

IP Core for Medium Data Rate PDT

Datasheet

Revision 8.0
Date 12/07/2018

Contents

1	Introduction	5
1.1	Scope	5
1.2	Applicable, Reference and Annex Documents	5
1.2.1	Applicable documents.....	5
1.3	Definitions and Acronyms	6
1.3.1	Definitions.....	6
1.3.2	Acronyms.....	6
2	Features	7
3	Applications	8
4	Description.....	9
4.1	Processing chain	9
4.2	Configuration and status	10
4.3	External memory interface	10
4.4	Single Event Effects mitigation.....	11
5	Port description.....	12
5.1	Input data interface	13
5.2	Output data interface	13
5.3	Configuration and status interface.....	14
5.4	LUT loader interface	16
5.5	Clock, reset and enable	17
6	Hard configuration options	18
7	Usage	20
8	Technology Mapping.....	23
9	BER performance	25

List of Figures

Figure 2-1 IP Core High-level functional block diagram.....	8
Figure 5-1 CCSDS Telemetry Transmitter IP Core top-level interface.....	12
Figure 5-2 Input data interface timing.....	13
Figure 5-3 Output data interface timing.....	14
Figure 5-4 Configuration and status interface timing.....	16
Figure 5-5 LUT loader interface timing.....	17
Figure 7-1 IP core generator Graphical User Interface.....	21
Figure 9-1 IP core BER performance for ACM1 and ACM7.....	25
Figure 9-2 IP core BER performance for ACM13 and ACM18.....	26
Figure 9-3 IP core BER performance for ACM23.....	26

List of Tables

Table 4-1 LUT loader memory map	10
Table 4-2 SEE mitigation techniques summary.....	11
Table 5-1 Input data interface	13
Table 5-2 Output data interface.....	13
Table 5-3 Configuration and status interface.....	15
Table 5-4 LUT loader interface.....	16
Table 5-5 Clock, reset and enable interface	17
Table 6-1 IP core hard configuration parameters	19
Table 8-1 ModCod support on the different technologies	23
Table 8-2 Multiple-ModCod instantiation synthesis results on the different technologies.....	24
Table 8-3 IP core post place and route performance.....	25

1 Introduction

1.1 Scope

This document represents the Datasheet for the activity “IP Core for Medium Data Rate PDT” initiated by the European Space Agency under ESTEC contract 400117901/16/NL/FE/eg.

The work has been performed by IngeniArs S.r.l., Pisa, Italy.

1.2 Applicable, Reference and Annex Documents

1.2.1 Applicable documents

AD	Doc. No.	Issue/ Rev.	Title
[AD01]	CCSDS 131.2-B-1	1	Flexible advanced coding and modulation scheme for high rate telemetry applications
[AD02]	TEC-ETC/2015.172	2	Statement of Work - IP Core for Medium Data Rate PDT
[AD03]	ING-IPCTM-SMP	2	IP Core for Medium Data Rate PDT – SEE Mitigation Plan

1.3 Definitions and Acronyms

1.3.1 Definitions

Recurring definitions are reported in the following table.

Definition	Description
Hard configuration	Configuration item selected during IP core generation or through code parameter, which cannot change at run-time. Hard configuration parameters are typically used to optimize the specific implementation of the IP core, while maintaining a more general-purpose IP core.
ModCod	Modulation and Coding scheme
Soft configuration	Configuration item which can be modified at run-time, e.g., through configuration registers.

1.3.2 Acronyms

Acronyms used in the present document are listed in the following table.

Acronym	Description
ACM	Adaptive Coding and Modulation
BER	Bit Error Rate
CADU	Channel Access Data Unit
CCSDS	Consultative Committee for Space Data Systems
ECSS	European Cooperation for Space Standardization
EDAC	Error Detection and Correction
FPGA	Field-Programmable Gate Array
GUI	Graphical User Interface
HPA	High-Power Amplifier
IPR	Intellectual Property Rights
LUT	Look-Up Table
PL	Physical Layer
RAM	Random Access Memory
SCCC	Serially Concatenated Convolutional Coding
SRRC	Square Root Raised Cosine
SEE	Single Event Effect
VHDL	VHSIC Hardware Description Language

2 Features

- Fully compliant with CCSDS 131.2-B standard;
- Support of all the 27 ModCods in a single instantiation for high capacity FPGA technologies for space (e.g. Microsemi RTG4 and Xilinx Virtex 5QV);
- Less than 0.2 dB implementation loss
- High data-rate IP Core options for symbol rates beyond 200 Mbauds on space FPGA technologies, and beyond 500 Mbauds on commercial FPGA technologies
- Low-complexity IP Core options for implementation on antifuse technology (e.g. RTAX2000 FPGA);
- IP core generator Graphical User Interface to configure and generate the IP core according to the needs of the end-user and in particular:
 - ModCods supported
 - Desired Symbol rate
 - Full performance vs Low Complexity Architecture
 - LUT implementation mode, internal vs external, to maximize the number of ModCods supported in a single instantiation
 - SRRC shaping filter enabling and roll-off selection
 - Target technology
 - Pilot insertion enabling and pseudo-randomiser parameter selection
- The IP core is coded in technology-independent, highly configurable VHDL;
- Mapping on the following reference space FPGA technologies:
 - Microsemi RTAX2000
 - Microsemi RTG4
 - Microsemi RT-ProASIC 3000L
 - Xilinx Virtex-5 XQRFVFX130
 - NanoXplore BRAVE NG-MEDIUM
- IP Core validation on the new Microsemi RTG4 the Xilinx Zynq-7000 SoC development kits

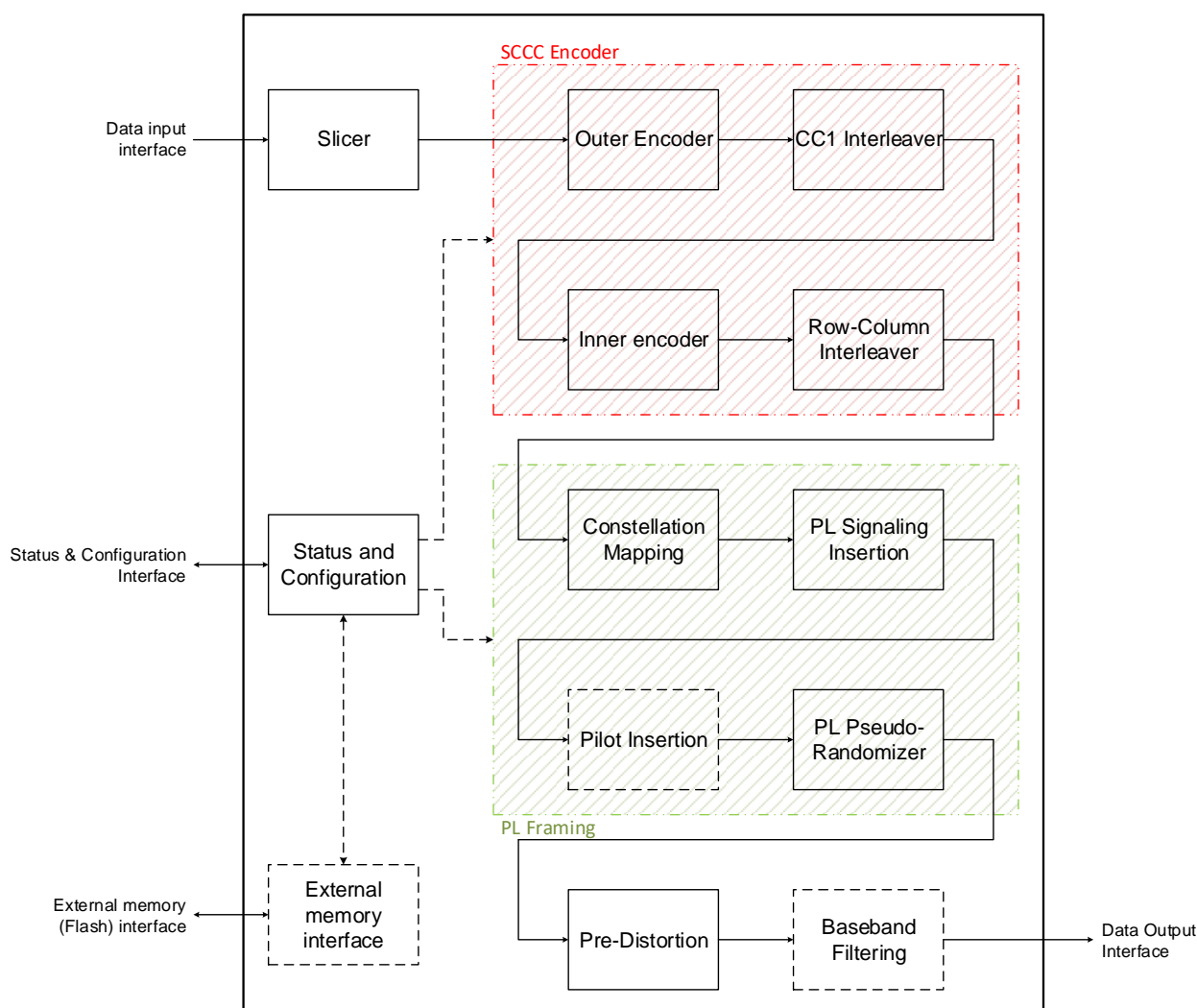


Figure 2-1 IP Core High-level functional block diagram

3 Applications

A number of Earth Observation missions will be based on small satellites class and will embark payloads producing substantial data rates, thus requiring a reliable, efficient and economical payload data transmitter specialised for mid-range data rate, i.e. few hundred Mb/s.

Such missions would benefit from employing state-of-the-art coding and modulation standard, allowing to exploit the protection offered by modern coding techniques while at the same time maximizing the supported data rates by using spectral efficient modulation formats. The CCSDS 131.2-B standard, combining powerful Serially Concatenated Convolutional Codes (SCCC) with modulations ranging from QPSK to 8PSK and 16-, 32- and 64-APSK, offers precisely such benefits, together with a high degree of flexibility. Such flexibility, due to the number of modulation and coding formats (ModCod) provided, will help configuring the system to better adapt to the specific target requirements.

Additionally, this will provide the means to adapt the payload data transmission (PDT) sub-system to variable channel conditions, due to either geometry (variable path slant travelled by the signal from a satellite in a LEO orbit to the ground station) or variable atmospheric conditions (which are particularly significant in the case of transmission using frequency bands such as 25.5-27 GHz). By changing ModCod based on a pre-defined, preprogrammed scheme (Variable Code and Modulation, VCM) or adapting dynamically the ModCod based on the actual channel conditions (Adaptive Code and Modulation, ACM), such system will allow maximizing the overall data return, allowing a dramatic performance increase over traditional system based on a single ModCod only.

4 Description

The IP Core implements a Serially Concatenated Convolutional Coding (SCCC) scheme for telemetry application, which functionality is compliant with CCSDS 131.2-B-1 standard. The transmitter makes use of a large variety of modulation schemes (including QPSK, 8-PSK, 16-APSK, 32-APSK and 64-APSK) and a wide range of coding rates.

In the following sections, the functionality of the IP core is briefly described.

4.1 Processing chain

The high-level functional block diagram of the IP Core is presented in Figure 2-1. The functionality of each block in the processing chain is described in the following. More details about algorithms implemented for each functional block can be found in the CCSDS 131.2-B standard.

- **Slicer:** this block has to split the input CADU stream into a sequence of information blocks of length K , corresponding to the information block size of the selected ModCod.
- **Outer encoder (CC1):** performs the first convolutional encoding and the output bit-stream is punctured by decimating the parity bits by half for an overall coding rate of 2/3.
- **CC1 interleaver:** this block implements the ad-hoc interleaving law described in the annex B of the CCSDS 131.2-B standard, which makes extensive use of Look-Up Tables.
- **Inner Encoder (CC2):** carries out the second convolutional encoding, with the same algorithm as the outer encoder. Puncturing adjusts the data size of the block in order to fit the 8100 output symbols. Differently from outer puncturing, the puncturing after inner encoder is much more complex as it is composed by different algorithms for puncturing systematic bits and parity bits. For systematic bits, an ad-hoc law based on Look-Up Tables is used. For parity bits, a rate-matching algorithm is used in order to fit the output symbol rate.
- **Row-Column Interleaver:** is used to interleave the bits that are assigned to symbols. The input to the Row-Column interleaver are the systematic bits followed by the parity bits from the punctured output of the inner encoder.
- **Constellation Mapping:** is used to map the SCCC encoded bit-stream to symbols, according to the modulation index defined in each ModCod.
- **PL Signalling Insertion:** is used to frame the sequence of codewords sections, each composed of 8100 symbols. A frame header segment is added every 16 codeword sections (codeword segment). The frame header together with the codeword segment constitutes the Physical Layer Frame.
- **Pilot Insertion:** is used to include a pilot sequence to facilitate carrier and phase synchronization. The pilot insertion is an optional functional block and it is possible to bypass it at IP Core instantiation.

- **PL Pseudo-Randomiser:** is used to have a sufficient number of bit transitions in order to allow proper synchronisation of the decoder.
- **Pre-Distortion:** performs a complex (I/Q) multiplication between the output of the constellation mapper and static pre-distortion parameters in order to compensate for amplitude/phase non-linearity at the HPA.
- **Baseband filtering:** it performs Square Root Raised Cosine (SRRC) pulse-shaping on I and Q arms, with the roll-off options given in the CCSDS 131.2-B standard. This functionality can be excluded by the user, through generic, in this case the output of the IP Core will coincide with the output of the Pre-Distortion block.

4.2 Configuration and status

The transmitter soft configuration can be set through the Configuration & Status Interface. The configuration and status block takes care of finishing the current processing before applying the new configuration. Depending on the hard configuration setting, the configuration change may require loading the algorithm LUTs from an external memory.

This block provides the status of the IP core, i.e., whether the IP core is ready for processing, the current ACM, inconsistency error flags (if SEE mitigation techniques are enabled), correctness of desired configuration (invalid ACMs).

4.3 External memory interface

As already mentioned, the CCSDS 131.2-B processing relies on the extensive use of look-up tables. In particular, nearly every ModCod makes use of specific LUTs, therefore a huge amount of logic is used, especially when multiple ModCods are supported within the same IP core instantiation. In order to maximize the number of ModCods supported, in particular on relatively small devices such as the Microsemi RTAX2000, the LUTs used in the processing chain can be stored in an external memory and only the values used for the selected ACM are loaded into internal RAM. In this case a technology independent interface to load internal LUTs is needed and will be included at IP Core instantiation, the LUT loader interface. When a soft configuration is triggered and LUTs are not stored internally, the IP core requires new LUT values through the LUT loader interface. The memory map for the LUT loading process is represented in Table 4-1.

Start Address	End Address	Address space	Description
0x000	0x3FF	1024	alfa/beta LUT
0x400	0x43F	64	Constellation mapper
0x480	0x57F	256	Row-Column Address Map LUT
0x600	0x 85F7	32760	S puncturing LUT

Table 4-1 LUT loader memory map

4.4 Single Event Effects mitigation

Several mitigation techniques are employed in the IP Core to reduce the effects of radiations. The inclusion of such countermeasures is configurable at IP core instantiation. SEE mitigation techniques include:

- Registers protection
- Safe FSM implementation
- RAM protection

Table 4-2 summarizes the mitigation techniques that are applied to the IP Core design.

Mitigation Technique	Target	Penalty
Local TMR	All FFs in the design	Triplicated number of FFs, added voting logic, timing degradation
Deadlock-free FSM	All FSM logic	Redundant logic
FSM "one hot" coding	All FFs that store FSM states	Increased number of FFs used to represent the FSM state (from $\lceil \log_2(N) \rceil$ to N)
EDAC	Algorithm LUTs, only if read externally.	Increased RAM size. Timing degradation due to additional EDAC combinatorial logic
Scrubbing	Algorithm LUTs, only if read externally.	Additional logic to implement scrubbing

Table 4-2 SEE mitigation techniques summary

5 Port description

The CCSDS Telemetry Transmitter IP core interface consists of:

- Input data interface
- Output data interface
- Configuration and status interface
- LUT loader interface
- Clock, reset and enable

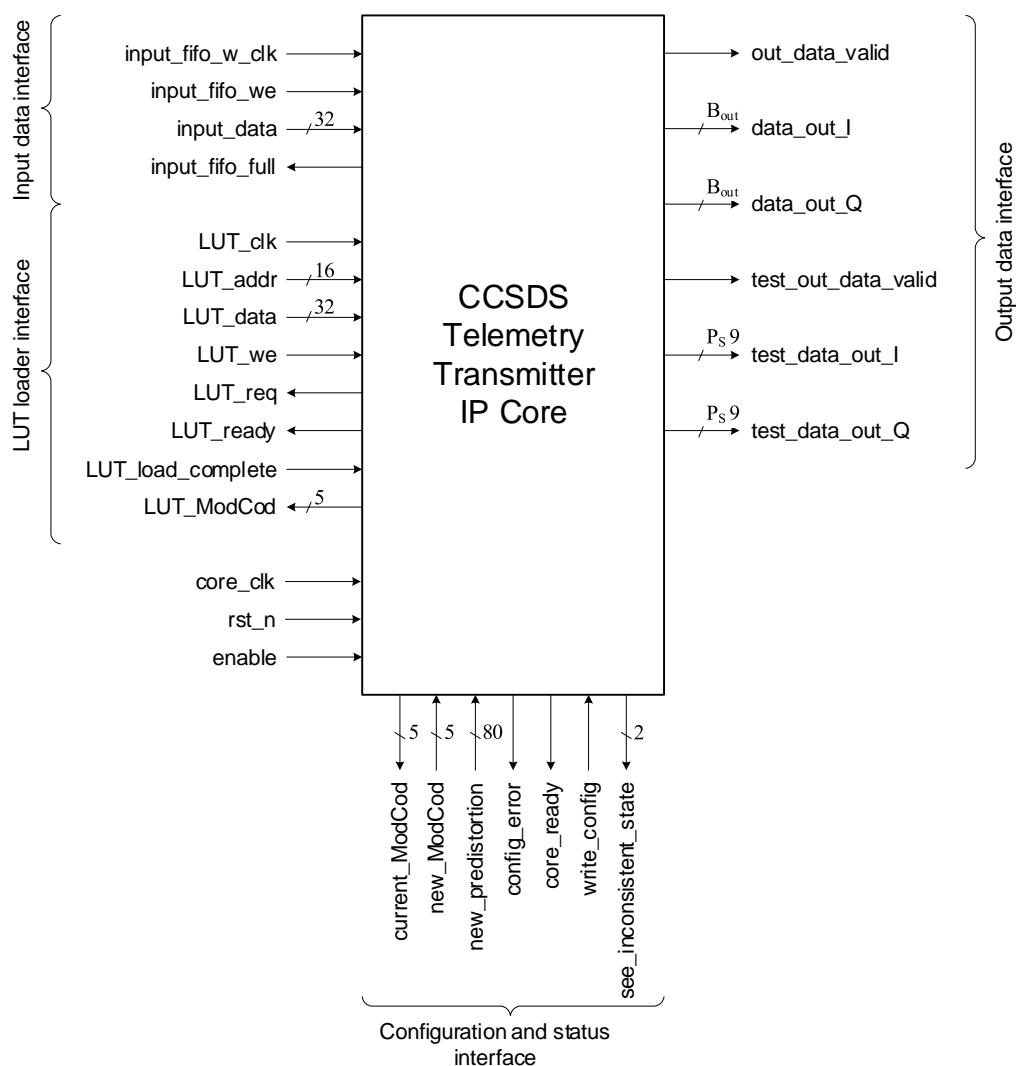


Figure 5-1 CCSDS Telemetry Transmitter IP Core top-level interface

In the following sections, each interface is detailed with relevant timings.

5.1 Input data interface

Port name	I/O	bits	Clock domain	Description
input_fifo_w_clk	I	1	input_fifo_w_clk	Write clock for the input FIFO buffer.
input_fifo_we	I	1	input_fifo_w_clk	Write enable for the input FIFO buffer. When this signal is asserted on the rising-edge of input_fifo_w_clk, input_data is written to the input FIFO buffer.
input_data	I	32	input_fifo_w_clk	Data bits to be written to the input FIFO buffer.
input_fifo_full	O	1	input_fifo_w_clk	Indicates that the input FIFO buffer is full and every write operation is ignored.

Table 5-1 Input data interface

The input data interface provides the means to feed data to the IP core for processing. The IP core has an asynchronous input buffer to overcome problems due to clock domain crossing when integrating the IP core in the system. The input data interface is made of a 32-bit data bus `input_data`, with simple write_enable/full control signals (`input_fifo_we`, `input_fifo_full`). The interface is synchronous to `input_fifo_w_clk`. When `input_fifo_full` is asserted, write operations are ignored, to avoid overwriting FIFO contents.

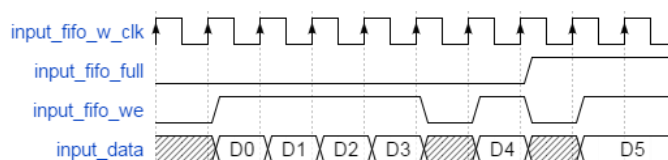


Figure 5-2 Input data interface timing

5.2 Output data interface

Port name	I/O	bits	Clock domain	Description
out_data_valid	O	1	core_clk	Indicates that data_out_I and data_out_Q outputs are valid.
data_out_I	O	Bout	core_clk	Output signal on I-arm. Bout depends on: - Ps the output symbol parallelism - whether the SRRC filter is enabled - the oversampling of the SRRC filter (SRRC_OVERSAMPLING = 4) - The number of bits used to represent the symbols/the filtered output When the SRRC filter is disabled: $Bout = P_s * B_OUT_UNFILTERED$, where $B_OUT_UNFILTERED = 8$ When the SRRC filter is enabled $Bout = P_s * SRRC_OVERSAMPLING * B_OUT_FILTERED$, where $B_OUT_FILTERED = 12$
data_out_Q	O	Bout	core_clk	Output signal on Q-arm. Bout is the same as data_out_I.
test_out_data_valid	O	1	core_clk	Indicates that test_data_out_I and test_data_out_Q outputs are valid. This port is used for test only and should be left unconnected in normal operating conditions.
test_data_out_I	O	$9 * P_s$	core_clk	Unfiltered I-arm output. This port is used for test only and should be left unconnected in normal operating conditions.
test_data_out_Q	O	$9 * P_s$	core_clk	Unfiltered Q-arm output. This port is used for test only and should be left unconnected in normal operating conditions.

Table 5-2 Output data interface

The modulated and encoded signal is produced through the output data interface. The in-phase component and quadrature component are presented on `data_out_I` and `data_out_Q`, respectively. In order to maximize the IP core throughput while keeping core clock frequency low, output samples are produced in parallel on B_{out} bits. The value of B_{out} depends on whether the SRRC filter is enabled or not. Each sample is represented on a different number of bits B_s depending on the presence of the SRRC filter: $B_s = 8$ bits in case the filter is not present and $B_s = 12$ bits in case the filter is present. In addition, when the filter is present, it interpolates the symbols by a factor of 4 (SRRC_OVERSAMPLING, ρ). Therefore, when the filter is not present, $B_{out} = P_s B_s$ (P_s is the number of symbols to be presented in parallel) and when the filter is present, $B_{out} = \rho P_s B_s$. `data_out_I[Ps* ρ *Bs-1:(Ps* ρ -1)*Bs]` and `data_out_Q[Ps* ρ *Bs-1:(Ps* ρ -1)*Bs]` represent the complex sample to be transmitted first, then `data_out_I[(Ps* ρ -1)*Bs-1:(Ps* ρ -2)*Bs]` and `data_out_Q[(Ps* ρ -1)*Bs-1:(Ps* ρ -2)*Bs]` is transmitted, and so on, the last complex sample is `data_out_I[Bs-1:0]` and `data_out_Q[Bs-1:0]`. `out_data_valid` indicates that the output complex sample is valid. Each sample is represented in 2's complement binary format.

The output data interface is synchronous to the core clock, `core_clk`.

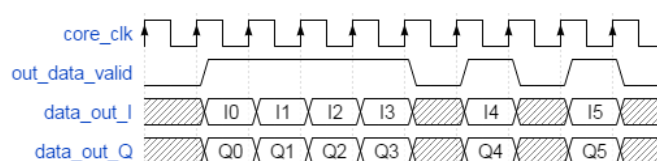


Figure 5-3 Output data interface timing

Figure 5-3 shows the timing of the output data interface. Please note that the timing shown for `out_data_valid` is illustrative. The actual behavior depends on the availability of data in the input FIFO. If the input FIFO always has data, then the transmission will be contiguous.

5.3 Configuration and status interface

Port name	I/O	bits	Clock domain	Description
<code>core_ready</code>	O	1	<code>core_clk</code>	When asserted, it indicates that the IP core is operating and is ready to receive the configuration.
<code>config_error</code>	O	1	<code>core_clk</code>	Asserted when the configuration written is not valid. In this case the IP core runs with the previous valid configuration. De-asserted when the configuration process is initialised.
<code>write_config</code>	I	1	<code>core_clk</code>	With a one clock-cycle pulse, the IP core enters the configuration process, if <code>core_ready</code> is asserted.
<code>new_ModCod</code>	I	5	<code>core_clk</code>	ModCod to be set in the next configuration process. This input must be stable on the rising edge of <code>core_clk</code> , when <code>write config</code> is asserted.
<code>new_predistortion</code>	I	80	<code>core_clk</code>	Predistortion value to be set in the next configuration process. This input must be stable on the rising edge of <code>core_clk</code> , when <code>write config</code> is asserted. This input is neglected when static predistortion is set as hard configuration parameter
<code>current_ModCod</code>	O	5	<code>core_clk</code>	Currently set ModCod.

Port name	I/O	bits	Clock domain	Description
see_inconsistent_state	O	2	core_clk	Asserted when an internal inconsistency due to SEE is detected. This could be due to EDAC on memories or invalid state in FSM.

Table 5-3 Configuration and status interface

Soft-configuration and information on the IP core status is provided through the configuration and status interface. The operational status is indicated by `core_ready` and `current_ModCod`. When `core_ready` is asserted, it indicates that the IP core is operating properly with the selected `current_ModCod`. When `core_ready` is asserted, it also indicates that the IP core is ready to receive a new configuration. The new configuration can be provided by the host system by producing a one clock cycle pulse of `write_config` when the desired `new_ModCod` and `new_predistortion` (when predistortion is on) signals are stable. The `write_config` pulse initiates the configuration process: `core_ready` is de-asserted, the IP core finishes the transmission of the buffered data, then the configuration is applied. External loading of the look-up tables may be required depending on the IP core hard configuration settings. The configuration process terminates when `core_ready` is asserted. If `config_error` is not asserted, then the configuration process succeeded. If `config_error` is also asserted, then an invalid ModCod was selected for configuration and the configuration process did not succeed. In this case, the previous ModCod is kept as current ModCod and the `config_error` flag remains asserted until the next configuration.

When SEE mitigation is enabled, internal inconsistencies due to SEE is reported on `see_inconsistent_state` port. In particular, unrecoverable errors on EDAC or invalid state detected at FSMs is reported:

- EDAC on RAMs is able to detect and recover a single error in a data word. It is also able to detect (with no correction) two errors in a data word, in this case `see_inconsistent_state` signal is asserted.
- when an internal FSM moves to a forbidden state because of a SEE (forbidden states in general exist because of the state coding or because the number of states is not a power of two), `see_inconsistent_state` is asserted. Registers that represent the states of internal FSMs are anyway TMR protected, this lowers the probability that this case may occur.

When `see_inconsistent_state` is asserted, IP Core outputs may produce wrong data or can be stuck in an unexpected state, to restore the IP Core to a valid state a reset pulse is needed in this case.

Once `see_inconsistent_state` signal is asserted, only a reset pulse can de-assert it.

The configuration and status interface is synchronous to the core clock, `core_clk`.

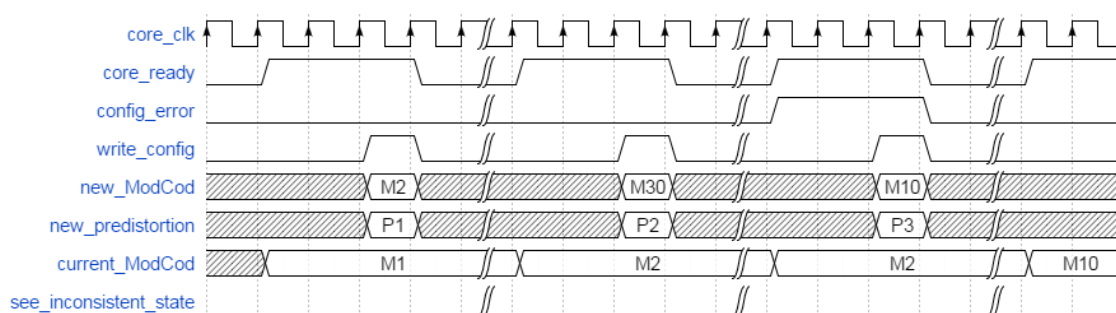


Figure 5-4 Configuration and status interface timing

5.4 LUT loader interface

Port name	I/O	bits	Clock domain	Description
LUT_clk	I	1	LUT_clk	Clock for LUT loader section.
LUT_addr	I	16	LUT_clk	Address for writing the LUTs to the IP-core.
LUT_data	I	32	LUT_clk	Data to write the LUTs to the IP-core
LUT_ready	O	1	LUT_clk	Indicates that the LUT loader section is ready to receive new data. If LUT_we is asserted while LUT_ready is asserted, then the write operation succeeds. Otherwise, if LUT_we is asserted while LUT_ready is de-asserted, then the write operation fails and the host system has to repeat it. LUT_ready allows the integration in systems where LUT_clk frequency is higher than core_clk frequency.
LUT_we	I	1	LUT_clk	When asserted on the rising edge of core_clk, LUT_data is written to LUT_addr memory location.
LUT_req	O	1	LUT_clk	When asserted, the IP core requests the LUTs to be written to its internal memory, for the requested LUT_ModCod.
LUT_load_complete	I	1	LUT_clk	Asserted when the external LUT loader has finished writing data to the IP core, therefore the IP core exits the configuration state.
LUT_ModCod	O	5	LUT_clk	Indicates the ModCod related to the LUTs to be loaded to the IP core.

Table 5-4 LUT loader interface

The LUT loader interface allows the IP core to load LUT data from an external memory, so that hardware complexity can be considerably reduced. When a soft configuration is initiated and LUTs are not stored internally, the IP core requests new LUT values by asserting `LUT_req` signal, specifying the ModCod required on the `LUT_ModCod` port. Then, the IP core expects that data is written onto its internal RAM with the signals `LUT_addr`, `LUT_data` and `LUT_we`, according to the memory map represented in Table 4-1. When the LUT loading is complete, the IP core expects `LUT_load_complete` asserted to continue its operations.

The LUT loader interface is synchronous to `LUT_clk` clock signal, in order to ease the integration with external memory. All cross-domains paths between `LUT_clk` and `core_clk` are managed internally to the IP Core.

This interface does not address a specific memory technology, a further bridging logic has to be implemented in order to perform reads commands through a specific memory interface (e.g. SPI or i2c).

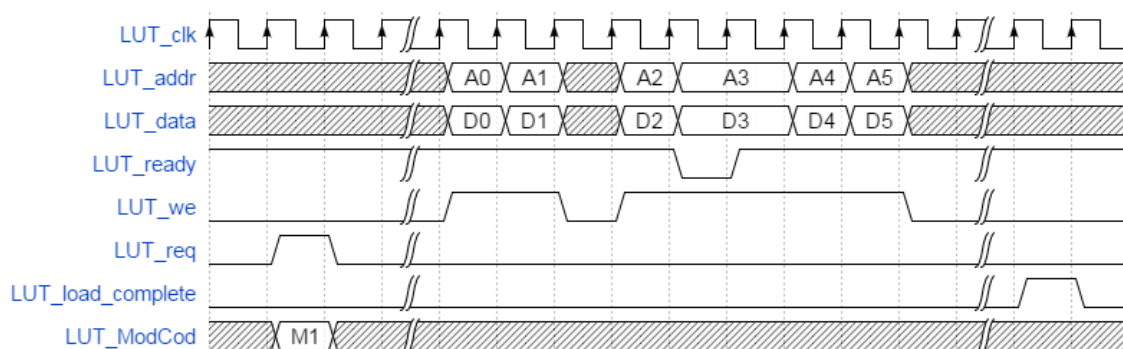


Figure 5-5 LUT loader interface timing

5.5 Clock, reset and enable

Port name	I/O	bits	Clock domain	Description
core_clk	I	1	core_clk	IP core clock
rst_n	I	1	core_clk	Global reset signal (synchronous/asynchronous, active-low/active-high, are based on hard configuration parameters)
enable	I	1	core_clk	Active-high enable signal

Table 5-5 Clock, reset and enable interface

6 Hard configuration options

Table 6-1 shows the hard configuration parameters for the IP core.

Hard configuration	Allowed values	Description
Supported ModCods	[0x0000001, 0x7FFFFFFF]	Specifies the ModCods supported by the specific instantiation of the IP Core.
Default ModCod	{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27}	Specifies the ModCod selected right after reset.
Buffering type	{Single, Double}	Specifies the buffering strategy at CC1 interleaver. Double buffering allows full performance with higher buffer usage, while single buffering allows lower hardware resource usage with reduced performance.
Output Symbol parallelism (P_S)	{1, 2, 4}	Number of symbols that the IP-core produces at its output in parallel.
Lookup tables	{Internal, External}	Specifies whether the look-up tables for the implementation of the interleaving and puncturing functions is stored internally or must be loaded from an external non-volatile memory.
SEE mitigation	{ON, OFF}	Enables/disables the SEE mitigation techniques in the IP Core.
Pilot insertion	{ON, OFF}	Enables/disables the pilot insertion in the PL framing.
SRRC filter enable	{ON, OFF}	Enables/disables the SRRC filtering.
SRRC filter roll-off	{0.2, 0.25, 0.3, 0.35}	Specifies the roll-off for the SRRC filter.
Static symbol predistortion type	{OFF, ON}	Enables/disables symbol predistortion.
Static symbol predistortion coefficients	10 real values in (-1.2,1.2) range	Specifies the constant complex multiplicative factor for static symbol predistortion used after reset.
Reset polarity	{active-high, active-low}	Specifies the polarity of the reset signal
Reset type	{Synchronous, Asynchronous}	Specifies whether the reset is asynchronous or synchronous
Scrambling code number	[0, 2 ¹⁸ -2]	Specifies the scrambling number used by the pseudo-randomizer
Size of the input FIFO buffer	Integer power of 2's	Specifies the size of the input FIFO in memory locations (32-bit words).
Architecture parallelism at CC1	{2,4,6,8,12,24}	Specifies the parallelization degree at the outer encoder (CC1). This parameter depends on the setting of P_S
Architecture parallelism at CC2	{3,4,5,6,8,10,12,15,20,24,30,40,60,120}	Specifies the parallelization degree at the inner encoder (CC2). This parameter depends on the setting of P_S

Hard configuration	Allowed values	Description
Architecture parallelism at CC1 interleaver	{2,3,4,5,6,8,10,12,15,20,24,30,40,60,120}	Specifies the parallelization degree at CC1 interleaver. This parameter depends on the setting of P_S
Pipeline registers	{OFF, ON}	Specifies the presence of pipeline registers in different places at the architecture, to help with timing closure

Table 6-1 IP core hard configuration parameters

7 Usage

As the CCSDS 131.2-B standard is rather complex and with a very wide scope, the CCSDS telemetry transmitter IP core is massively configurable, so that the final user is provided with a powerful and flexible IP Core able to adapt to a wide range of applications. In general, a high number of generic parameters necessary to guarantee large flexibility, often hinders the usability in the hands of the final user, as a deep understanding of the architectural features is necessary to set all the parameters in a correct way.

In order to overcome this issue, besides having all the generic parameters in the VHDL top-level, the IP core is provided with an IP-core generator software with a simple Graphical User Interface (GUI) to guide the end-user to the IP configuration process, see Figure 7-1. Within the tool, the architectural parameters are translated at system-level, so that the user can exploit the most advanced features of the IP core without the deep knowledge required at architectural level. This will of course reduce IP core integration time while at the same time minimizing the risk of mistakes in the IP core configuration.

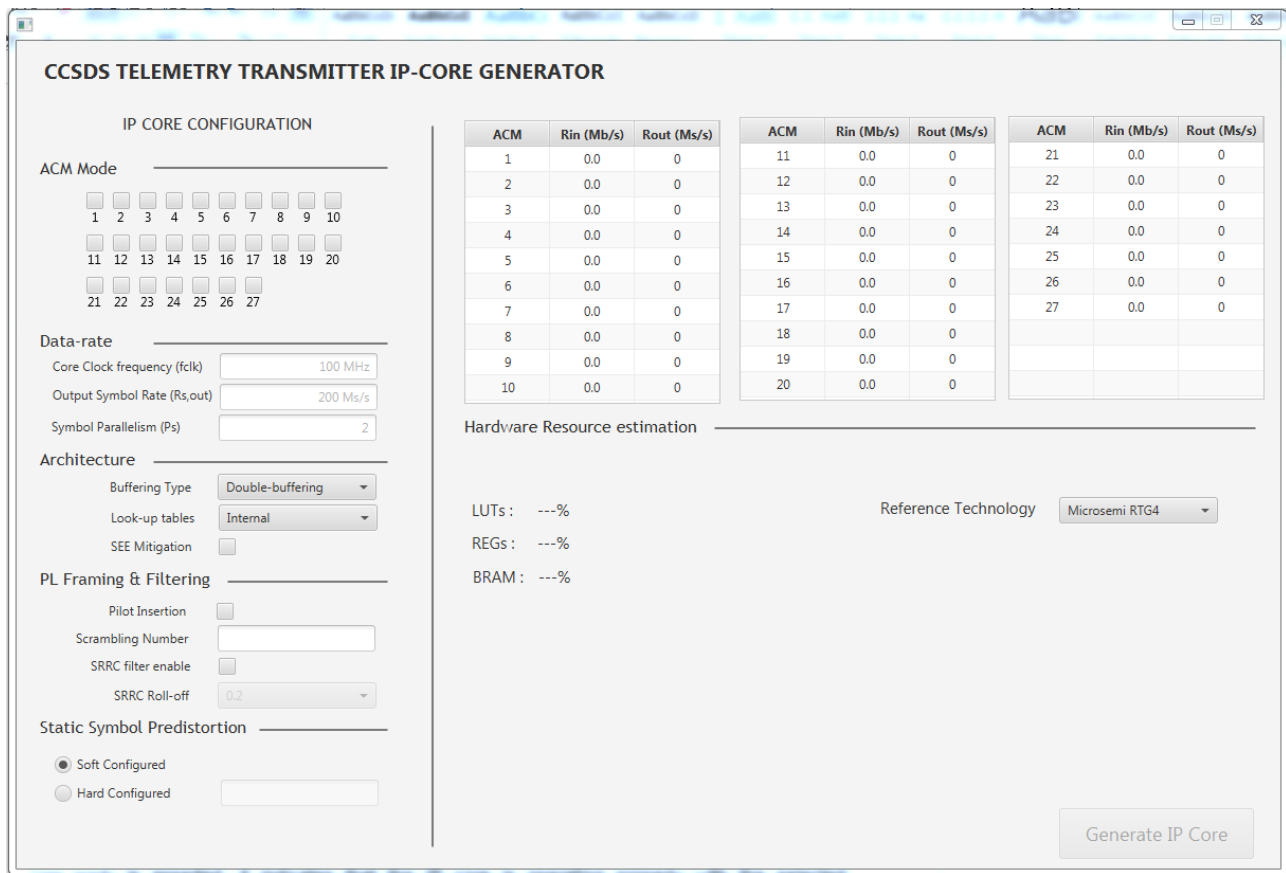
The end-user can then trade-off hardware resource occupation vs performance/functionality and set of ModCods supported in order to have an efficient implementation for the specific application and fit the IP-core onto the target technology (even on relatively small FPGAs such as Microsemi RTAX2000).

The IP-core generation tool guides the end-user towards the generation of a fully functional IP-core, giving estimations on the performance and complexity of the fitted instantiation. Note that the configuration system parameters are not all independent and not all the combinations generate a valid implementable IP Core. However the software takes care of this aspect automatically so the end-user can start configuring the IP-core by setting any of the parameters and while the configuration goes on, the non-compatible options will be grayed-out by the software. For example, if the user selects the Microsemi RTAX2000S as target technology with high ModCods selected, then the 200 Ms/s output baud-rate will be grayed-out.

The IP core user can configure the IP-core through the following parameters:

- **ModCods to be supported:** by acting on the check-boxes the user can enable the desired ModCods. The supported ModCods have a substantial impact on the architecture complexity as:
 - The size of the internal buffers is given by the maximum ModCod to be supported
 - Each ModCod involves the use of large look-up tables for the implementation of the SCCC processing
 - The parallelism of each stage in the processing chain depends on the ModCods selected, since the required internal data-rate is different for each ModCod.

The IP-core generator software will automatically select the optimal architecture and RAM size based on the selected ModCods, in order to have an efficient hardware implementation.



CCSDS TELEMETRY TRANSMITTER IP-CORE GENERATOR

IP CORE CONFIGURATION

ACM Mode

1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27

Data-rate

Core Clock frequency (fclk) 100 MHz
Output Symbol Rate (Rs,out) 200 Ms/s
Symbol Parallelism (Ps) 2

Architecture

Buffering Type Double-buffering
Look-up tables Internal
SEE Mitigation ☐

PL Framing & Filtering

Pilot Insertion ☐
Scrambling Number
SRRC filter enable ☐
SRRC Roll-off 0.2

Static Symbol Predistortion

☒ Soft Configured
☐ Hard Configured

ACM	Rin (Mb/s)	Rout (Ms/s)
1	0.0	0
2	0.0	0
3	0.0	0
4	0.0	0
5	0.0	0
6	0.0	0
7	0.0	0
8	0.0	0
9	0.0	0
10	0.0	0
11	0.0	0
12	0.0	0
13	0.0	0
14	0.0	0
15	0.0	0
16	0.0	0
17	0.0	0
18	0.0	0
19	0.0	0
20	0.0	0
21	0.0	0
22	0.0	0
23	0.0	0
24	0.0	0
25	0.0	0
26	0.0	0
27	0.0	0

Hardware Resource estimation

LUTs: ---%
REGs: ---%
BRAM: ---%

Reference Technology Microsemi RTG4

Generate IP Core

Figure 7-1 IP core generator Graphical User Interface

- **Data-rate**, which is defined by:

- The **clock frequency** of the IP-core (f_{ck})
- The **output symbol-rate** of the IP-core ($R_{s,out}$)
- The output **symbol parallelism**, i.e., how many symbols the IP-core provides in parallel in one clock cycle (P_s)

These parameters are not independent as the following relation holds:

$$R_{s,out} = f_{ck} \cdot P_s$$

which forces one parameter when the other two are set. The GUI allows the user to set any couple of parameters and will automatically calculate the remaining one, also providing warnings/errors in case of conflicts with other parameters, e.g., an infeasible clock frequency selected on a particular technology.

- **Architecture**, which involves:
 - The **buffering type**, which can be **double-buffering** or **single-buffering**. This setting affects buffering strategy at CC1 interleaver, which is critical because the whole data block must be processed before the next processing step can be initiated.

Double buffering allows continuous processing as a ping-pong strategy is used. Double-buffering comes at the price of higher hardware complexity due to replicated buffers.

On the other hand, with the single-buffering approach, performance is sacrificed in order to reduce hardware complexity. In this case, buffers are either written or read. Single-buffering cuts in a half the architecture throughput, but can be the only viable solution to implement higher ModCods on low capacity FPGA (e.g. Microsemi RTAX).
 - **Look-up tables** for the implementation of the interleaving function and puncturing can be stored **internally** into the IP-core or the IP-core can be configured to load the LUTs from an **external** non-volatile memory. The CCSDS standard foresees an extensive use of LUTs, which are different for each ModCod, and therefore internal LUTs might not fit into the device if many ModCods are selected. By using an external non-volatile memory, this potential issue is solved as the parameters needed are loaded at start-up into the internal memory resources of the device.
 - **SEE mitigation**, through which the user can activate SEE mitigation techniques in the IP Core, such as EDAC on the RAMs.
- **Target silicon technology** for implementation, which allows an estimation of the IP-core resource occupation on the selected device. Additionally, in case the device is fixed and cannot be changed for a certain application, the user can select the target technology as the first parameter, so that only the compatible options with this choice will be allowed by the GUI (feasible clock frequency, ModCods supported, etc.).
- **PL Framing & Filtering** allows the user:
 - to enable the pilot insertion in the PL framing,
 - to choose the scrambling number (n) for the pseudo-randomizer
 - to enable SRRC filtering and the roll-off used in the implementation

In addition to the configuration parameters, the GUI shows the performance achieved by the selected instantiation in terms of:

- The **input data-rate** for each ModCod selected
- The **output symbol-rate** for each ModCod selected (the same as the one selected as parameter)
- The **hardware resources usage estimation** on the selected target technology

8 Technology Mapping

This section contains the results of technology mapping of the IP-core on the reference technologies. As the configuration possibilities of the IP core are a very high number (2^{27} just considering all the combinations of ModCods), a representative set of results are reported here to give an idea of the IP core complexity and performance.

Table 8-1 shows an overview of the ModCods that can be implemented for each technology considering single ModCod support. Table 8-1 is based on synthesis results for the IP core.

ModCod	Microsemi RTAX2000		Microsemi ProASIC		Microsemi RTG4			Xilinx Virtex 5 QV			NanoXplore NG-MEDIUM	
	Ps = 1	Ps = 2	Ps = 1	Ps = 2	Ps = 1	Ps = 2	Ps = 4	Ps = 1	Ps = 2	Ps = 4	Ps = 1	Ps = 2
1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
3	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
4	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
5	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
7	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
9	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
10	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
11	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
12	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
13	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
14	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
15	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
16	✓		✓	✓	✓	✓	✓	✓	✓	✓	✓	
17			✓	✓	✓	✓	✓	✓	✓	✓	✓	
18			✓	✓	✓	✓	✓	✓	✓	✓	✓	
19			✓	✓	✓	✓	✓	✓	✓	✓	✓	
20			✓	✓	✓	✓	✓	✓	✓	✓	✓	
21			✓	✓	✓	✓	✓	✓	✓	✓		
22			✓	✓	✓	✓	✓	✓	✓	✓		
23			✓	✓	✓	✓	✓	✓	✓	✓		
24			✓	✓	✓	✓	✓	✓	✓	✓		
25			✓	✓	✓	✓	✓	✓	✓	✓		
26			✓		✓	✓	✓	✓	✓	✓		
27			✓		✓	✓	✓	✓	✓	✓		

Table 8-1 ModCod support on the different technologies

Regarding multiple-ModCod instantiations, the IP core was characterized for incremental configurations starting from ModCod 1 ([1], [1,2], [1,2,3] ... [1,2,3...,26, 27]) to find the maximum number of multiple ModCods supported in a single instantiation. Of course, the results presented here do not exclude multiple-ModCod instantiation with higher order ModCods by removing support for some lower order ModCods. Again, this assumption was made to limit the configuration space of the IP core. Table 8-2 shows multi-ModCod IP core instantiation synthesis results on the target technologies.

Technology	ModCods	Ps	LUT	Buffering	SRRC	Comb / Tiles	Regs	BRAM	DSP
Virtex 5 QV	ALL	2	INT	DOUBLE	YES	46%	24%	19%	3%
Virtex 5 QV	ALL	2	EXT	DOUBLE	YES	33%	23%	9%	3%
Virtex 5 QV	1 to 25	4	INT	DOUBLE	YES	72%	55%	26%	5%
Virtex 5 QV	ALL	4	EXT	DOUBLE	YES	68%	55%	13%	5%
RTG4	ALL	2	INT	DOUBLE	YES	78%	21%	21%	24%
RTG4	ALL	2	EXT	DOUBLE	YES	28%	17%	24%	24%
RTG4	1 to 24	4	INT	DOUBLE	YES	80%	31%	29%	37%
RTG4	ALL	4	EXT	DOUBLE	YES	55%	39%	34%	50%
ProASIC RT	1 to 8	1	INT	DOUBLE	NO	58%		31%	
ProASIC RT	1 to 20	1	EXT	DOUBLE	NO	44%		75%	
ProASIC RT	1 to 9	2	INT	DOUBLE	NO	44%		51%	
ProASIC RT	1 to 12	2	EXT	DOUBLE	NO	52%		76%	
RTAX2000	1 to 4	1	INT	DOUBLE	NO	69%	37%	31%	
RTAX2000	1, 2, 3, 7, 8, 9, 10, 13	1	EXT	DOUBLE	NO	60%	50%	78%	
NG-MEDIUM	1 to 8	1	INT	DOUBLE	YES	94%	12%	55%	92%
NG-MEDIUM	1 to 16	1	EXT	DOUBLE	YES	95%	16%	82%	96%
NG-MEDIUM	1 to 5	2	INT	DOUBLE	NO	62%	13%	52%	0%
NG-MEDIUM	1 to 9	2	EXT	DOUBLE	NO	95%	17%	80%	0%

Table 8-2 Multiple-ModCod instantiation synthesis results on the different technologies

Table 8-3 shows some IP core post place & route results. These results were obtained by targeting both a relevant number of supported ModCods within a single instantiation and 200 Mbaud for Virtex 5, RTG4 and NG-MEDIUM, and 100 Mbaud for RTAX2000 and ProASIC. Of course, a higher symbol rate can be achieved by reducing the number of supported ModCods and, conversely, a higher number of ModCods in a single instantiation can be achieved by targeting a lower symbol rate. Finally, results on commercial Xilinx Kintex 7 and Ultrascale devices are presented.

Technology	ModCods	Ps	LUT	Buffering	SRRC	Max clock freq (MHz)	Max input data rate (Mb/s)	Max output Symbol Rate (Mbaud)
RTAX2000	1 to 3	4	INT	DOUBLE	NO	30	124	120
ProASIC RT	1 to 2	4	INT	DOUBLE	YES	30	103	120
Virtex 5 QV	1 to 24	4	INT	DOUBLE	YES	50	888	200
Virtex 5 QV	25 to 27	4	INT	DOUBLE	YES	50	1078	200
RTG4	1 to 18	4	INT	DOUBLE	YES	56	716	224
Virtex 5 QV	1, 7, 13, 18, 23	2	INT	DOUBLE	YES	76	626	152
RTG4	1, 7, 13, 18, 23	2	INT	DOUBLE	YES	78	642	156

NG-MEDIUM	1 to 4	1	INT	DOUBLE	NO	18	22	18
NG-MEDIUM	1 to 14	1	EXT	DOUBLE	NO	21	55	21
NG-MEDIUM	1 to 4	2	INT	DOUBLE	NO	20	49	40
NG-MEDIUM	1 to 4	2	EXT	DOUBLE	NO	23	56	46
NG-MEDIUM	1 to 4	1	INT	DOUBLE	SI	24	29	24
NG-MEDIUM	1 to 8	1	EXT	DOUBLE	SI	22	36	22
Kintex 7 Ultrascale xcku15pffva1156-2	ALL	4	INT	DOUBLE	YES	138	2980	553
Kintex 7 xc7k325t-ffg900	ALL	4	INT	DOUBLE	YES	117	2538	471

Table 8-3 IP core post place and route performance

9 BER performance

Figure 9-1, Figure 9-2, and Figure 9-3 show the coded BER performance of the CCSDS 131.2-B Telemetry Transmitter IP core. The results were obtained by means of simulation with the bit-true model of the IP core, with SRRC filter enabled and ideal receiver. The floating point reference curve is also reported to evaluate the implementation loss, which overall is below 0.2 dB.

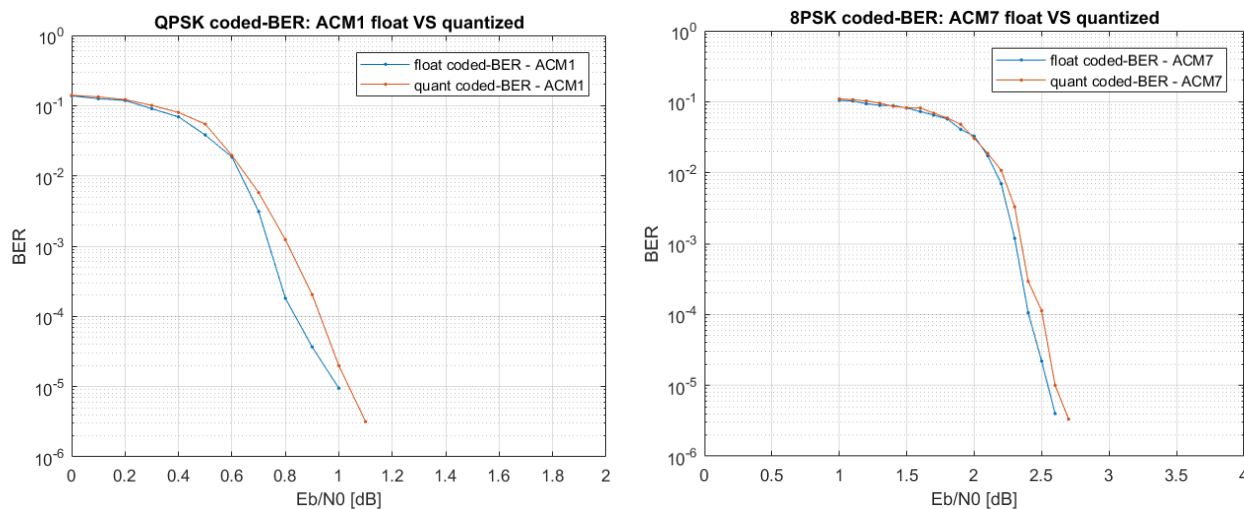


Figure 9-1 IP core BER performance for ACM1 and ACM7

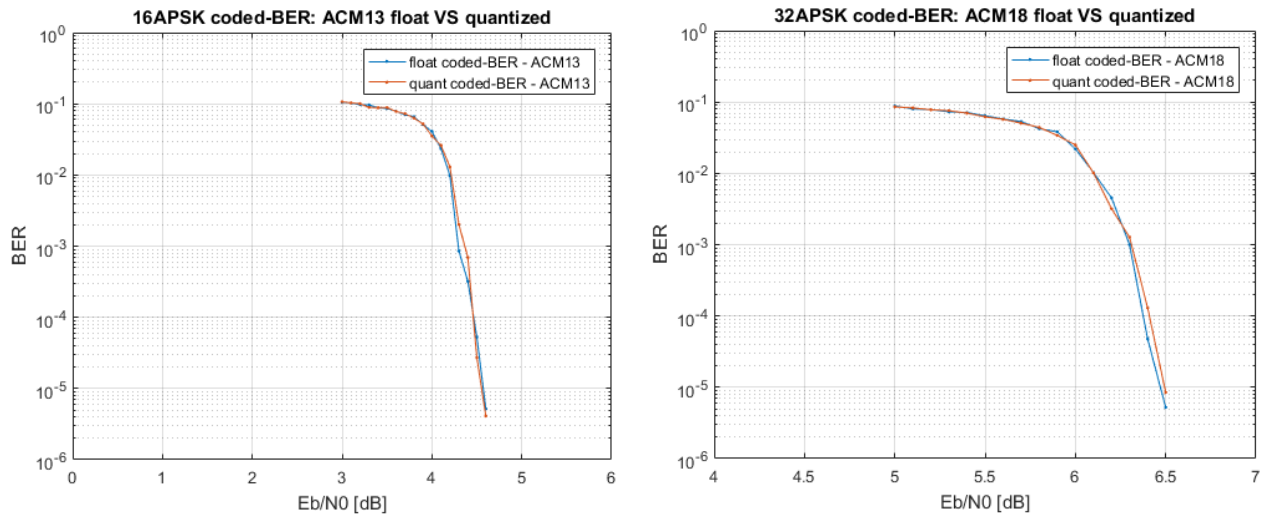


Figure 9-2 IP core BER performance for ACM13 and ACM18

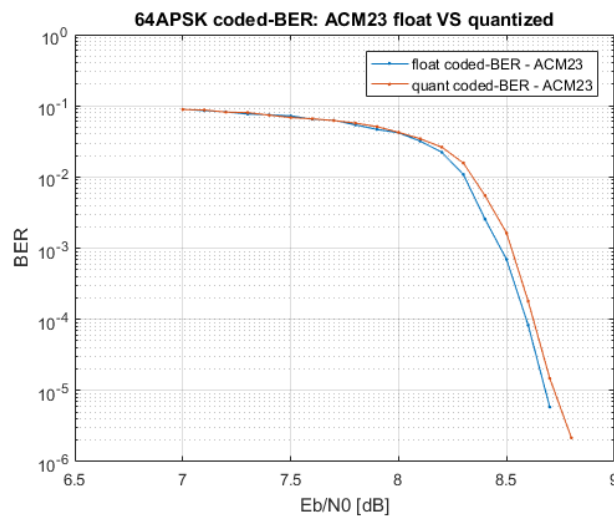


Figure 9-3 IP core BER performance for ACM23