LogiCORE IP DVB-S.2 FEC Encoder v2.0

DS505 December 2, 2009 **Product Specification**

Introduction

The Xilinx® DVB-S.2 FEC Encoder core provides designers with a Forward Error Correction (FEC) Encoding block for DVB-S.2 systems.

Features

- Drop-in module for Virtex®-6, Virtex-5, Virtex-4, Spartan®-6, Spartan-3/XA, Spartan-3E/XA and Spartan-3A/AN/3A DSP/XA FPGAs
- Forward Error Correction for DVB-S.2 (compatible with ETSI EN 302 307 V1.1.2 (2006-06)) supports:
  - Normal and short frames
  - All code rates
  - Input and output buffer, BCH outer coding, LDPC encoder, and bit-interleaver
- Support for Constant Coding and Modulation (CCM) and Adaptive Coding and Modulation (ACM) operation.
- Optional input and output formatting to ease system integration
- Fully optimized for speed and area
- Fully synchronous design using a single clock
- For use with the Xilinx® CORE Generator™ v11.4 software and higher

Overview

The DVB-S.2 FEC Encoder core provides a complete Forward Error Correction (FEC) encoding solution for DVB-S.2. Figure 1 illustrates a block diagram of the main blocks: the Outer (BCH), Inner (LDPC) encoding, and bit-interleaving stages. The core is also buffered on both the input and the output.

The encoder supports all rates and both normal and short frames. Each encoding stage adds a number of bits dictated by the coding rate (where each k and n in Figure 1 refers to the number of input and output bits at each stage, respectively). The final number of coded output bits (n_{ldpc}) is 64,800 for a normal frame, and 16,200 for a short frame. For reference purposes, the number of uncoded input bits in each frame (k_{bch}) is provided for each code rate in Table 1.

The bit interleaver supports the interleaving required by all modulation types, which may be QPSK, 8PSK, 16APSK, or 32APSK.

Rate, frame size, and modulation can be changed on a frame-by-frame basis. Two levels of throughput are supported. To ease system integration data may be input as bytes with a bit ordering compatible with MPEG, and output as symbols for direct connection to a mapper.

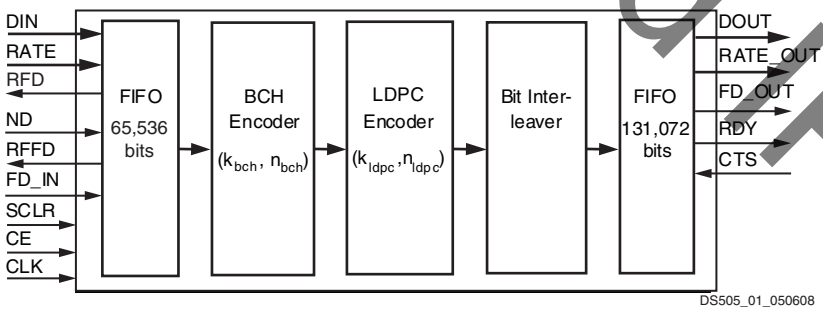

Figure 1: **Block Diagram of DVB-S.2 FEC Core**

Table 1: **Uncoded Frame Length**

Code Rate	Short frame	Normal frame
	k_{bch}	k_{bch}
1/4	3072	16008
1/3	5232	21408
2/5	6312	25728
1/2	7032	32208
3/5	9552	38688
2/3	10632	43040
3/4	11712	48408
4/5	12432	51648
5/6	13152	53840
8/9	14232	57472
9/10	Not defined	58192

Pinout

Table 2 summarizes the signals illustrated in Figure 1. All control signals are active High.

Table 2: **Core Signal Pinout**

Name	Port Width	Direction	Active State	Description
DIN	Win=1, 4 or 8	INPUT		**Data Input**: Message to be encoded, presented in slices of 1 or 4 bits when raw input format is adopted, or 8 bits when MPEG byte-wide format is adopted.
RATE	16	INPUT		**Rate Input**: Parameters to be used to encode frame (frame length, code rate, modulation information, pilot insertion and frame identification).
ND	1	INPUT	High	**New Data**: Set High when data is valid on DIN.
RFD	1	OUTPUT	High	**Ready For Data**: Set High by the core when it is ready for data on DIN.
FD_IN	1	INPUT	High	**First Data**: Set High to indicate the start of a frame of data input.
RFFD	1	OUTPUT	High	**Ready For First Data**: Set High by the core when ready to accept FD_IN.
CE	1	INPUT	High	**Clock Enable** (optional).
SCLR	1	INPUT	High	**Synchronous Reset** (optional).
CLK	1	INPUT	Rising edge	Clock
DOUT	Wout = 1, 4 or 5	OUTPUT		**Data Output**: Encoded message output, provided in slices of 1 or 4 bits when raw output format is adopted, or 5 bits when symbol format is adopted.
RATE_OUT	16	OUTPUT		**Rate Output**: Rate and frame identification for current output frame.

Table 2: **Core Signal Pinout *(Cont'd)***

Name	Port Width	Direction	Active State	Description
RDY	1	OUTPUT	High	**Ready**: Set High by core when output is valid on DOUT.
CTS	1	INPUT	High	**Clear To Send**: Set High to enable output of data over DOUT. Set Low to stall data output.
FD_OUT	1	OUTPUT	High	**First Data Output**: Set High by the core to indicate that the current data output is the first item of a frame.

DIN

The uncoded data input to the core is supplied via DIN. The width of data input depends upon the input format and throughput specified when the core is generated. For more information, see "CORE Generator Software Parameters," page 8.

RATE

The run-time parameters used to encode a particular frame of data, along with a frame identification number, if required, are supplied to the core via RATE. See "Run-Time Parameters," page 6 for more information about the bit-encoding adopted. This port is read by the core on the start of a frame. See the following description of "FD_IN" for when a frame will start.

FD_IN

FD_IN should be asserted to start the encoding of a frame of data, as shown in Figure 2. Note that both ND and FD_IN must be asserted while the core is ready for a new frame (that is, while RFFD is asserted) for the core to start. Data will be accepted with a valid FD_IN as shown in the figure. Note that RFFD is only asserted with RFD, and so the start of frame is not conditional on RFD.

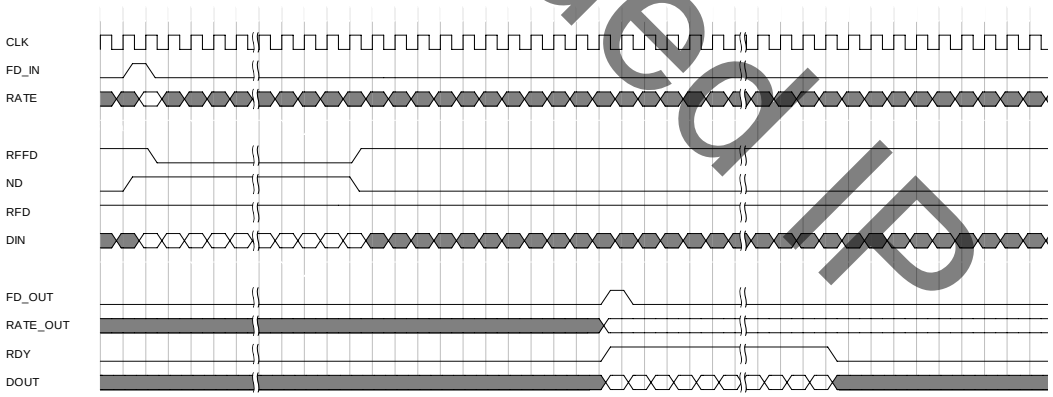

Figure 2: **Control Signals on Data Input and Output**

ND

ND should be asserted to indicate that data is valid on DIN. Deasserting ND causes inputs to be ignored, as shown in Figure 3 (where the greyed-out data inputs are those ignored by the core). Note that data is only input when both ND and RFD are asserted once the core has started.

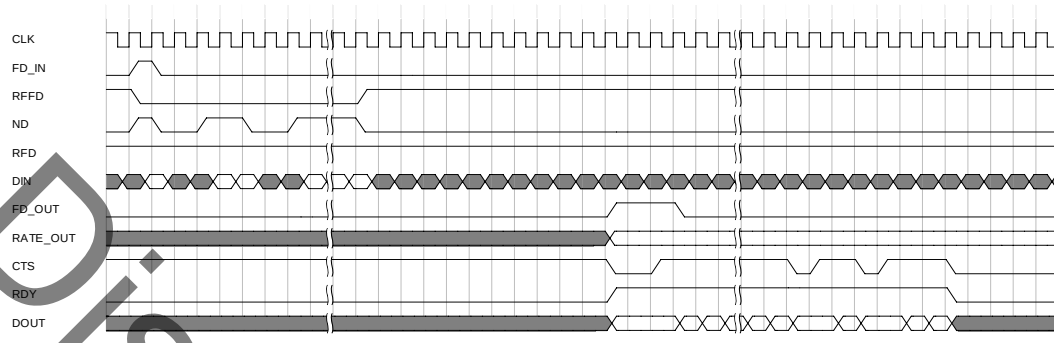

Figure 3: **End-to-End Flow Control with the ND and CTS Signals**

RFD

RFD is asserted by the core to indicate that it is ready to accept a sample of data on DIN. Data is input when both ND and RFD are asserted once the core has started.

RFFD

The RFFD output is asserted by the core when it is ready for a new frame of data, as shown in Figure 4. Note RFFD is only asserted when the core is also ready for data (that is, when RFD is asserted), and after a frame of input has started RFFD is deasserted until the whole frame has been input.

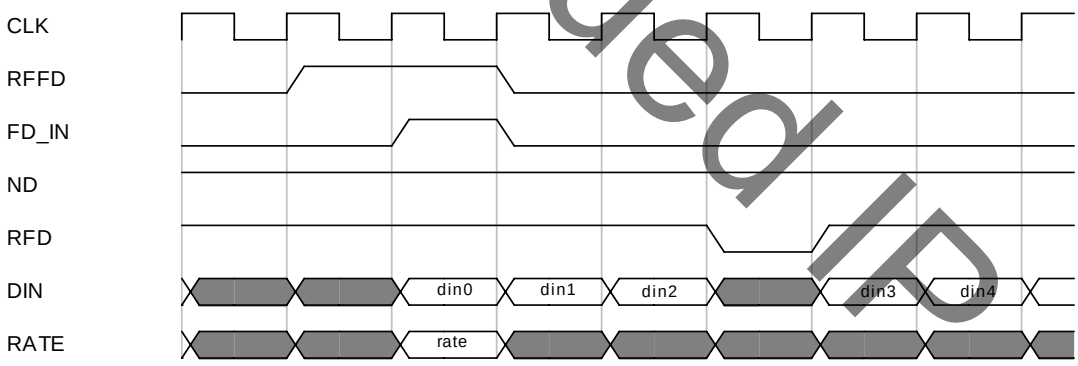

Figure 4: **FD_IN Input with RFFD (ND Must be High for FD_IN to be Valid)**

Note that FD_IN may be permanently asserted in order to start frames as soon as the core is ready to start a new frame (that is, when RFFD is asserted) and the input buffer is ready for data (that is, RFD is asserted).

CE

CE is an optional signal. When CE is deasserted, all inputs, including SCLR, are ignored and the core remains in its current state.

SCLR

SCLR is an optional signal. When SCLR is asserted the core is synchronously set to its initial state. Any intermediate results are discarded. It takes one cycle to achieve this initial state, and the state is maintained until SCLR is deasserted. The initial state is where the core is ready for FD_IN (that is, both RFFD and RFD are asserted).

If CE is present then it needs to be asserted with SCLR to reset the core.

CLK

All signals are synchronous to the CLK input.

DOUT

The encoded frame is output on DOUT. The width of data output depends upon the output format and throughput specified when the core is generated. For more information, see "CORE Generator Software Parameters," page 8.

RATE_OUT

The run-time parameters used to encode a particular frame of data along with frame identification if required are output via RATE_OUT (using the same encoding as RATE). RATE_OUT is valid when FD_OUT is asserted and its value is maintained until the next FD_OUT.

FD_OUT

FD_OUT is asserted when the first item of data in a frame is presented on DOUT. It remains asserted until that item of data is accepted (that is, when CTS is asserted along with FD_OUT). Note that RDY is always asserted with FD_OUT.

RDY

RDY is asserted by the core to indicate that data on DOUT is valid.

CTS

The output from the core is stalled when CTS is deasserted, as shown in Figure 3. The outputs FD_OUT, RDY, RATE_OUT, and DOUT maintain their previous state. DOUT is valid when both RDY and CTS are asserted.

Run-Time Parameters

All the values required by the core are supplied through the input port, RATE. This 16-bit bus is composed of 5 fields as shown in Figure 5.

Bit	15-8	7	6	5	4	3	2	1	0
	FI	PI	MT		FS	CR			

Figure 5: **Bit Fields within RATE Input Signal**

Code Rate (CR)

The supported code rates are those defined in the DVB S.2 standard. Each Code Rate is assigned the values specified in Table 3. Note that for short frames the rate 9/10 is not defined by the standard. If this rate is requested along with short frames, then the behavior of the core is undefined. The behavior is also undefined if values of CR greater than 10 are employed.

Table 3: **Valid Values for the Code Rate Field of RATE**

CR Values (RATE[3:0])	Code Rate	
	Short Frame	Normal Frame
0	1/4	
1	1/3	
2	2/5	
3	1/2	
4	3/5	
5	2/3	
6	3/4	
7	4/5	
8	5/6	
9	8/9	
10	RESERVED	9/10
11-15	RESERVED	

Frame Size (FS)

The Frame Size can be either normal or short. Associated values are specified in Table 4.

Table 4: **Valid Values for the Frame Size Field of RATE**

FS Values (RATE[4])	Frame Size
0	Normal
1	Short

Modulation Type (MT)

The Modulation Type can take four values, and are encoded as defined in Table 5. Selecting QPSK (MT=0) disables bit interleaving.

Table 5: **Valid Values for the MT Field of the RATE**

MT Values (RATE[6...5])	Modulation
0	QPSK
1	8PSK
2	16APSK
3	32APSK

Pilot Insertion (PI)

The Pilot Insertion bit is not used by the core but can be used as a marker pilot bit for the current frame. The run-time parameters used to encode a particular frame of data along with PI are output via RATE_OUT.

Frame Identification (FI)

The Frame Identification bits are not used by the core but can be used as an identification for the current frame. The bits can be used as general purpose control or signalling bits which maintain synchronization with frames of data as they pass through the core. The run-time parameters used to encode a particular frame of data along with FI are output via RATE_OUT.

CORE Generator Software Parameters

The CORE Generator software provides a Graphical User Interface (GUI) for generating the core. The following customization options are available (the XCO parameter names, and associated values are provided in brackets):

- **Component Name:** The user-defined component name. The name must begin with a letter and be composed of the following characters: a to z, A to Z, 0 to 9, and "_".
- **Throughput (Width)**: Specifies the number of bits processed by the core; can be 1 or 4. When Width is 4, the processing logic is parallel and provides greater throughput, but uses more resources.
- **Input format (Ip_Format)**: Specifies whether the input should be the raw message (Raw) with width 1 or 4 bits, depending upon the Width chosen, or byte-wide (MPEG_Bytes).
- **Output format (Op_Format)**: Specifies whether the output should be the raw encoded message (Raw) with width 1 or 4 bits, depending upon the Width chosen, or symbols (Symbol).
- **Input Buffer Mode (Ip_Buffer_Mode)**: Specifies whether a whole frame should be input before processing starts (Wait_For_Whole_Frame) or whether data should be processed as soon as possible (Process_ASAP). Processing the data as soon as possible allows the latency of the core to be reduced.
- **Output Buffer Size (Op_Buffer_Size)**: Optimizes the size of the output buffer for either constant frame size (Constant_Frame_Size) or changing frame size (Adaptive_Frame_Size). The latter results in greater memory requirement, but ensures that throughput is maintained when frame size changes.
- **Optional Pins:** SCLR (**Has_Sclr**) and CE (**Has_Ce**). Select to enable these pins (True), or leave unselected to leave them off the core (False).

Further information on some of these parameters follows:

Input Format

Raw Input Format

When the input format is Raw, then the bit-width of the data input, DIN, is given by Width, and is either 1 or 4 bits.

When Width = 1, the data to be encoded is input a bit at a time, starting with the first bit of the uncoded frame, m(0).

When Width = 4, 4 bits are input at a time, where for each input i, DIN(3 downto 0) provides uncoded frame bits m(4*i+3), m(4*i+2), m(4i+1), and m(4i), respectively. Note that DIN(0) provides the first bit of the frame.

MPEG_Bytes Input Format

When the input format is MPEG_Bytes, the data to be encoded is input 8 bits at a time. For an input i, DIN(7 downto 0) provides bits m(8i), m(8i+1), m(8i+2), m(8i+3), m(8i+4), m(8i+5), m(8i+6), and m(8i+7), respectively. Note that the order is the reverse of Raw data, but is representative of the order expected at the decoder when transmitting MPEG data.

Output Format

Raw Output Format

When the output format is Raw, then the bit-width of the data output, DOUT, is given by Width, and is either 1 or 4 bits.

When Width = 1, the encoded frame is output one bit at a time, starting with m(0).

When Width = 4, then 4 bits are output at a time. For each output i, DOUT(3 downto 0) provides bits m(4*i+3), m(4*i+2), m(4i+1), and m(4i) of the encoded frame, respectively. Note that DIN(0) provides the first bit of the encoded frame.

Symbol Output Format

When the output format is Symbol, DOUT has width 5-bits and provides the encoded frame as symbols whose width depends upon the modulation specified for the frame. The ordering of bits for each modulation type is summarized in Table 6.

Table 6: **Bit-Mapping of DOUT to Symbols for each Modulation Type**

Rate	QPSK	8PSK	16APSK	32APSK
DOUT(0)	2i+1	3i+2	4i+3	5i+4
DOUT(1)	2i	3i+1	4i+2	5i+3
DOUT(2)	Logic 0	3i	4i+1	5i+2
DOUT(3)	Logic 0	Logic 0	4i	5i+1
DOUT(4)	Logic 0	Logic 0	Logic 0	5i

Note that the bit ordering is reversed over Raw output format. For example, when 16APSK is adopted, the symbol output is only 4 bits, and DOUT(3 downto 0) provides encoded frame bits m(4*i), m(4*i+1), m(4i+2), and m(4i+3). This is the bit-reversal of that provided by Raw output format when Width = 4.

System Integration

Data Input

Flow of data into the core is controlled at two levels. The core input FD_IN and output RFFD provide a handshake to control the start of each frame. During a frame, the input ND and output RFD provide a handshake to control the flow of data into the core on a cycle-by-cycle basis.

Note that data must be available when the core is started, so ND must be asserted with FD_IN (while RFFD is asserted by the core) for the input of a frame to start.

Also note, if the core is ready for a new frame (RFFD is asserted by the core), then asserting ND without FD_IN does not result in data being consumed. The core must be started by asserting FD_IN with ND for this to happen.

If frame-level flow control is not required, then FD_IN can be tied to RFFD. The core automatically starts processing data when it is ready. In these circumstances, data is consumed by the core when both ND and RFD are asserted. Note that the data input to the core must be synchronized to FD_IN, with the first item in the frame being provided with FD_IN or in the next cycle that both ND and RFD are asserted.

Data Output

The number of cycles between the first item of a frame entering the core and the encoded message becoming available is described under "Latency," page 11. Once started a frame of data can be streamed out as a contiguous block. If the data rate is too high, then it can be modified by periodically deasserting CTS to stall output. If necessary, the core also deasserts RFD to stop the flow of data into the core to prevent loss of data.

End-to-End Flow Control

The inclusion of data flow handshake signals on both the core's input and output provides the means to implement end-to-end flow control, whereby the data flow can be either regulated by the source of uncoded input data or the destination of the encoded output data, as shown in Figure 3.

Throughput

Limiting Factors

The throughput of the core, which can be measured in terms of bits-per-second, depends on four limits:

- Rate of data input
- Rate at which data is processes by the core
- Rate of data output
- Output frame buffer size when frame size is changed

The maximum rate of data input and output depends upon the width of the DIN and DOUT ports and the clock rate. For example, if the width is 4 bits (Raw format, with Width = 4), then the maximum rate is 4-bits per cycle, that is, 400 Mbps at 100 MHz.

Sustained Throughput

The number of bits which the core may process per cycle when averaged over a frame is a little less than the specified throughput Width. It depends upon RATE; however, the worst case sustained throughput across all rates is summarized in Table 7 for Width 1 and 4.

Table 7: **Valid Values for the Frame Size Field of RATE**

Width	Minimum Throughput (bits per cycle)		
	Normal	Short	Both
1	0.9976	0.9906	0.9906
4	3.883	3.587	3.587

The worst case throughput has been established for Width 4 in the following way.

Table 8 and Table 9 summarize the number of cycles needed to process a frame for each RATE when Width = 4 and the data input and output formats are Raw (that is, they will not limit throughput).

The sustained throughput, in terms of bit per second, is given by:

$$Throughput = \left(\frac{n_{ldpc}}{C_P}\right) f_{max}$$

where f_{max} is the maximum clock frequency for the core, n_{ldpc} is the output frame size, and C_P is the number of cycles that it takes to process a frame at a particular rate (assuming data availability at input and no stalling of output). This is given for each rate in Table 8 for normal frames and Table 9 for short frames.

The minimum sustained throughput across all rates can be obtained by taking the maximum value of C_P across all rates. For normal frames, this is 16686 cycles, as provided in the bottom row of Table 8. This represents a throughput of 64800/16686 = 3.883 bps, or 388.3 Mbps at 100 MHz.

Likewise, for short frames, the maximum number of cycles per frame across all rates is 4516 cycles (as given in the bottom row of Table 9). This represents a throughput of 16200/4516 = 3.587 bps or 358.7 Mbps at 100 MHz.

Likewise, the sustained throughput for Width 1 can be calculated from Table 10 and Table 11.

Note that to achieve sustained throughput, it is necessary to supply input and output data at this rate. This will not be possible if the throughput Width is 4 with Symbol output format, as for a modulation type QPSK and 8PSK, only 2 bits and 3 bits of data respectively, would be output per cycle. As a consequence, the throughput would be limited to 2 and 3 bits per cycle, that is, 200 Mbps and 300 Mbps at 100 MHz. Both of which are less than the processing throughput 388.3 Mbps. To avoid this limitation, a Raw output format can be adopted, and the symbol encoding performed externally.

Output Buffer Size

If frame size is to be changed between frames then the Output Buffer Size should be set to Adaptive_Frame_Size. With Constant_Frame_Size, the buffer is smaller, but it is only possible to provide maximum sustained throughput when there are a minimum of 4 short frames between normal frames. Reducing the number of short frames to less than 4 can result in output stalling between frames. Note that the frame rate is maintained, but as the smaller frames contain less data, the overall data rate is reduced.

Latency

The latency of the core is defined as the number of cycles between input of the first bit of an uncoded input frame to the first bit of an encoded output frame.

The latency of the core consists of the number of cycles taken to write the input data into the input buffer plus the latency through the three FEC blocks.

Input Buffer Latency

The number of cycles to write the input buffer depends upon the input buffer mode. When the mode is Wait_For_Whole_Frame the number of cycles is $C_{ip} = \frac{k_{bch}}{Win}$. The width of input data, Win, depends upon the input format and the throughput Width selected for the core (and may be 1, 4, or 8 bits). For input buffer mode, Process_ASAP $C_{ip} = 1$.

Latency Through FEC Blocks

The latency through the FEC blocks is equal to $\frac{n_{ldpc}}{Width}$ plus a delay K_{rate} that is dependent on the frame size, the throughput Width employed, and whether Symbol output format is used.

Total Latency

The total latency, L, is the sum of these parts, that is:

$$L = C_{ip} + \frac{n_{ldpc}}{Width} + K_{rate}$$

Note that this latency value assumes that unencoded data is input to the core as a contiguous block.

Table 8 and Table 9 summarize the actual latency and throughput of the core for each rate for Width = 4, and Table 10 and Table 11 for Width = 1.

These tables assume that the input and output format are Raw. If the output format is Symbol and Width = 1, then K_{rate} should be increased by 15 cycles over that given in Table 10 and Table 11. If the output format is Symbol and the Width is 4, then use the values in the table unmodified.

Note that latency is not dependent upon modulation type.

Table 8: **Latency and Throughput of Core for Normal Frames and Width=4**

Rate	k_{bch}	$k_{bch}/4$	$n_{ldpc}/4$	K_{rate}(1)	Min. Cycles per Frame (C_P)	Latency (Wait_For_ Full_Frame)	Latency (Process_ ASAP)
1/4	16008	4002	16200	429	16598	20631	16630
1/3	21408	5352	16200	324	16491	21876	16525
2/5	25728	6432	16200	312	16479	22944	16513
1/2	32208	8052	16200	474	16641	24726	16675
3/5	38688	9672	16200	276	16443	26148	16477
2/3	43040	10760	16200	264	16431	27224	16465
3/4	48408	12102	16200	519	16686	28821	16720
4/5	51648	12912	16200	240	16407	29352	16441
5/6	53840	13460	16200	414	16581	30074	16615
8/9	57472	14368	16200	224	16391	30792	16425
9/10	58192	14548	16200	402	16569	31150	16603
Maximum value					16686	31150	16720

Notes:

1. K_{rate} is a constant number of cycles associated with a particular rate. See "Latency Through FEC Blocks" for further details.

Table 9: **Latency and Throughput of Core for Short Frames and Width=4**

Rate	k_{bch}	$k_{bch}/4$	$n_{ldpc}/4$	K_{rate}(1)	Min. Cycles per Frame	Latency (Wait_For_ Full_Frame)	Latency (Process_ ASAP)
1/4	3072	768	4050	240	4257	5058	4291
1/3	5232	1308	4050	414	4431	5772	4465
2/5	6312	1578	4050	321	4338	5949	4372
1/2	7032	1758	4050	499	4516	6307	4450
3/5	9552	2388	4050	402	4419	6840	4453
2/3	10632	2658	4050	309	4326	7017	4360
3/4	11712	2928	4050	216	4233	7194	4267
4/5	12432	3108	4050	394	4411	7552	4445
5/6	13152	3288	4050	212	4229	7550	4263
8/9	14232	3558	4050	479	4496	8087	4530
Maximum value					4516	8087	4530

Notes:

1. K_{rate} is a constant number of cycles associated with a particular rate. See "Latency Through FEC Blocks" for further details.

Table 10: **Latency and Throughput of Core for Normal Frames and Width=1**

Rate	k_{bch}	n_{ldpc}	K_{rate}	Min. Cycles per Frame	Latency (Wait_For_ Full_Frame)	Latency (Process_ ASAP)
1/4	16008	64800	171	64953	80979	64972
1/3	21408	64800	171	64953	86379	64972
2/5	25728	64800	171	64953	90699	64972
1/2	32208	64800	171	64953	97179	64972
3/5	38688	64800	171	64953	103659	64972
2/3	43040	64800	171	64953	108011	64972
3/4	48408	64800	171	64953	113379	64972
4/5	51648	64800	171	64953	116619	64972
5/6	53840	64800	171	64953	118811	64972
8/9	57472	64800	171	64953	122443	64972
9/10	58192	64800	171	64953	123163	64972
Maximum value				64953	123163	64972

Table 11: **Latency and Throughput of Core for Short Frames and Width=1**

Rate	k_{bch}	n_{ldpc}	K_{rate}	Min. Cycles per Frame	Latency (Wait_For_Full_Frame)	Latency (Process_ASAP)
1/4	3072	16200	171	16353	19443	16372
1/3	5232	16200	171	16353	21603	16372
2/5	6312	16200	171	16353	22683	16372
1/2	7032	16200	171	16353	23403	16372
3/5	9552	16200	171	16353	25923	16372
2/3	10632	16200	171	16353	27003	16372
3/4	11712	16200	171	16353	28083	16372
4/5	12432	16200	171	16353	28803	16372
5/6	13152	16200	171	16353	29523	16372
8/9	14232	16200	171	16353	30603	16372
Maximum value				16353	30603	16372

Using the DVB-S.2 FEC Encoder v2.0 Core

Simulation Models

The core has a number of options for simulation models:

- VHDL UniSim-based simulation model
- Verilog UniSim-based structural simulation model

The models required may be selected in the CORE Generator software project options.

Xilinx recommends that simulations utilizing UniSim-based structural models are run using a resolution of 1 ps. Some Xilinx library components require a 1 ps resolution in either functional or timing simulation. The UniSim-based structural models may produce incorrect results if simulated with a resolution other than 1 ps. See the "Register Transfer Level (RTL) Simulation Using Xilinx Libraries" section in *Chapter 6* of the Synthesis and Simulation Design Guide. This document is part of the ISE® Software Manuals set available at www.xilinx.com/support/software_manuals.htm.

Test Bench

A VHDL test bench is generated by the core. This can be used with the UniSim-based simulation model to exercise the core. The name of the test bench is the component name prefixed by "tb_". It is located with the other CORE Generator output files.

The generated test bench depends on whether the Output Buffer Size is Constant_Frame_Size or Adaptive_Frame_Size:

Test Bench Configured for Constant_Frame_Size

When the Output Buffer Size is Constant_Frame_Size, the main purpose of the test bench is to provide a simple example of the core being driven with data, and generating associated output.

The input is applied as soon as the core is able to accept data, and the output is taken as soon as it is available. The sequence of frames is predefined within get_rates() function. The sequence consists of 4 short frames between normal frames, with a Modulation Type that can be set within get_rates() function. These can be redefined, or new cases added as required.

Test Bench Configured for Adaptive_Frame_Size

When the Output Buffer Size is Adaptive_Frame_Size the main purpose of this test bench is to demonstrate how the core can be used to provide sustained data output in circumstances where the frame size changes. There are two options each for Frame Input and Frame Output:

Frame Input

Frames of random data are input as soon as possible (that is, the input is only throttled by `RFD` and `RFFD`). The frames are determined according to two scenarios:

SCENARIO = 1 (default)

This scenario exercises the core with a sequence of frames with random size, modulation type and code rate.

Note that this is only used when Output Buffer Size is set to Adaptive_Frame_Size as it may generate sequences with less than 4 short frames between normal frames, and so cause the sustained throughput check on the output, described later in this section, to fail.

SCENARIO = 0

This is the same predefined sequence of frames that is provided when the test bench is configured for Constant_Frame_Size. These can be redefined, or new cases added as required.

Frame Output

The frames are read out of the core in one of two ways (as determined by the value given to the constant SUSTAINED_THROUGHPUT):

SUSTAINED_THROUGHPUT = TRUE (default)

The output is read in a way that demonstrates that a sustained data output rate is possible when the frame size changes. It works in the following way.

At start-up, data output is delayed for the worst-case latency of a normal-sized frame in order to ensure that the availability of subsequent frames will be only be governed by the time it takes to process each frame (that is, Min Cycles per Frame as documented in Table 8, Table 9, Table 10 and Table 11). This is achieved by setting `CTS` to '0'. After the worst-case latency, the output of a frame is enabled by setting `CTS` to '1'.

Within a frame, data is read out as fast as possible (that is, `CTS` is maintained at '1'). Once all the data associated with a frame has been read, `CTS` is set to '0' to halt further output until the clock cycles-per-frame count has been reached.

Note that CTS is set to '0' when the last item of data is on DOUT. This retains the last item of data on DOUT until the start of the next frame, and ensures that RDY is maintained high when entering the next frame output period. In this way the state of RDY is known at the start of the next frame, and the timing of subsequent frames is consistent. (This overcomes the cycle of latency on the CTS control, and if this were not done, an additional cycle may/or may not be incurred transitioning to the RDY state at the beginning of the next frame, depending upon the availability of data within the output buffer).

A new frame is then started by setting CTS to '1'. This is done when the cycle-per-frame count is reached in order to start the frame on the next cycle. On the next cycle FD_OUT should be '1', if not, a failure is asserted as this indicates that the expected throughput is not being achieved.

The overall effect of this control of CTS is to provide a stream of frames out of the core at a predefined rate. Over each frame output period the data can be read out of the core as required subject to the second item in the following notes, and providing that the last item of data is read out on the last cycle of the frame period.

Note that:

- Within the test bench the number of cycles-per-frame is set differently for short and normal frames, and for each is the worst-case value across all rates across that frame size. Alternatively, the normal frame period could be set to four times the short frame period to give a constant bit rate (when averaged over a frame).
- When the Output Format is Symbol, and the symbol size is greater than the Width of the core, then readout within a frame is not contiguous (i.e. RDY is not '1' for the whole frame), but has a rate that is governed by the Width. For example, if Width were 1 bit and symbol width 3 bits, then one output every 3 cycles would be generated.
- If the Output Format is Symbol, and the symbol width is less than Width, then the number of cycles-per-frame will be governed by the symbol width (that is, Modulation Type). For example, if the Width is 4, and the symbol width is 2 bits (that is, modulation is 8PSK), then the number of cycles to read out a frame would be 64,800/2 = 32,400.

SUSTAINED_THROUGHPUT = FALSE

The output is read as soon as it becomes available (that is, CTS is permanently set to '1').

Core Resource Requirements and Performance

The resource requirements of the core and the achievable clock rates for Virtex-6 and Spartan-6 FPGAs are summarized in Table 12 and Table 13. These measurements were made for a core with Raw input and output format, input buffer mode Wait_For_Full_Frame, output buffer size Constant_Frame_Rate, and both SCLR and CE implemented.

These results were obtained with ISE® v11.4 tools. The resource count and speed of the core can change depending on the surrounding circuitry of your design. Therefore, these figures should be taken only as a guide.

The maximum clock frequency results were obtained by double-registering input and output ports (using IOB flip-flops) to reduce dependence on I/O placement. The first level of registers used a separate clock signal to measure the path from the input registers to the first output wrapper register through the core.

The resource usage results do not include the aforementioned wrapper registers and represent the true logic used by the core. LUT counts include SRL16s or SRL32s (according to device family).

The tool settings to achieve these results were as follows:

```
map -ol high
par -ol high
```

Note: The tool settings can have a significant effect on area use and speed. The Xilinx Xplorer script can be used to find the optimal settings.

The Virtex-6 FPGA test cases in Table 12 used an XC6VLX75T-FF484 (-1 speed grade) device and ISE software speed file version "ADVANCED 1.02b 2009-10-13."

Table 12: **Virtex-6 FPGA Family Performance and Resource Utilization**

Width	LUT/ FF Pairs	LUTs	FFs	Block RAMs (36k/18k)	DSP48E1s	Max. Clock Frequency (MHz)	Maximum Sustained Encoded Output Rate (Mbps)
1	1661	1623	1434	9/7	2	214	214
4	2459	2419	2039	10/7	2	213	764

Notes:

1. Area and maximum clock frequencies are provided as a guide. They may vary with new releases of Xilinx implementation tools, etc.
2. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

The Spartan-6 FPGA test cases in Table 13 used an XC6SLX45-FGG484 (-2 speed grade) device and ISE software speed file version "ADVANCED 1.02d 2009-10-13".

Table 13: **Spartan-6 FPGA Family Performance and Resource Utilization**

Width	LUT/ FF Pairs	LUTs	FFs	Block RAMs (18k/9k)	DSP48A1s	Max. Clock Frequency (MHz)	Maximum Sustained Encoded Output Rate (Mbps)
1	1729	1678	1438	23/2	2	135	135
4	2503	2455	2038	25/2	2	125	448

Notes:

1. Area and maximum clock frequencies are provided as a guide. They may vary with new releases of Xilinx implementation tools, etc.
2. Clock frequency does not take jitter into account and should be de-rated by an amount appropriate to the clock source jitter specification.

References

1. ETSI, DVB S.2 standard, ETSI EN 302 307 V1.1.2, 2006-06.

Support

Xilinx provides technical support for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

See the *IP Release Notes Guide* XTP025 for further information on this core. On the first page there will be a link to "All DSP IP." The relevant core can then be selected from the displayed list.

For each core, there is a link to the master Answer Record that contains the Release Notes and Known Issues list. The following information is listed for each version of the core:

- New Features
- Bug Fixes
- Known Issues

Ordering Information

The DVB-S.2 FEC Encoder core is provided under the SignOnce IP Site License and can be generated using the Xilinx® CORE Generator software v11.4 or higher. The CORE Generator software is shipped with the Xilinx ISE Design Suite.

To access the full functionality of the core, including simulation and FPGA bitstream generation, a full license must be obtained from Xilinx. For more information, visit the DVB S-2 FEC Encoder product page.

Contact your local Xilinx sales representative for pricing and availability of additional Xilinx® LogiCORE IP modules and software. Information about additional Xilinx® LogiCORE IP modules is available on the Xilinx IP Center.

Revision History

Date	Version	Revision
04/28/05	1.0	Xilinx initial release.
06/20/05	1.1	Modified for version 1.1
01/18/06	1.2	Updated for IP1I release.
05/17/07	1.3	Updated for version 1.3 and to support Spartan-3A DSP FPGAs.
06/27/08	1.4	Updated for core version 1.4.
12/02/09	2.0	Updated for version 2.0 and to support Spartan-6 and Virtex-6 FPGAs.

Notice of Disclaimer