4G Turbo-V Intel® FPGA IP User Guide

Updated for Quartus® Prime Design Suite: **24.1**

IP Version: **2.0.4**

Online Version
Send Feedback

UG-20279

683882
2024.04.01

Contents

1. About the 4G Turbo-V Intel® FPGA IP

Forward-error correction (FEC) channel codes commonly improve the energy efficiency of wireless communication systems. Turbo codes are suitable for 3G and 4G mobile communications (e.g., in UMTS and LTE) and satellite communications. You can use Turbo codes in other applications that require reliable information transfer over bandwidth- or latency-constrained communication links in the presence of data-corrupting noise. The 4G Turbo-V Intel® FPGA IP comprises a downlink and uplink accelerator for vRAN and includes the Turbo Intel FPGA IP. The downlink accelerator adds redundancy to the data in the form of parity information.The uplink accelerator exploits redundancy to correct a reasonable number of channel errors.

Related Information

- Turbo Intel FPGA IP User Guide
- 3GPP TS 36.212 version 15.2.1 Release 15

1.1. 4G Turbo-V Intel FPGA IP Features

- 3GPP LTE compliant with support for block sizes 40 to 6,144
- C and MATLAB bit-accurate models.

The downlink accelerator includes:

- Code block cyclic redundancy code (CRC) attachment
- Turbo encoder
- Rate matcher with:
  - Subblock interleaver
  - Bit collector
  - Bit selector
  - Bit pruner

The uplink accelerator includes:

- Subblock deinterleaver
- Turbo decoder with CRC check

1.2. 4G Turbo-V Intel FPGA IP Device Family Support

The IP supports the following devices families:

- Agilex™ 5 devices
- Agilex 7devices
- Arria® 10 devices
- Stratix® 10 devices

For the device support levels for Intel FPGA IP, refer to *Device Support and Pin-Out Status* in the latest version of the *Intel Quartus Prime Pro Edition: Software and Device Support Release Notes*.

Related Information

Device Support and Pin-Out Status

1.3. 4G Turbo-V IP Release Information

Intel FPGA IP versions match the Quartus® Prime Design Suite software versions until v19.1. Starting in Quartus Prime Design Suite software version 19.2, Intel FPGA IP has a new versioning scheme.

The Intel FPGA IP version (X.Y.Z) number can change with each Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Quartus Prime software, you must regenerate the IP.
- Y indicates the IP includes new features. Regenerate your IP to include these new features.
- Z indicates the IP includes minor changes. Regenerate your IP to include these changes.

Table 1. 4G Turbo-V IP Release Information

Item	Description
Version	2.0.3
Release date	August 2023
Ordering code	IP-TURBOV

1.4. 4G Turbo-V Performance and Resource Utilization

Intel generated the resource utilization and performance by compiling the designs with Intel Quartus Prime software v23.2. Only use these approximate results for early estimation of FPGA resources (e.g. adaptive logic modules (ALMs)) that a project requires.

Table 2. Downlink Accelerator Performance and Resources

Family	Device	Average f_{MAX} (MHz) [1]	ALM	M20K	DSP
Agilex 5	SM7_PART_4S	248	8.1k	51	6
	A5EC065BB32AE5SR0	204	8.2k	51	6
	A5EC065BB32AE6SR0	189	8.2k	51	6
Agilex 7	AGFB014R24B1E1V	430	8.7k	51	6
	AGFB014R24B2E2V	382	8.7k	51	6
	AGFB014R24B2E3V	328	8.7k	51	6
	AGFB014R24B2E3E	328	8.7k	51	6
	AGFB014R24B2E4X	243	8.7k	51	6
	AGFB014R24B2E4F	243	8.7k	51	6
	AGFB014R24B1I1V	430	8.7k	51	6
	AGFB014R24B2I2V	382	8.7k	51	6
	AGFB014R24B2I3V	328	8.7k	51	6
	AGFB014R24B2I3E	328	8.7k	51	6
Intel Arria 10	10AT115S1F45E1SG	288	7.7k	51	6
Intel Stratix 10	1SG280HU2F50E2VG	275	8.4k	51	6
	1SG280HU2F50E2LG	263	8.4k	51	6

Table 3. Uplink Accelerator Performance and Resource Utilization

Family	Device	Average f_{MAX} (MHz) [2]	ALM	M20K	DSP
Agilex 5	SM7_PART_4S	302	26.6k	45	0
	A5EC065BB32AE5SR0	252	26.3k	45	0
	A5EC065BB32AE6SR0	231	26.3k	45	0
Agilex 7	AGFB014R24B1E1V	519	28.8k	45	0
	AGFB014R24B2E2V	460	28.8k	45	0
	AGFB014R24B2E3V	414	28.8k	45	0
	AGFB014R24B2E3E	414	28.8k	45	0
	AGFB014R24B2E4X	362	28.8k	45	0
	AGFB014R24B2E4F	324	28.8k	45	0
	AGFB014R24B1I1V	515	28.8k	45	0
	AGFB014R24B2I2V	461	28.8k	45	0
	AGFB014R24B2I3V	417	28.8k	45	0
	AGFB014R24B2I3E	417	28.8k	45	0

continued...

(1) Reduced 15% for margin

(2) Reduced 15% for margin

Family	Device	Average f_{MAX} (MHz) [2]	ALM	M20K	DSP
Intel Arria 10	10AT115S1F45E1SG	304	23.8k	45	0
Intel Stratix 10	1SG280HU2F50E2VG	304	28.1k	45	0
	1SG280HU2F50E2LG	283	28.0k	45	0

(2) Reduced 15% for margin

2. Getting Started with 4G Turbo-V IP

2.1. Installing and Licensing Intel FPGA IP Cores

The Quartus Prime software installation includes the Intel FPGA IP library. This library provides many useful IP cores for your production use without the need for an additional license. Some Intel FPGA IP cores require purchase of a separate license for production use. The Intel FPGA IP Evaluation Mode allows you to evaluate these licensed Intel FPGA IP cores in simulation and hardware, before deciding to purchase a full production IP core license. You only need to purchase a full production license for licensed ® IP cores after you complete hardware testing and are ready to use the IP in production.

The Quartus Prime software installs IP cores in the following locations by default:

Figure 1. IP Core Installation Path

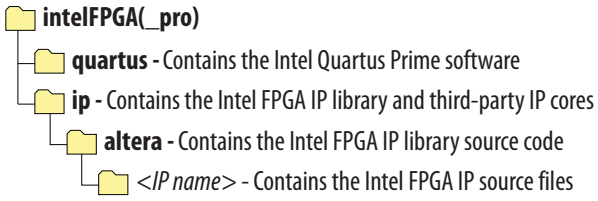


Table 4. IP Core Installation Locations

Location	Software	Platform
<drive>:\intelFPGA_pro\quartus\ip\altera	Quartus Prime Pro Edition	Windows*
<drive>:\intelFPGA\quartus\ip\altera	Quartus Prime Standard Edition	Windows
<home directory>:/intelFPGA_pro/quartus/ip/altera	Quartus Prime Pro Edition	Linux*
<home directory>:/intelFPGA/quartus/ip/altera	Quartus Prime Standard Edition	Linux

Note: The Quartus Prime software does not support spaces in the installation path.

2.1.1. Intel FPGA IP Evaluation Mode

The free Intel FPGA IP Evaluation Mode allows you to evaluate licensed Intel FPGA IP cores in simulation and hardware before purchase. Intel FPGA IP Evaluation Mode supports the following evaluations without additional license:

- Simulate the behavior of a licensed Intel FPGA IP core in your system.
- Verify the functionality, size, and speed of the IP core quickly and easily.
- Generate time-limited device programming files for designs that include IP cores.
- Program a device with your IP core and verify your design in hardware.

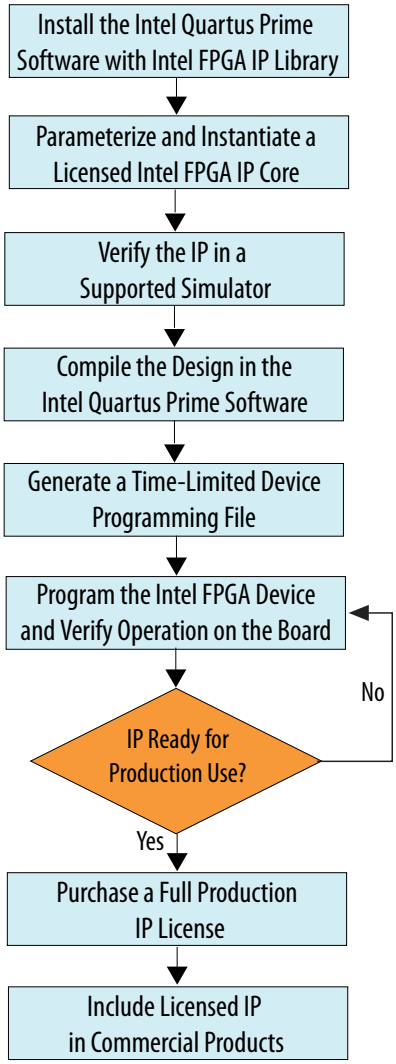
Intel FPGA IP Evaluation Mode supports the following operation modes:

- **Tethered**—Allows running the design containing the licensed Intel FPGA IP indefinitely with a connection between your board and the host computer. Tethered mode requires a serial joint test action group (JTAG) cable connected between the JTAG port on your board and the host computer, which is running the Quartus Prime Programmer for the duration of the hardware evaluation period. The Programmer only requires a minimum installation of the Quartus Prime software, and requires no Quartus Prime license. The host computer controls the evaluation time by sending a periodic signal to the device via the JTAG port. If all licensed IP cores in the design support tethered mode, the evaluation time runs until any IP core evaluation expires. If all of the IP cores support unlimited evaluation time, the device does not time-out.
- **Untethered**—Allows running the design containing the licensed IP for a limited time. The IP core reverts to untethered mode if the device disconnects from the host computer running the Quartus Prime software. The IP core also reverts to untethered mode if any other licensed IP core in the design does not support tethered mode.

When the evaluation time expires for any licensed Intel FPGA IP in the design, the design stops functioning. All IP cores that use the Intel FPGA IP Evaluation Mode time out simultaneously when any IP core in the design times out. When the evaluation time expires, you must reprogram the FPGA device before continuing hardware verification. To extend use of the IP core for production, purchase a full production license for the IP core.

You must purchase the license and generate a full production license key before you can generate an unrestricted device programming file. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (<*project name*>`_time_limited.sof`) that expires at the time limit.

Figure 2. Intel FPGA IP Evaluation Mode Flow



Note: Refer to each IP core's user guide for parameterization steps and implementation details.

licenses IP cores on a per-seat, perpetual basis. The license fee includes first-year maintenance and support. You must renew the maintenance contract to receive updates, bug fixes, and technical support beyond the first year. You must purchase a full production license for Intel FPGA IP cores that require a production license, before generating programming files that you may use for an unlimited time. During Intel FPGA IP Evaluation Mode, the Compiler only generates a time-limited device programming file (`<project name>_time_limited.sof`) that expires at the time limit. To obtain your production license keys, visit the FPGA Self-Service Licensing Center.

The FPGA Software License Agreements govern the installation and use of licensed IP cores, the Quartus Prime design software, and all unlicensed IP cores.

Related Information

- FPGA Licensing Support Center
- Introduction to FPGA Software Installation and Licensing

2.1.2. 4G Turbo-V IP Timeout Behavior

All IP cores in a device time out simultaneously when the most restrictive evaluation time is reached. If a design has more than one IP core, the time-out behavior of the other IP cores may mask the time-out behavior of a specific IP core.

For IP cores, the untethered time-out is 1 hour; the tethered time-out value is indefinite. Your design stops working after the hardware evaluation time expires. The Quartus Prime software uses Intel FPGA IP Evaluation Mode Files (`.ocp`) in your project directory to identify your use of the Intel FPGA IP Evaluation Mode evaluation program. After you activate the feature, do not delete these files..When the evaluation time expires, the data output port `reset_n` goes low, which keeps the IP core permanently in its reset state.

Related Information

AN 320: Using the Intel FPGA IP Evaluation Mode

3. Designing with the 4G Turbo-V Intel FPGA IP

3.1. Generating a 4G Turbo-V IP

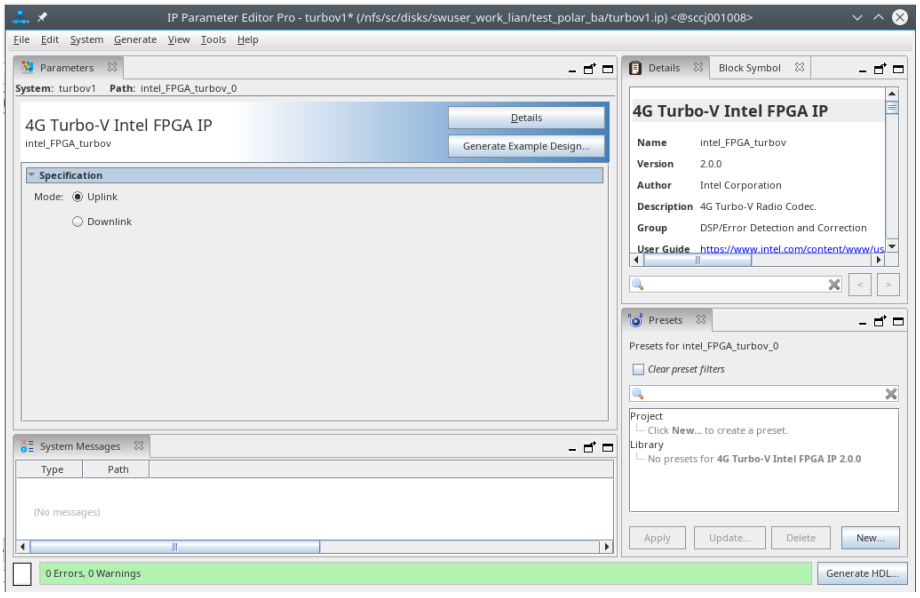
You can generate a downlink or uplink accelerator. To include the IP in a design, generate the IP in the Quartus Prime software. Or optionally, you can generate a design example that includes the generated 4G Turbo-V IP, a C model, a MATLAB model, and simulation scripts. The software generates no hardware example in **Generate Example Design**.

1. Create a New Quartus Prime project
2. Open IP Catalog.
3. Select **DSP ➤ Error Detection and Correction ➤ 4G Turbo-V** and click **Add**
4. Enter a name for your IP variant and click **Create**.

   The name is for both the top-level RTL module and the corresponding `.ip` file.

   The parameter editor for this IP appears.
5. Choose **Uplink** or **Downlink**.

Figure 3. 4G Turbo-V Parameters



6. For an optional design example, click **Generate Example Design**

   No hardware example gets generated when you click **Generate Example Design**. If you upgrade the IP to a newer version, regenerate the example design.

The software creates a design example and files that you can use for MATLAB, C, or RTL simulations.

7. Click **Generate HDL**.

Quartus Prime generates the RTL and the files necessary to instantiate the IP in your design and synthesize it.

Related Information

Generating IP Cores
Use this link for the Quartus Prime Pro Edition Software.

3.2. Simulating the IP with the RTL Simulator

Before simulating, generate a design example from the IP parameter editor. No hardware example gets generated when you click **Generate Example Design**. If you upgrade the IP to a newer version, regenerate the example design.

1. To run the simulation with Synopsys VCS® simulator, run `vcsmx_setup.sh` from `<example_design_directory>\simulation_scripts\synopsys\vcsmx\` directory by typing the following commands:

```
>> source vcsmx_setup.sh
>> simv
```

2. To run the simulation with Cadence NCSim® simulator, run `ncsim_setup.sh` from `<example_design_directory>\simulation_scripts\cadence\` directory by typing the following command:

```
sh ./ncsim_setup.sh USER_DEFINED_ELAB_OPTIONS='"-timescale 1ps/1ps"'
USER_DEFINED_SIM_OPTIONS='"-input \"@run; exit\""'
```

3. To run the simulation with Xcelium® simulator, run `xcelium_setup.sh` from `<example_design_directory>\simulation_scripts\xcelium\` directory by typing the following command:

```
sh ./xcelium_setup.sh USER_DEFINED_ELAB_OPTIONS='"-timescale 1ps/1ps"'
USER_DEFINED_SIM_OPTIONS='"-input \"@run; exit\""'
```

4. To run the simulation with the ModelSim or Questa® simulator, run `msim_setup.tcl` from `<example_design_directory>\simulation_scripts\mentor\` directory by typing the following commands:

```
vsim -c do msim_setup.tcl
ld
run -all
```

5. To run the simulation with Aldec® simulator. run `rivierapro_setup.tcl` from `<example_design_directory>\simulation_scripts\aldec\` directory by typing the following commands:

```
vsim -c do rivierapro_setup.tcl
ld
run -all
```

3.3. Simulating a 4G Turbo-V IP with MATLAB

Before simulating, generate a design example from the IP parameter editor. No hardware example gets generated when you click **Generate Example Design**. If you upgrade the IP to a newer version, regenerate the example design. This task is for simulating an uplink accelerator. To simulate an downlink accelerator replace `ul` with `dl` in each directory or file name.

1. Run the MATLAB script from the <*Example Design Folder*>`\matlab\directory` in MATLAB.

   ```
   >> main_turbov_ul
   ```

   The MATLAB model reads and generates the same `.txt` files as the C model.

You can use the MATLAB `generate_test_data.m` to generate different test data. Newly generated `.txt` files replace the existing ones in the <*Example Design Folder*>`\test_data` directory.

3.4. Simulating a 4G Turbo-V IP with C-Models

Before simulating, generate a design example from the IP parameter editor. No hardware example gets generated when you click **Generate Example Design**. If you upgrade the IP to a newer version, regenerate the example design. This task is for simulating an uplink accelerator. To simulate an downlink accelerator replace `ul` with `dl` in each directory or file name. You compile and execute the C code from <*Example Design Folder*>`\c\` directory.

1. For GCC compiler from terminal on Linux, type:

   ```
   >> gcc -lm main_turbov_ul.c -o run_ul
   >> ./run_dl
   ```

2. For GCC compiler (e.g. MinGW-w64) from a command prompt on Windows, type:

   ```
   >> gcc -lm main_turbov_ul.c -o run_ul
   >> ./run_ul.exe
   ```

3. For Visual Studio on Windows:
   a. Create an empty VS project in <*Example Design Folder*> directory
   b. Add the single source file `main_turbov_ul.c` to the project
   c. Build and run the project

- The executable reads `../test_data/turbov_ul_input_data.txt` and `../test_data/turbov_ul_input_info.txt` as inputs.
- The executable generates `../test_data/turbov_ul_output_data_gold.txt` and `../test_data/turbov_ul_output_info_gold.txt` as outputs.
- You can use the same input `.txt` files in RTL simulations.
- The output `.txt` file provides a golden output, which you can use to check the correctness of the output from RTL simulations.
- You can create a VS project at a different location. You must move the `test_data` folder to <*vs project*>`/../test_data` directory to avoid the error `cannot open ../test_data/turbov_ul_input_data.txt`.

4. 4G Turbo-V Intel FPGA IP Functional Description

The 4G Turbo-V Intel FPGA IP comprises a downlink accelerator and an uplink accelerator.

Table 5. LTE Block K Size

The table shows codeblock size K valid values for LTE codeblock index idx

idx	K	idx	K	idx	K	idx	K
1	40	48	416	95	1120	142	3200
2	48	49	424	96	1152	143	3264
3	56	50	432	97	1184	144	3328
4	64	51	440	98	1216	145	3392
5	72	52	448	99	1248	146	3456
6	80	53	456	100	1280	147	3520
7	88	54	464	101	1312	148	3584
8	96	55	472	102	1344	149	3648
9	104	56	480	103	1376	150	3712
10	112	57	488	104	1408	151	3776
11	120	58	496	105	1440	152	3840
12	128	59	504	106	1472	153	3904
13	136	60	512	107	1504	154	3968
14	144	61	528	108	1536	155	4032
15	152	62	544	109	1568	156	4096
16	160	63	560	110	1600	157	4160
17	168	64	576	111	1632	158	4224
18	176	65	592	112	1664	159	4288
19	184	66	608	113	1696	160	4352
20	192	67	624	114	1728	161	4416
21	200	68	640	115	1760	162	4480
22	208	69	656	116	1792	163	4544
23	216	70	672	117	1824	164	4608
24	224	71	688	118	1856	165	4672
25	232	72	704	119	1888	166	4736
26	240	73	720	120	1920	167	4800

continued...

idx	K	idx	K	idx	K	idx	K
27	248	74	736	121	1952	168	4864
28	256	75	752	122	1984	169	4928
29	264	76	768	123	2016	170	4992
30	272	77	784	124	2048	171	5056
31	280	78	800	125	2112	172	5120
32	288	79	816	126	2176	173	5184
33	296	80	832	127	2240	174	5248
34	304	81	848	128	2304	175	5312
35	312	82	864	129	2368	176	5376
36	320	83	880	130	2432	177	5440
37	328	84	896	131	2496	178	5504
38	336	85	912	132	2560	179	5568
39	344	86	928	133	2624	180	5632
40	352	87	944	134	2688	181	5696
41	360	88	960	135	2752	182	5760
42	368	89	976	136	2816	183	5824
43	376	90	992	137	2880	184	5888
44	384	91	1008	138	2944	185	5952
45	392	92	1024	139	3008	186	6016
46	400	93	1056	140	3072	187	6080
47	408	94	1088	141	3136	188	6144

Table 6. Definition

The 4G Turbo-V IP uses these definitions.

Abbreviation	Description
E	Rate-matcher output block size
Nenc	Number of parallel encoder engines
Ndec	Number of parallel decoder engines
WLLR	Bit width of input LLR
Wout	Bit width of output data
I	Number of full iterations, where each iteration consists of two half iterations.

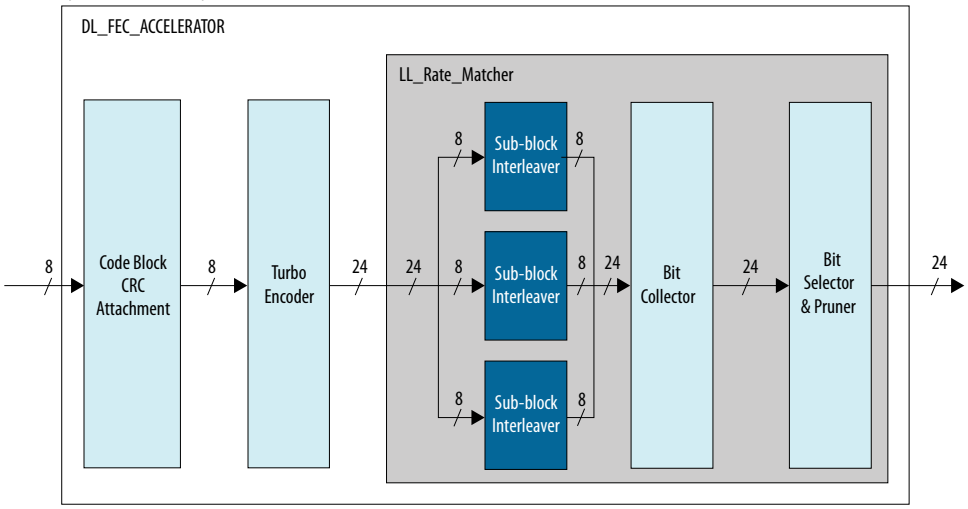
4G Turbo-V Downlink Accelerator on page 17
4G Turbo-V Uplink Accelerator on page 22
4G Turbo-V Signals and Interfaces on page 25

4.1. 4G Turbo-V Downlink Accelerator

The downlink accelerator comprises a code block CRC attachment block and a Turbo encoder (Intel Turbo FPGA IP) and rate matcher.

Figure 4. 4G Downlink Accelerator

The input data is 8-bit wide and the output data is 24-bit wide. The rate matcher consists of three subblock interleavers, a bit selector, and a bit collector.



The 4G downlink accelerator implements a code block CRC attachment with 8-bit parallel CRC computation algorithm. The input to the CRC attachment block is 8-bit wide. In the normal mode, the number of inputs to the CRC block is K-24, where K is the block size based on the size index. The additional CRC sequence of 24 bits is attached to the incoming code block of data in the CRC attachment block and then passes to the Turbo encoder. In the CRC bypass mode, the number of inputs is K size of 8-bit wide passed to the Turbo encoder block.

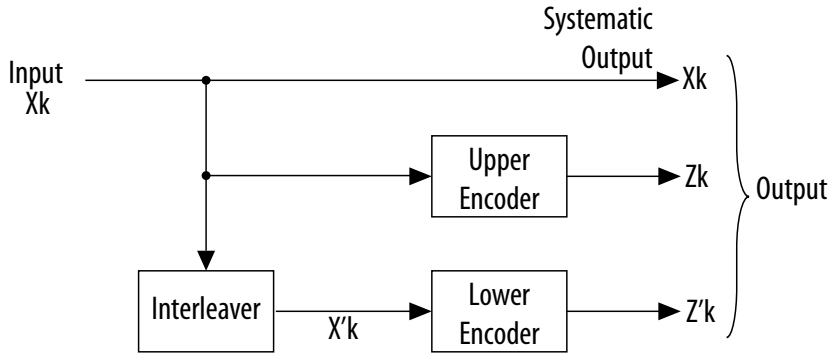
Related Information

Turbo IP Core User Guide

4.1.1. Turbo Encoder

The 3GPP Turbo encoder uses a parallel concatenated convolutional code. A convolutional encoder encodes an information sequence and another convolutional encoder encodes an interleaved version of the information sequence. The Turbo encoder has two 8-state constituent encoders and one Turbo code internal interleaver. The Turbo encoder accepts K bits and outputs 3K+12 bits, having a natural code rate 1/3. The last 12 output bits of every packet are termination bits, which guarantee that the state of the encoder is back to state zero in the end of encoding process.

Figure 5. Turbo Encoder Block Diagram



The output from the turbo coder is:

$X_0, Z_0, Z'_0, X_1, Z_1, Z'_1, \ldots, X_{K-1}, Z_{K-1}, Z'_{K-1}$

Where:

- Bits $X_0, X_1, \ldots, X_{K-1}$ are input to both the first 8-state constituent encoder and the internal interleaver (K is the number of bits).
- Bits $Z_0, Z_1, \ldots, Z_{K-1}$ and $Z'_0, Z'_1, \ldots, Z'_{K-1}$ are output from the first and second 8-state constituent encoders.
- The bits output from the internal interleaver (and input to the second 8-state constituent encoder) are $X'_0, X'_1, \ldots, X'_{K-1}$.
- Additionally, encoder outputs 12 termination bits, X_K , X_{K+1}, X_{K+2}, X'_K, X'_{K+1}, X'_{K+2}, Z_K, Z_{K+1}, Z_{K+2}, Z'_K, Z'_{K+1}, Z'_{K+2}.

4.1.2. Rate Matcher

The rate matcher matches the number of bits in transport block to the number of bits that the IP transmits in that allocation. The input and output of the rate matcher is 24 bits wide. The IP defines the rate matching for Turbo coded transport channels for each code block. The rate matcher comprises: subblock interleaver, bit collector and bit selector.

The rate matcher sets up the subblock interleaver for each output stream from Turbo encoder. The streams include a message bit stream $d_k^{(0)}$, 1st parity bit stream $d_k^{(1)}$ and 2nd parity bit stream $d_k^{(2)}$. Where

- $d_k^{(0)} = X_k$
- $d_k^{(1)} = Z_k$
- $d_k^{(2)} = Z'_k$

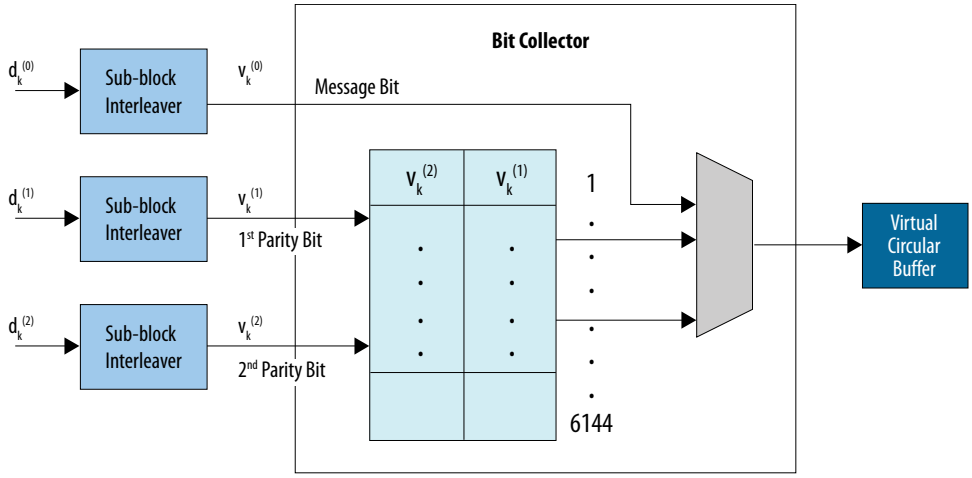
for k= 0,1,2,…,K-1, and

- $d_K^{(0)} = X_K$, $d_{K+1}^{(0)} = Z_{K+1}$, $d_{K+2}^{(0)} = X'_K$, $d_{K+3}^{(0)} = Z'_{K+1}$
- $d_K^{(1)} = Z_K$, $d_{K+1}^{(1)} = X_{K+2}$, $d_{K+2}^{(1)} = Z'_K$, $d_{K+3}^{(1)} = X'_{K+2}$
- $d_K^{(2)} = X_{K+1}$, $d_{K+1}^{(2)} = Z_{K+2}$, $d_{K+2}^{(2)} = X'_{K+1}$, $d_{K+3}^{(2)} = Z'_{K+2}$

for the termination bits, when k = K, K+1 and K+2.

The input and output of the subblock interleaver is 24 bits wide. The bit collector combines the streams that come from the subblock interleaver and stores the output in a circular buffer. Bit selector selects a total of E bits from the circular buffer and outputs E bits with 24 bits per cycle.

Figure 6. Rate Matcher Block Diagram



4.1.3. Input and Output Data Formats

Downlink accelerator data formats.

Table 7. Input Data Format

The downlink accelerator takes 8 bits per clock cycle for a total of K-24 bits, when crc_enable = 1.

Input Order	sink_data
	7…0
0	c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0
1	c_{15} c_{14} c_{13} c_{12} c_{11} c_{10} c_9 c_8
…	…
K/8-4	c_{K-25} c_{K-26} c_{K-27} c_{K-28} c_{K-29} c_{K-30} c_{K-31} c_{K-32}

Table 8. Input Data Format

The downlink accelerator takes 8 bits per clock cycle for a total of K bits, when crc_enable = 0.

Input Order	sink_data
	7…0
0	c_7 c_6 c_5 c_4 c_3 c_2 c_1 c_0
1	c_{15} c_{14} c_{13} c_{12} c_{11} c_{10} c_9 c_8
…	…
K/8-1	c_{K-1} c_{K-2} c_{K-3} c_{K-4} c_{K-5} c_{K-6} c_{K-7} c_{K-8}

Table 9. Output Data Formats

The downlink accelerator outputs 24 bits per clock cycle for a total of E bits.

Output Order	source_data
	23…0
0	e_{23} e_{22} e_{21} … e_2 e_1 e_0
1	e_{47} e_{46} e_{45} … e_{26} e_{25} e_{24}
…	…
E/24-1	e_{E-1} e_{E-2} e_{E-3} … e_{E-22} e_{E-23} e_{E-24}

Table 10. Output Data Formats

The downlink accelerator outputs 24 bits per clock cycle for a total of 3K+12 bits, when rate matcher is bypassed.

Output Order	source_data		
	23…16	15…8	7…0
0	Z'_7 Z'_6 Z'_5 Z'_4 Z'_3 Z'_2 Z'_1 Z'_0	Z_7 Z_6 Z_5 Z_4 Z_3 Z_2 Z_1 Z_0	X_7 X_6 X_5 X_4 X_3 X_2 X_1 X_0
1	Z'_{15} Z'_{14} Z'_{13} Z'_{12} Z'_{11} Z'_{10} Z'_9 Z'_8	Z_{15} Z_{14} Z_{13} Z_{12} Z_{11} Z_{10} Z_9 Z_8	X_{15} X_{14} X_{13} X_{12} X_{11} X_{10} X_9 X_8
…	…	…	…
K/8-1	Z'_{K-1} Z'_{K-2} Z'_{K-3} Z'_{K-4} Z'_{K-5} Z'_{K-6} Z'_{K-7} Z'_{K-8}	Z_{K-1} Z_{K-2} Z_{K-3} Z_{K-4} Z_{K-5} Z_{K-6} Z_{K-7} Z_{K-8}	X_{K-1} X_{K-2} X_{K-3} X_{K-4} X_{K-5} X_{K-6} X_{K-7} X_{K-8}
K/8	4'bx Z'_{K+2} X'_{K+1} Z_{K+2} X_{K+1}	4'bx X'_{K+2} Z'_K X_{K+2} Z_K	4'bx Z'_{K+1} X'_K Z_{K+1} X_K

The outputs of the last clock cycle are corresponding to 12 termination bits from turbo encoder, where output bits 23, 22, 21, 20, 15, 14, 13, 12, 7, 6, 5, 4 in the last clock cycle are unused (don't care) bits.

4.1.4. Latency Calculations

The overall latency of the downlink accelerator is the sum of the latency of the subcomponents, CRC, encoder, subblock-interleaver, pruning, and the output latency L(output). The calculations do not include control and data overhead delay, which may vary from block to block. Therefore, the calculated D is the shortest processing delay.

L(crc) = K/8+8

L(encoder) = K/8+14

L(subblock-interleaver) = Kп/8+42

L(pruning) = Kп/8+14

L(output) = ceil(E/24)+46

Example 1: K = 6144, Kп = 6176, E = 18444

L(crc) = 6144/8+8 = 776 cycles

L(encoder) = 6144/8+14 = 782 cycles

L(subblock-interleaver) = 6176/8+42 = 814 cycles

L(pruning) = 6176/8+14 = 786 cycles

L(output) = ceil(18444/24)+46 = 815 cycles

L(downlink) = 776 + 782 + 814 + 786 + 815 = 3973 cycles

Example 2: K = 40, Kп = 64, E = 132

L(crc) = 40/8+8 = 13 cycles

L(encoder) = 40/8+14 = 19 cycles

L(subblock-interleaver) = 64/8+42 = 50 cycles

L(pruning) = 64/8+14 = 22 cycles

L(output) = ceil(132/24)+46 = 52 cycles

L(downlink) = 13 + 19 + 50 + 22 + 52 = 156 cycles

4.1.5. Throughput Calculations

The downlink accelerator has buffers between two neighboring subcomponents, so that all sub-components can process in parallel at the same time.

The overall throughput can be calculated as

Throughput= K/max(L(crc), L(encoder), L(subblock-interleaver), L(pruning))*freq. (bits per second).

Example 1: K = 6144, Kп = 6176, E = 18444, freq. = 300 MHz

L(crc) = 776 cycles

L(encoder) = 782 cycles

L(subblock-interleaver) = 814 cycles

L(pruning) = 786 cycles

L(output) = 815 cycles

Throughput(downlink) = 6144/815*300M = 2262 Mbps

Example 2: K = 40, Kп= 64, E = 132, freq. = 300 MHz

L(crc) = 13 cycles

L(encoder) = 19 cycles

L(subblock-interleaver) = 50 cycles

L(pruning) = 22 cycles

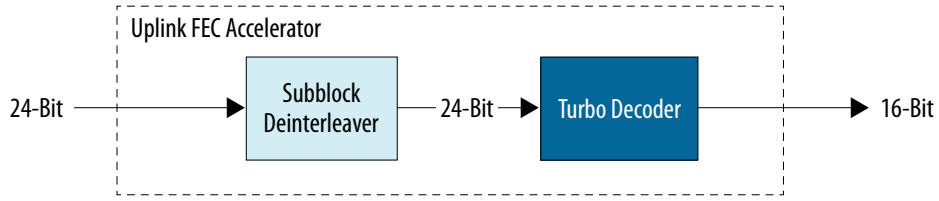
L(output) = 52 cycles

Throughput(downlink) = 40/52*300M = 231 Mbps

4.2. 4G Turbo-V Uplink Accelerator

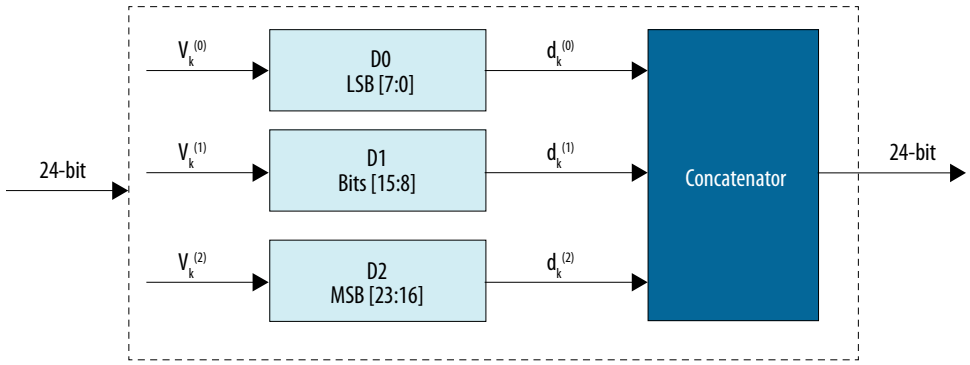
The uplink accelerator comprises a subblock deinterleaver and a turbo decoder (Intel Turbo FPGA IP)

Figure 7. 4G Channel Uplink Accelerator



The deinterleaver consists of three blocks in which the first two blocks are symmetrical and the third block is different.

Figure 8. Deinterleaver



If you turn on the bypass mode for the subblock deinterleaver, the IP reads the data as it writes the data in the memory blocks in the successive locations. The IP reads the data as and when it writes the data without any interleaving. The number of input data into the subblock deinterleaver is K_п in the bypass mode and the output data length is k size (k is the code block size based on the cb_size_index value).

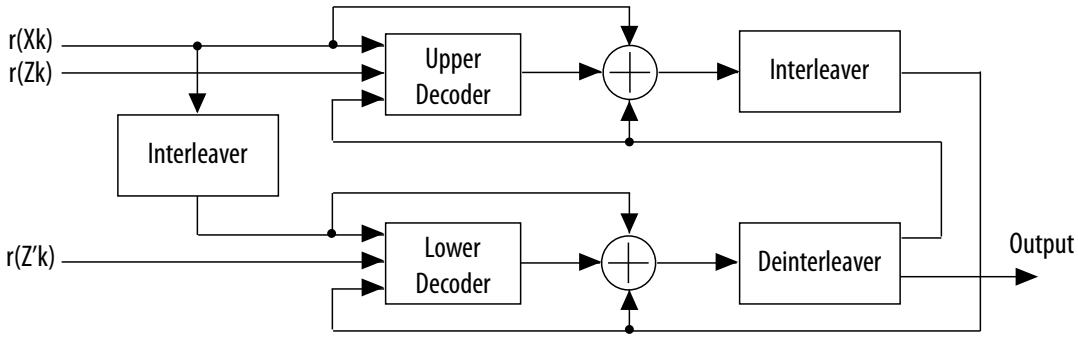
Related Information

Turbo IP Core User Guide

4.2.1. Turbo Decoder

The Turbo decoder consists of two single soft-in soft-out (SISO) decoders, which work iteratively. The output of the first (upper decoder) feeds into the second to form a Turbo decoding iteration. Interleaver and deinterleaver blocks re-order data in this process.

Figure 9. Turbo Decoder Block Diagram



The Turbo decoder supports the MaxLogMAP decoding algorithm. This algorithm is a simplified version of LogMAP that uses less logic resource and offers slightly reduced BER performance relative to LogMAP.

4.2.2. Input and Output Data Formats

Uplink accelerator data formats.

Table 11. Input Data Format

The uplink accelerator takes Kп clock cycles and 3 LLRs per clock cycle, where input LLR bitwidth (W_{LLR}) = 8.

Input Order	sink_data		
	$3W_{LLR}$-1 down to $2W_{LLR}$	$2W_{LLR}$-1 down to W_{LLR}	W_{LLR}-1 down to 0
0	$V_0^{(2)}$	$V_0^{(1)}$	$V_0^{(0)}$
1	$V_1^{(2)}$	$V_1^{(1)}$	$V_1^{(0)}$
2	$V_2^{(2)}$	$V_2^{(1)}$	$V_2^{(0)}$
…	…	…	…
Kп-1	$V_{Kп-1}^{(2)}$	$V_{Kп-1}^{(1)}$	$V_{Kп-1}^{(0)}$

Table 12. Input Data Format

In deinterleaver bypass mode, the uplink accelerator takes K+4 clock cycles and 3 LLRs per clock cycle, where input LLR bitwidth (W_{LLR}) = 8.

Input Order	sink_data		
	$3W_{LLR}$-1 down to $2W_{LLR}$	$2W_{LLR}$-1 down to W_{LLR}	W_{LLR}-1 down to 0
0	Z'_0	Z_0	X_0
1	Z'_1	Z_1	X_1
…	…	…	…
K-1	Z'_{K-1}	Z_{K-1}	X_{K-1}
K	X_{K+1}	Z_K	X_K
K+1	Z_{K+2}	X_{K+2}	Z_{K+1}
K+2	X'_{K+1}	Z'_K	X'_K
K+3	Z'_{K+2}	X'_{K+2}	Z'_{K+1}

Table 13. Output Data Format

The uplink accelerator outputs W_{out}=16 bits per clock cycle and the output block size is K

Output Order	source_data
	W_{out} down to 0
0	$X_{W_{out}-1}$, …,X_2, X_1,X_0
1	$X_{2W_{out}-1}$, …,$X_{W_{out}+2}$, $X_{W_{out}+1}$, $X_{W_{out}}$
…	…
K/W_{out}-1	X_{K-1}, …,$X_{K-W_{out}+2}$, $X_{K-W_{out}+1}$, $X_{K-W_{out}}$

4.2.3. Latency Calculations

The overall latency of the uplink accelerator is the sum of the latency of the subblock deinterleaver, the turbo decoder and the output. The IP reads and stores the input data into decoder's input buffer, with de-interleaved memory write address.

The latency of the subblock deinterleaver is

L(subblock-deinterleaver) = Kπ+14.

Where this latency is measured from `sink_sop` to the time of the decoder starts to decode, when the IP was idle. The latency of turbo decoder depends on the block size (K), the number of full decoding iterations (I) to perform as follows

L(decoder) = 26 + (2 × f(K,Ndec) + 14) × 2 × I, when f(K, Ndec) <= 32

26 + (f(K, Ndec) + 46) × 2 × I, when f(K, Ndec) > 32

Where: I is the number of full decoding iterations and K is the block size.

f(K, Ndec) = K/4, if K <= 128;

K/8, if 128 < K <= 512;

K/16, otherwise.

L(output) = ceil(K/16) + 3

Example 1: K = 6144, I = 8, Kπ= 6176

L(deinterleaver) = 6176+14 = 6190 cycles

L(decoder) = 26 + (6144/16 + 46) × 2 × 8 = 6906 cycles

L(output) = ceil(6144/16)+3 = 387 cycles

L(uplink) = 6190 + 6906 + 387 = 13483 cycles

Example 2: K = 40, I = 8, Kπ= 64

L(deinterleaver) = 64+14 = 78 cycles

L(decoder) = 26 + (2*40/4 + 14) × 2 × 8 = 570 cycles

L(output) = ceil(40/16)+3 = 6 cycles

L(uplink) = 78 + 570 + 6 = 654 cycles

4.2.4. Throughput Calculations

The uplink accelerator turbo decoder employs a ping-pong input buffer. Therefore the subblock deinterelaver and turbo decoder can process in parallel at the same time.

The overall throughput can be calculated as

Throughput= K/max(L(deiniterleaver), L(decoder), L(output))*freq. (bits per second).

Example 1: K = 6144, I = 8, Kπ= 6176, freq. = 300 MHz

L(deinterleaver) = 6190 cycles

L(decoder) = 6906 cycles

L(output) = 387 cycles

Throughput(uplink) = 6144/6906*300M = 267 Mbps

Example 2: K = 40, I = 8, Kπ= 64, freq. = 300 MHz

L(deinterleaver) = 78 cycles

L(decoder) = 570 cycles

L(output) = 6 cycles

Throughput(uplink) = 40/570*300M = 21 Mbps

4.3. 4G Turbo-V Signals and Interfaces

Downlink Accelerator

Figure 10. Downlink Accelerator Interfaces

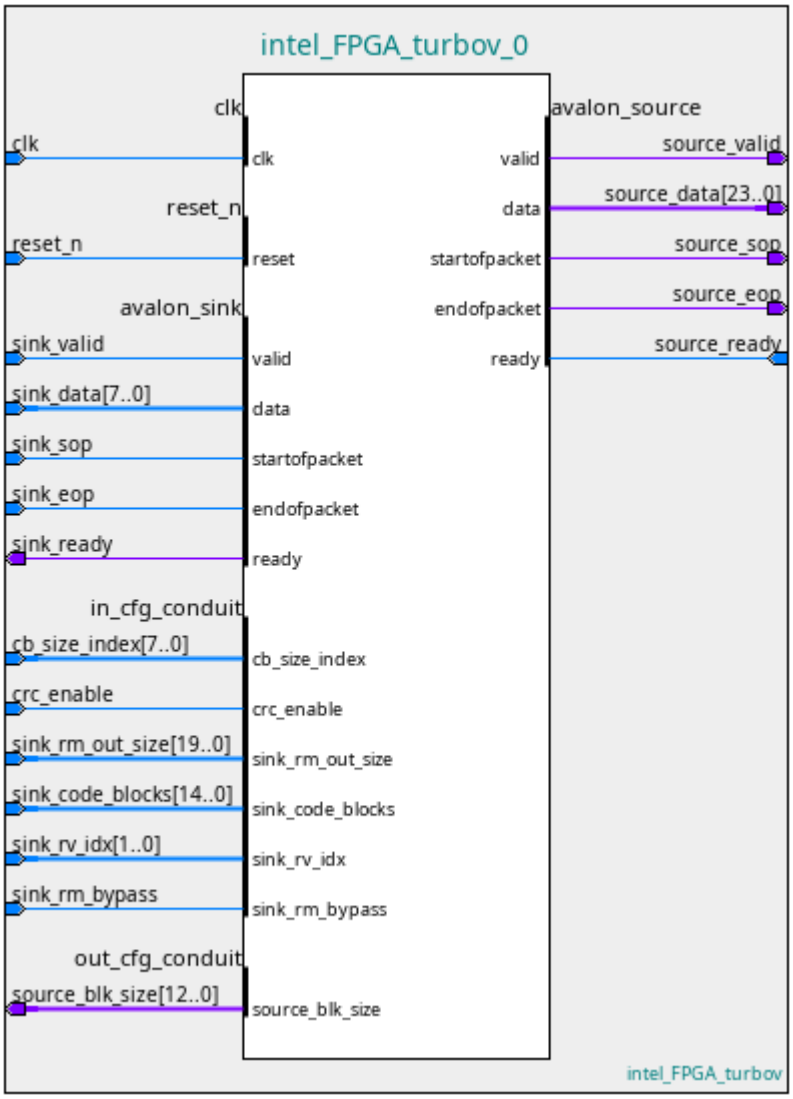


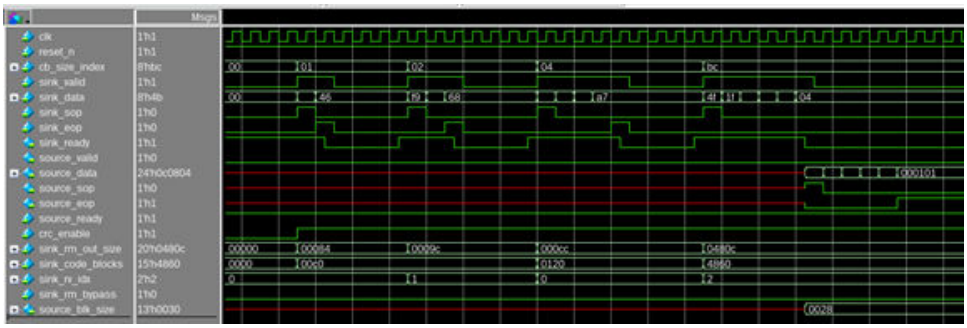
Table 14. Downlink Accelerator Signals

Signal Name	Direction	Bit Width	Description
clk	Input	1	Clock input. All Turbo-V IP interface signals are synchronous to this clock.
reset_n	Input	1	Resets the internal logic of whole IP.
sink_valid	Input	1	Asserted when data at sink_data is valid.
sink_data	Input	8	Typically carries the bulk of the information being transferred.
sink_sop	Input	1	Indicates the start of an incoming packet
sink_eop	Input	1	Indicates the end of an incoming packet

continued...

Signal Name	Direction	Bit Width	Description
sink_ready	Output	1	Indicates when the IP can accept data
crc_enable	Input	1	Enables the CRC block
cb_size_index	Input	8	Input code block size K
sink_rm_out_size	Input	20	Rate matcher output block size, corresponding to E.
sink_code_blocks	Input	15	Soft buffer size for current code block *Ncb*
sink_rv_idx	Input	2	Redundancy version index (0,1,2 or 3)
sink_rm_bypass	Input	1	Enables bypass mode in the rate matcher
source_valid	Output	1	Asserted by the IP when there is valid data to output.
source_data	Output	24	Carries the bulk of the information transferred. This information is available where valid is asserted.
source_sop	Output	1	Indicates the beginning of a packet.
source_eop	Output	1	Indicates the end of a packet.
source_ready	Input	1	Data reception is valid where the ready signal is asserted.
Source_blk_size	Output	13	Output code block size K

Figure 11. Downlink Signals



Uplink Accelerator

Figure 12. Uplink Accelerator Interfaces

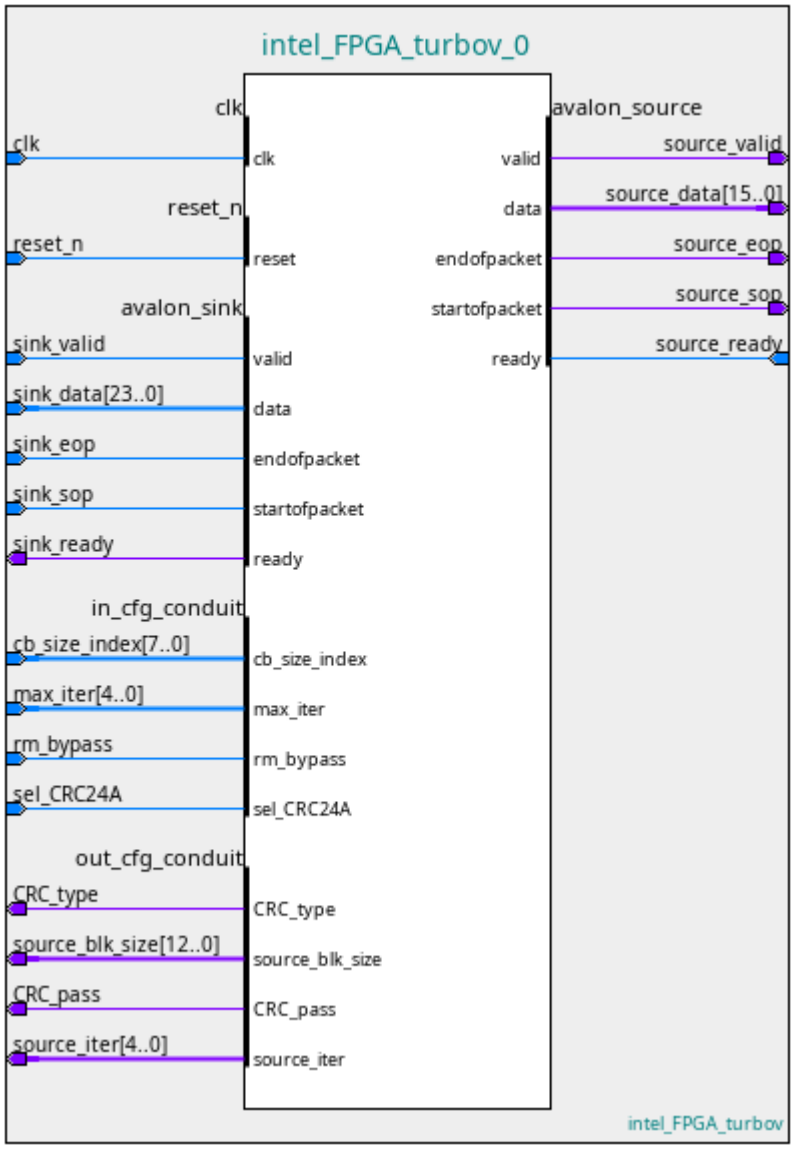


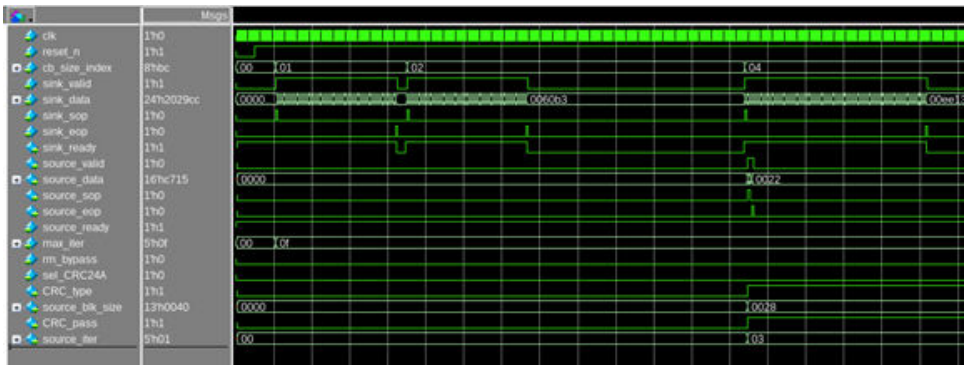
Table 15. Uplink Accelerator Signals

Signal	Direction	Bit Width	Description
clk	Input	1	Clock input. All Turbo-V IP interface signals are synchronous to this clock.
reset_n	Input	1	Reset of input clock signal
sink_valid	Input	1	Avalon streaming input valid
sink_data	Input	24	Avalon streaming input data

continued...

Signal	Direction	Bit Width	Description
sink_sop	Input	1	Avalon streaming input start of packet
sink_eop	Input	1	Avalon streaming input end of packet
sink_ready	Input	1	Avalon streaming input ready
cb_size_index	Input	8	Block size index
max_iteration	Input	5	Specify the maximum number of half-iterations
rm_bypass	Input	1	Enables bypass mode
sel_CRC24A	Input	1	Specify the type of CRC that you need for the current data block: • 0: CRC24B • 1: CRC24A
source_valid	Output	1	Avalon streaming output valid
source_data	Output	16	Avalon streaming output data
source_sop	Output	1	Avalon streaming output start of packet
source_eop	Output	1	Avalon streaming output end of packet
source_ready	Output	1	Avalon streaming output ready
CRC_type	Output	1	Indicates the type of CRC that was used for the current data block: • 0: CRC24A • 1: CRC24B
source_blk_size	Output	13	Specifies the outgoing block size
CRC_pass	Output	1	Indicates whether CRC was successful: • 0: Fail • 1: Pass
source_iter	Output	5	Shows the number of half iterations after which the Turbo decoder stops processing the current data block.

Figure 13. Uplink Signals



4.3.1. Avalon Streaming Interfaces in DSP Intel FPGA IP

Avalon streaming interfaces define a standard, flexible, and modular protocol for data transfers from a source interface to a sink interface.

The input interface is an Avalon streaming sink and the output interface is an Avalon streaming source. The Avalon streaming interface supports packet transfers with packets interleaved across multiple channels.

Avalon streaming interface signals can describe traditional streaming interfaces supporting a single stream of data without knowledge of channels or packet boundaries. Such interfaces typically contain data, ready, and valid signals. Avalon streaming interfaces can also support more complex protocols for burst and packet transfers with packets interleaved across multiple channels. The Avalon streaming interface inherently synchronizes multichannel designs, which allows you to achieve efficient, time-multiplexed implementations without having to implement complex control logic.

Avalon streaming interfaces support backpressure, which is a flow control mechanism where a sink can signal to a source to stop sending data. The sink typically uses backpressure to stop the flow of data when its FIFO buffers are full or when it has congestion on its output.

Related Information

Avalon Interface Specifications

5. 4G Turbo-V Intel FPGA IP User Guide Document Archive

For the latest and previous versions of this document, refer to: 4G Turbo-V Intel FPGA IP User Guide. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

6. Document Revision History for the 4G Turbo-V Intel FPGA IP User Guide

Date	IP Version	Quartus Prime Software Version	Changes
2024.04.01	2.0.4	24.1	• Added support for Agilex 5 devices • Updated *Performance and Resource Utilization*
2023.04.03	2.0.0	23.1	• Updated the product family name to "Intel Agilex 7." • Added product ordering code
2022.10.25	2.0.0	22.2	Corrected `sel_CRC24A` signal description
2022.06.20	2.0.0	22.2	• Updated: — *Performance and resources* • Added: — *Input and Output Data Formats* — *Latency Calculations* — *Throughput Calculations* • Deleted: — *Release Information* — `sink_filler_bits`,`sink_error` , and `source_error` from *Downlink Signals* and `conf_valid`, `source_error`, and `conf_ready` from *Uplink Signals*
2020.11.18	1.0.0	20.1	Removed table in *4G Turbo-V Performance and Resource Utilization*
2020.06.02	1.0.0	20.1	Initial release.