables

VHDL source code
HDL simulation models
Comprehensive documentation

Block Diagram


MODE ADAPTATION
Single Input Stream
DATA
ACM COMMAND
Input interface
Input Stream Synchroniser
Null-packet Deletion (ACM, TS)
CRC-8 Encoder
Buffer
BB Signalling
Merger Slicer
Multiple Input Streams
Dotted sub-systems are not relevant for single transport stream broadcasting applications
Mode Adaptation Input Interface (optional)
PADDER
BB SCRAMBLER
STREAM ADAPTATION
rates 1/4,1/3,2/5 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9, 9/10
BCH Encoder (n_{bch},k_{bch})
LDPC Encoder (n_{ldpc},k_{ldpc})
Bit Inter-leaver
FEC ENCODING
QPSK, 8PSK, 16APSK, 32APSK
Bit mapper into constel-lations
MAPPING
I
Q
PL Signalling & Pilot insertion
Dummy PLFRAME Insertion
PL SCRAMBLER
PL FRAMING
α=0,35, 0,25, 0,20
BB Filter and Quadrature Modulation
MODULATION
BBHEADER DATAFIELD
BBFRAME
LP stream for BC modes
FECFRAME
PLFRAME
to the RF satellite channel

Functional block diagram of the DVB-S2 System from ETSI EN 302 307-1 V1.4.1

System Components

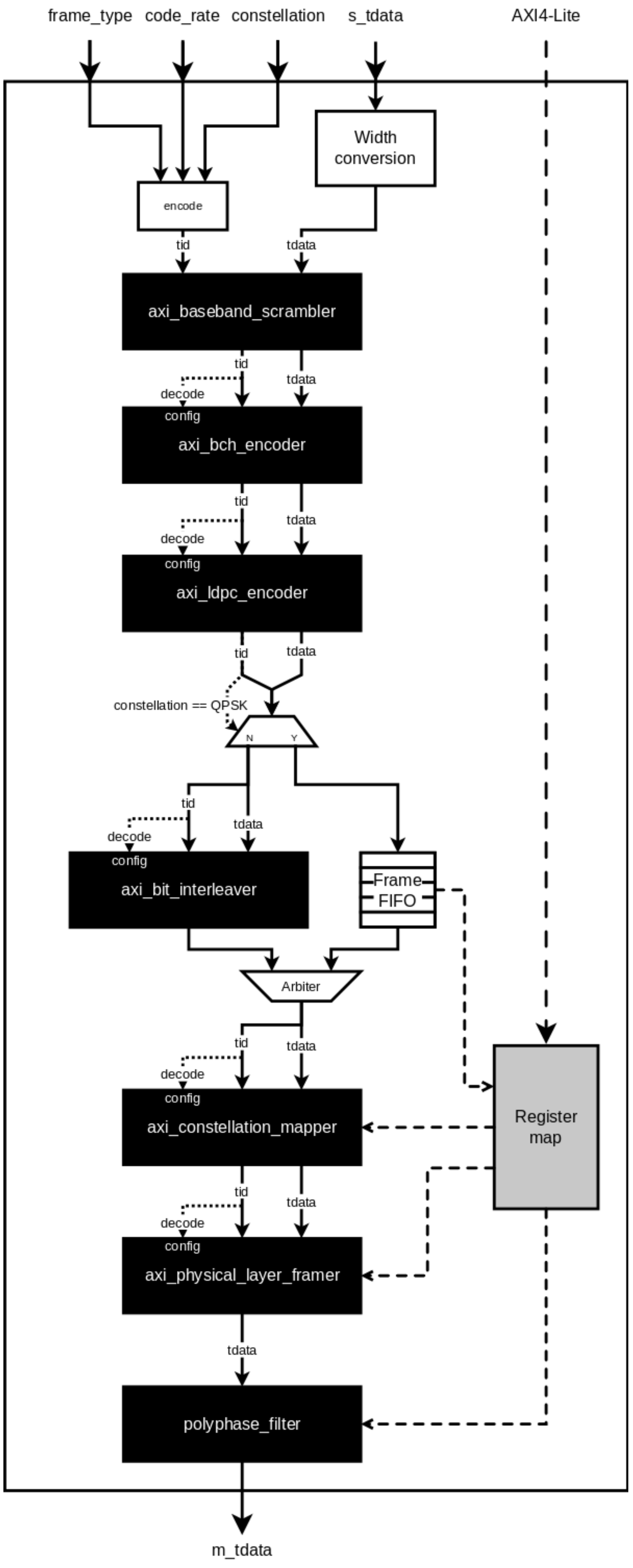
Core DVB-S2 components

Baseband scrambler

BCH encoder
Bit interleaver
Constellation mapper
LDPC Encoder
Physical layer framing

DVB S2 TX Diagram

dvbs2_tx block diagram

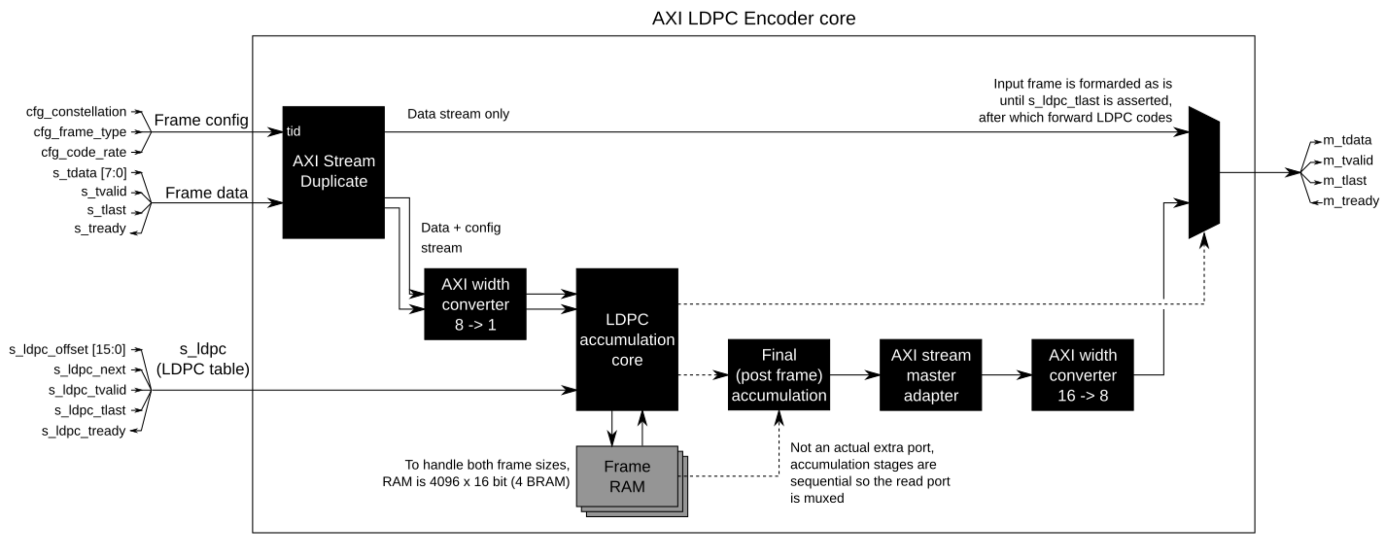


DVB-S2 Tx block diagram

This is the top module for the DVB-S2 transmitter, which instantiates the subcomponents needed by the standard. Note that a frame's configuration parameters accompany the data using AXI's TID/TUSER, meaning each input frame can have different parameters.

AXI LDPC Encoder Core

axi_ldpc_encoder_core_block_diagram


AXI LDPC Encoder core
cfg_constellation
cfg_frame_type
cfg_code_rate
Frame config
s_tdata [7:0]
s_tvalid
s_tlast
s_tready
Frame data
tid
AXI Stream Duplicate
Data stream only
Data + config stream
AXI width converter 8 -> 1
LDPC accumulation core
s_ldpc_offset [15:0]
s_ldpc_next
s_ldpc_tvalid
s_ldpc_tlast
s_ldpc_tready
s_ldpc (LDPC table)
Final (post frame) accumulation
AXI stream master adapter
AXI width converter 16 -> 8
Input frame is formarded as is until s_ldpc_tlast is asserted, after which forward LDPC codes
m_tdata
m_tvalid
m_tlast
m_tready
To handle both frame sizes, RAM is 4096 x 16 bit (4 BRAM)
Frame RAM
Not an actual extra port, accumulation stages are sequential so the read port is muxed

AXI LDPC Encoder block diagram

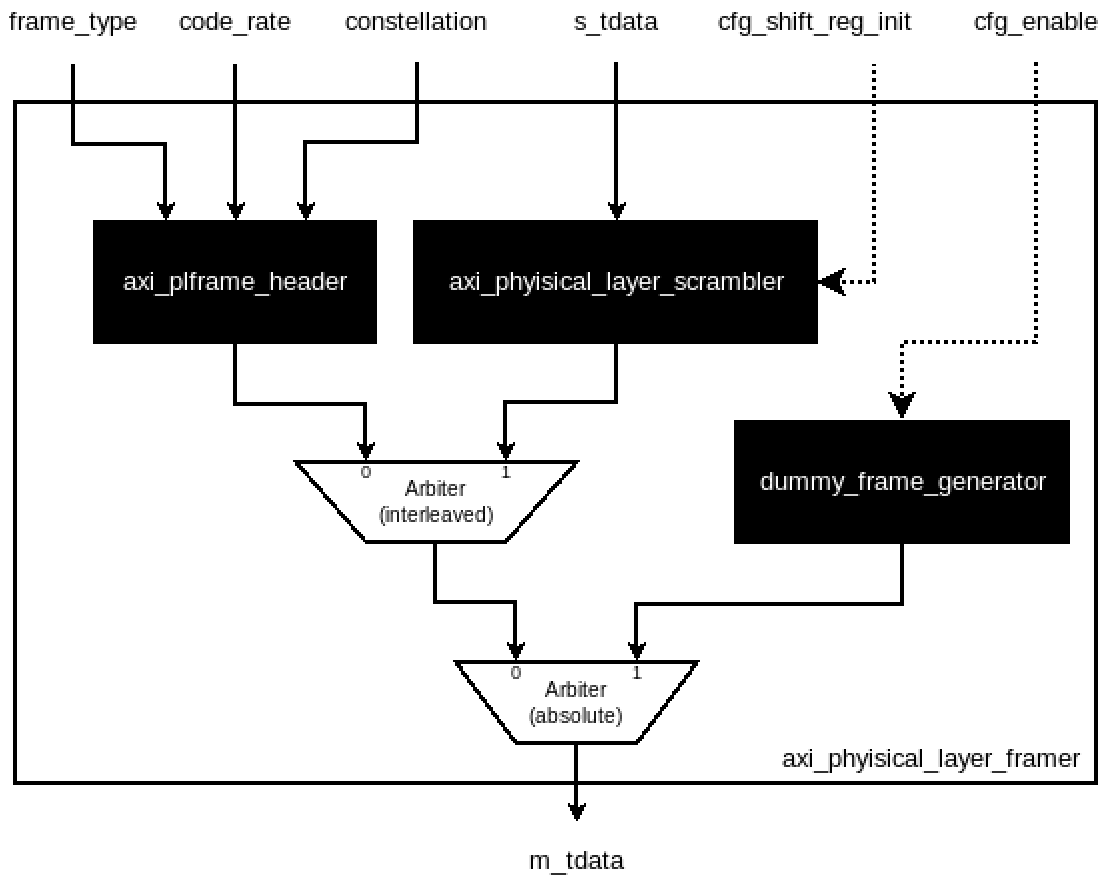
Data frame is fed to the frame data (s_tdata) and DVB parameters should be synchronized with it.

Values from the DVB tables (Annexes B and C from ETSI EN 302 307-1 V1.4.1), are expanded to per bit and fed to the s_ldpc interface. This interface need 2 values: the actual frame offsets (s_ldpc_offset) and when to process another frame bit (s_ldpc_next).

When s_ldpc_tlast is received, accumulation is considered completed and the final accumulation begins, in which point the component will not accept any data until the output frame is fully written.

AXI Physical Layer Framer

axi_physical_layer_framer



AXI Physical Layer Framer block diagram

Frame parameters are fed to the AXI PLFRAME Header block while data goes through the physical layer scrambler. The first arbiter is set up in interleaved mode to alternate between a header frame and a data frame. The resulting data stream is connected to a second arbiter whose other input is connected to the dummy frame generator. This arbiter is set up in absolute mode so that actual data frames take precedence over dummy frames.

Performance and Resource Usage

The dvbs2_modulator.vhd top level has been run through Vivado targeting a xczu4cg-sfvc784-1LV-i and with a clock frequency of 300 MHz (both arbitrary). No timing issues were reported and the resource usage post implementation is show below. Table below assumes default values for generics, i.e., POLYPHASE_FILTER_NUMBER_TAPS = 33, POLYPHASE_FILTER_RATE_CHANGE = 2 and DATA_WIDTH = 32.

Component	LUTs	FFs	RAMB	DSPs
axi_baseband_scrambler	277	46	0	0
axi_bch_encoder	1397	1380	0	0
axi_ldpc_encoder	1017	558	6	0
axi_bit_interleaver	339	262	10	0

Component	**LUTs**	**FFs**	**RAMB**	**DSPs**
axi_constellation_mapper	599	253	0	0
axi_physical_layer_framer	280	268	0	0
+ axi_plframe_header	13	100	0	0
+ axi_physical_layer_scrambler	39	70	0	0
+ dummy_frame_generator	48	21	0	0
polyphase_filter_i/q	162	2168	0	64
Register map	586	318	0	0
Debug infrastructure	374	581	0	0
Others	1470	283	4	0
TOTAL	**~6.5k**	**~6.1k**	**20**	**64**

Source File Description

Mode Adaption

CRC8
CRC8 (cyclic redundancy check)

InputInterface
Input Interface

input_Buffer
Input Buffer

BBSignaling
BBSignaling Block generates the BBHeader

merger_slicer_1
The Merger Slicer stores the UP-packets in a Datafield.

Merger_Slicer_BBSignaling
In this Block the Merger-Slicer- and the BBSignaling-Block will be instantiated

Stream Adaption

axi_baseband_scrambler
Baseband scrambling

FEC Encoding

axi_bch_encoder
BCH encoder

axi_ldpc_encoder
LDPC encoder

axi_bit_interleaver
Bit interleaver

Mapping

axi_constellation_mapper
Constellation mapper

PL Framing

axi_physical_layer_framer
Physical layer framer

Modulation

polyphase_filter
BB Filter

Quadrature_modulation
Quadrature Modulation

Synthesis

Running synthesis

Scripts are provided as an example to get things going, currently this has not been tested in real hardware.

Yosys

```
./misc/run_synth.sh
```

Vivado

```
vivado -source ./build/vivado/build.tcl
```

Testing

Each module tested individually for all relevant DVB-S2 base spec parameters (+ AXI streaming specifics), for a total of more than 1100 tests, designed to handle parameters varying on a per frame basis

Tests can be run locally or on a Docker container. Running locally will require GNU Radio, VUnit and a VHDL simulator.

Using Docker

Uses the same container used for CI

```
# Clone this repo and submodules
git clone --recurse-submodules    https://github.com/phase4ground/dvb_fpga
cd dvb_fpga
# Run the tests
./misc/run_tests.sh
```

Arguments passed to docker/run_tests.sh will be passed to run.py and, by extension, to VUnit (no environment variable is passed on though). This will generate gnuradio_data (test data).

Running locally

Requirements

- GNU Radio
- A VHDL simulator
- VUnit

```
# Install VUnit
pip install vunit-hdl
# Clone this repo and submodules
git clone --recurse-submodules    https://github.com/phase4ground/dvb_fpga
cd dvb_fpga
# Run the tests
./run.py
```

The first invocation of run.py will run GNURadio and create stimulus files.

To list tests use ./run.py -l:

```
$ ./run.py -l
lib.axi_bit_interleaver_tb.data_width=8,all_parameters.back_to_back
lib.axi_bit_interleaver_tb.data_width=8,all_parameters.slow_master
lib.axi_bit_interleaver_tb.data_width=8,all_parameters.slow_slave
lib.axi_bit_interleaver_tb.data_width=8,all_parameters.both_slow
lib.axi_ldpc_table_tb.test_all_configs.back_to_back
lib.axi_ldpc_table_tb.test_all_configs.slow_master
lib.axi_ldpc_table_tb.test_all_configs.slow_slave
lib.axi_ldpc_table_tb.test_all_configs.slow_master,slow_slave
lib.axi_bch_encoder_tb.test_all_configs.back_to_back
lib.axi_bch_encoder_tb.test_all_configs.slow_master
lib.axi_bch_encoder_tb.test_all_configs.slow_slave
lib.axi_bch_encoder_tb.test_all_configs.both_slow
lib.dvbs2_encoder_tb.test_all_configs.back_to_back
lib.axi_ldpc_encoder_core_tb.test_all_configs.back_to_back
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=0.5,table=1.0,slave=1.0
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=1.0,table=1.0,slave=0.5
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=0.75,table=1.0,slave=0.75
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=1.0,table=0.5,slave=1.0
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=1.0,table=0.75,slave=0.75
lib.axi_ldpc_encoder_core_tb.test_all_configs.data=0.8,table=0.8,slave=0.8
lib.axi_baseband_scrambler_tb.test_all_configs.back_to_back
lib.axi_baseband_scrambler_tb.test_all_configs.slow_master
lib.axi_baseband_scrambler_tb.test_all_configs.slow_slave
lib.axi_baseband_scrambler_tb.test_all_configs.both_slow
Listed 24 tests
```

Integration Guide

Continuous integration using open source tools

- GHDL for VHDL simulation
- VUnit for unit testing support
- Yosys for RTL synthesis
- Ensures code is both functionally correct and synthesis friendly

See INTEGRATION GUIDE.pdf

Developer

Mark Chen
Angelia Technology / ICTech
http://www.angelia.eu.org

License

See “License Agreement”.