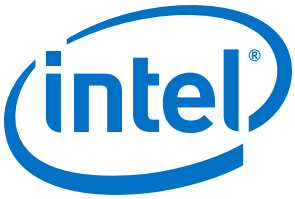


4G Turbo-V Intel® FPGA IP User Guide

Updated for Intel® Quartus® Prime Design Suite: **20.1**

IP Version: **1.0.0**

Subscribe
Send Feedback

UG-20279 | 2020.11.18
Latest document on the web: **PDF** | **HTML**

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

The downlink accelerator includes:

- Code block cyclic redundancy code (CRC) attachment
- Turbo encoder
- Turbo 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

Intel offers the following device support levels for Intel FPGA IP:

- Advance support—the IP is available for simulation and compilation for this device family. FPGA programming file (`.pof`) support is not available for Quartus Prime Pro Stratix 10 Edition Beta software and as such IP timing closure cannot be guaranteed. Timing models include initial engineering estimates of delays based on early post-layout information. The timing models are subject to change as silicon testing improves the correlation between the actual silicon and the timing models. You can use this IP core for system architecture and resource utilization studies, simulation, pinout, system latency assessments, basic timing assessments (pipeline budgeting), and I/O transfer strategy (data-path width, burst depth, I/O standards tradeoffs).
- Preliminary support—Intel verifies the IP core with preliminary timing models for this device family. The IP core meets all functional requirements, but might still be undergoing timing analysis for the device family. You can use it in production designs with caution.
- Final support—Intel verifies the IP with final timing models for this device family. The IP meets all functional and timing requirements for the device family. You can use it in production designs.

Table 1. 4G Turbo-V IP Device Family Support

Device Family	Support
Intel Agilex™	Advance
Intel Arria® 10	Final
Intel Stratix® 10	Advance
Other device families	No support

1.3. Release Information for the 4G Turbo-V Intel FPGA IP

Intel FPGA IP versions match the Intel Quartus® Prime Design Suite software versions until v19.1. Starting in Intel 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 Intel Quartus Prime software version. A change in:

- X indicates a major revision of the IP. If you update the Intel 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 2. 4G Turbo-V IP Release Information

Item	Description
Version	1.0.0
Release Date	April 2020

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 v19.1. Only use these approximate results for early estimation of FPGA resources (e.g. adaptive logic modules (ALMs)) that a project requires. The target frequency is 300 MHz.

Table 3. Downlink Accelerator Resource Utilization and Maximum Frequency for Intel Arria 10 Devices

Module	f_{MAX} (MHz)	ALMs	ALUTs	Registers	Memory (Bits)	RAM Blocks (M20K)	DSP Blocks
Downlink accelerator	325.63	9,373	13,485	14,095	297,472	68	8
CRC attachment	325.63	39	68	114	0	0	0
Turbo encoder	325.63	1,664	2,282	1154	16,384	16	0
Rate matcher	325.63	7,389	10,747	12,289	274,432	47	8
Subblock interleaver	325.63	2,779	3,753	5,559	52,416	27	0
Bit collector	325.63	825	1,393	2,611	118,464	13	4
Bit selector and pruner	325.63	3,784	5,601	4,119	103,552	7	4

Table 4. Uplink Accelerator Resource Utilization and Maximum Frequency for Intel Arria 10 Devices

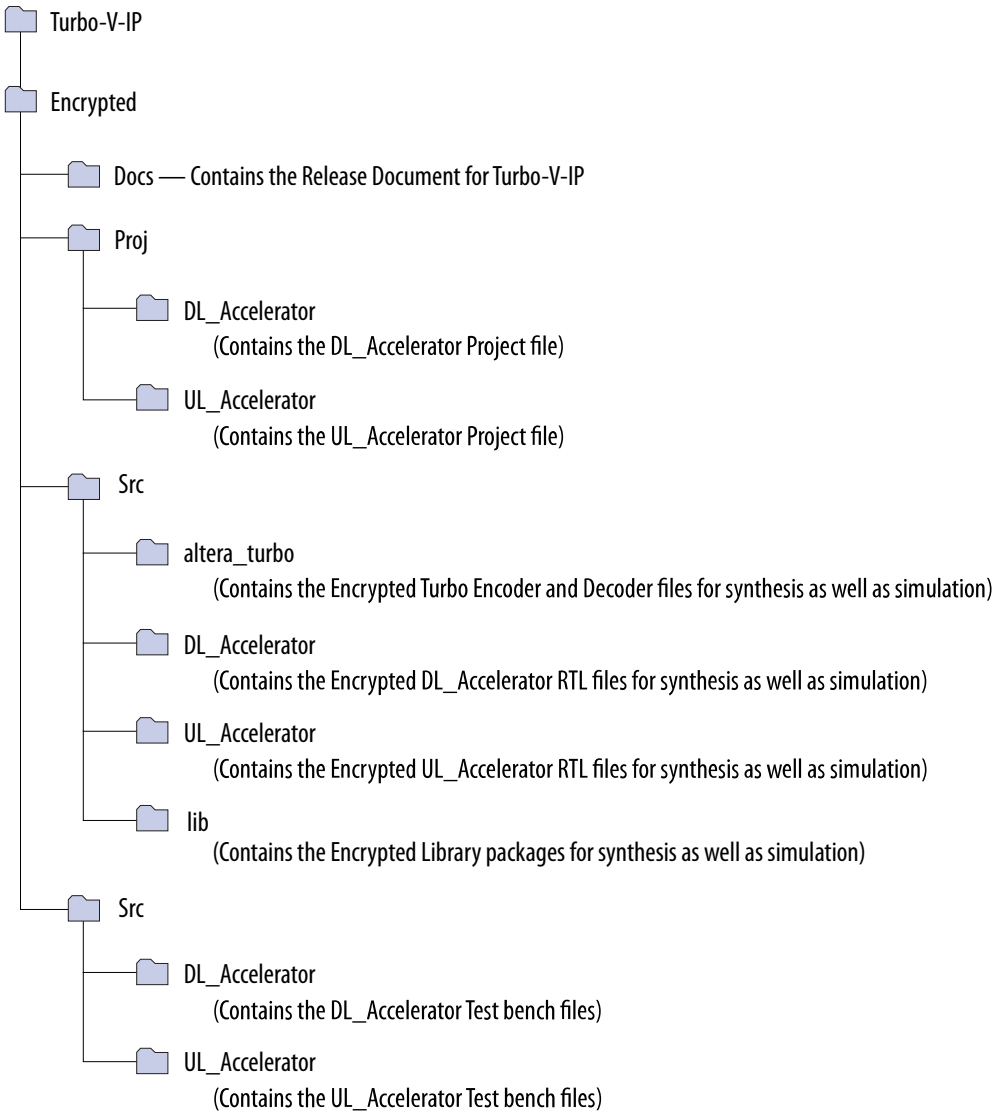
Module	f_{MAX} (MHz)	ALMs	Registers	Memory (Bits)	RAM Blocks (M20K)	DSP Blocks
Uplink accelerator	314.76	29480	30,280	868,608	71	0
Subblock deinterleaver	314.76	253	830	402,304	27	0
Turbo decoder	314.76	29,044	29,242	466,304	44	0

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

2.1. 4G Turbo-V IP Directory Structure

You must manually install the IP from the IP installer.

Figure 1. Installation Directory Structure



2.2. Generating a 4G Turbo-V IP

You can generate a downlink or uplink accelerator. For the uplink accelerator, replace `dl` with `ul` in directory or file names.

1. Open the Intel Quartus Prime Pro software.
2. Select **File ➤ New Project Wizard**.
3. Click **Next**.
4. Enter Project name `dl_fec_wrapper_top` and enter the project location.
5. Select **Arria 10** device.
6. Click **Finish**.
7. Open the `dl_fec_wrapper_top.qpf` file available at project directory

   The project wizard appears.
8. On the **Platform Designer** tab:
   a. Create the `dl_fec_wrapper_top.ip` file using `hardware tcl` file.
   b. Click **Generate HDL** to generate the design files.
9. On the **Generate** tab, click **Generate Test bench system**.
10. Click **Add All** to add the synthesis files to the project.

    The files are in `src\ip\dl_fec_wrapper_top\dl_fec_wrapper_10\synth`.
11. Set `dl_fec_wrapper_top.v` file as top level entity.
12. Click **Start Compilation** to compile this project.

2.3. Simulating a 4G Turbo-V IP

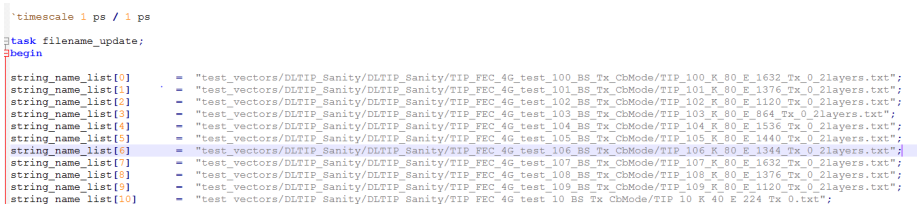
This task is for simulating a downlink accelerator. To simulate an uplink accelerator replace `dl` with `ul` in each directory or file name.

1. Open the ModelSim 10.6d FPGA Edition simulator.
2. Change the directory to `src\ip\dl_fec_wrapper_top_tb \dl_fec_wrapper_top_tb\sim\mentor`
3. Change the `QUARTUS_INSTALL_DIR` into your Intel Quartus Prime directory in the `msim_setup.tcl` file, which is in `\sim\mentor` directory
4. Enter the command `do load_sim.tcl` command in transcript window.

   This command generates the library files and compiles and simulates the source files in the `msim_setup.tcl` file. The test vectors are in `filename_update.sv` in the `\sim` directory.

Figure 2. The `filename update` File Structure

- Corresponding test vector files are in `sim\mentor\test_vectors`
- `Log.txt` contains the result of every test packets.
- For the downlink accelerator, `encoder_pass_file.txt` contains the pass report of every index of test packets and `encoder_file_error.txt` contains the fail report of every index of test packets.
- For the uplink accelerator, `Error_file.txt` contains the fail report of every index of test packets.



3. 4G Turbo-V Intel FPGA IP Functional Description

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

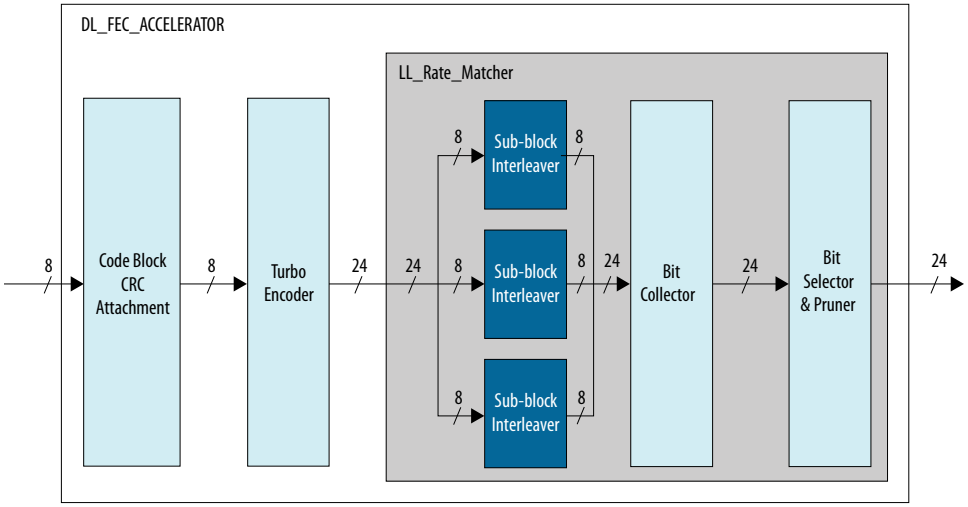
3.1. 4G Turbo-V Architecture

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

4G Downlink Accelerator

Figure 3. 4G Downlink Accelerator

The 4G Turbo downlink accelerator consists of a code block CRC attachment block and a Turbo encoder (Intel Turbo FPGA IP) and rate matcher. 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.

The 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 convolutional encoders and one Turbo code internal interleaver. For more information about the Turbo encoder, refer to the *Turbo IP Core User Guide*.

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. 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.

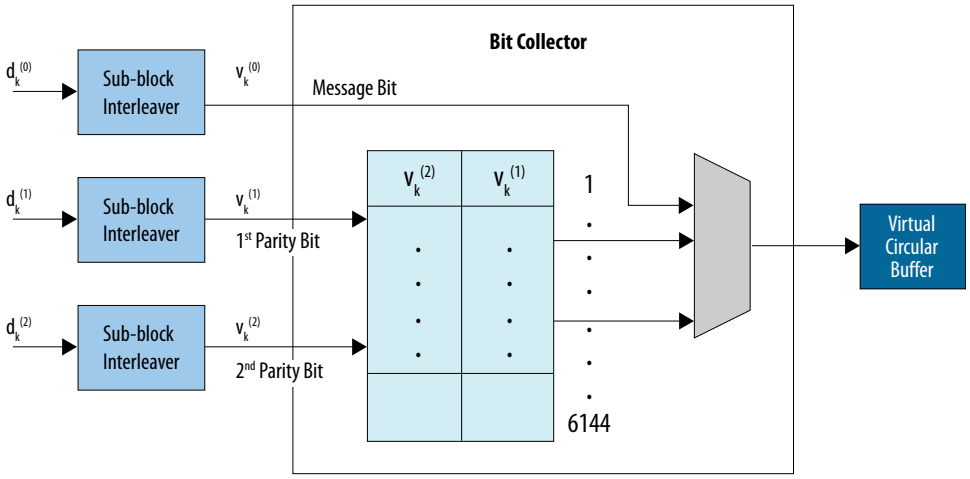
The downlink accelerator sets up the subblock interleaver for each output stream from Turbo coding. The streams include a message bit stream, 1st parity bit stream and 2nd parity bit stream.

The input and output of the subblock interleaver is 24 bits wide.

The bit collector combines the streams that come from the subblock interleaver. This block contains buffers that store:

- Messages and filler enabling bits from the subblock interleaver.
- The subblock interleaver parity bits and their respective filler bits.

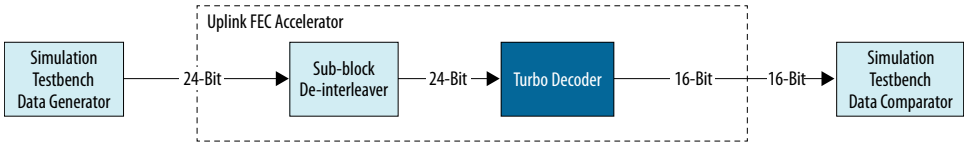
Figure 4. Bit Collector



4G Channel Uplink Accelerator

Figure 5. 4G Channel Uplink Accelerator

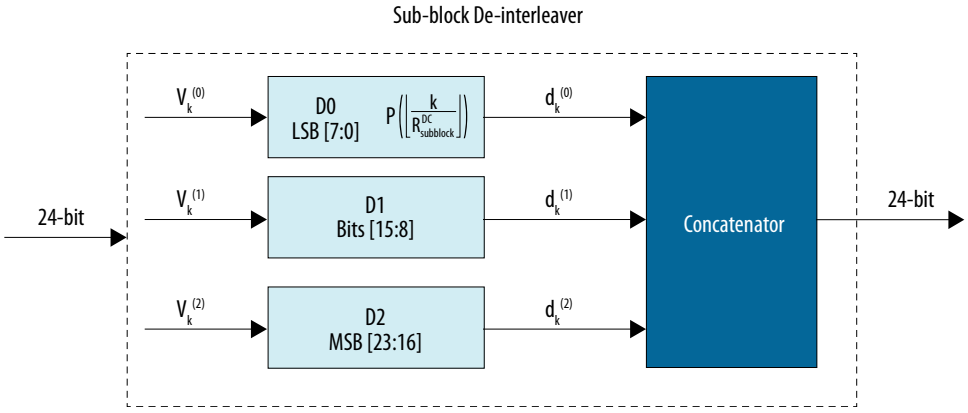
The 4G Turbo uplink accelerator consists of an subblock deinterleaver and a turbo decoder (Intel Turbo FPGA IP).



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

The latency of the ready signal is 0.

Figure 6. 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).

The latency of the output data of the subblock deinterleaver depends on the input block size K_п. The IP reads the data only after you write the K_п code block size of input data. Hence the latency of the output also includes the write time. The latency in the subblock interleaver output data is K_п+17.

The Turbo decoder calculates the most likely transmitted sequence, based on the samples that it receives. For a detailed explanation, refer to the *Turbo Core IP User Guide*. Decoding of error correcting codes is a comparison of the probabilities for different convolutional codes. 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 reorder data in this process.

Related Information

Turbo IP Core User Guide

3.2. 4G Turbo-V Signals and Interfaces

Downlink Accelerator

Figure 7. Downlink Accelerator Interfaces

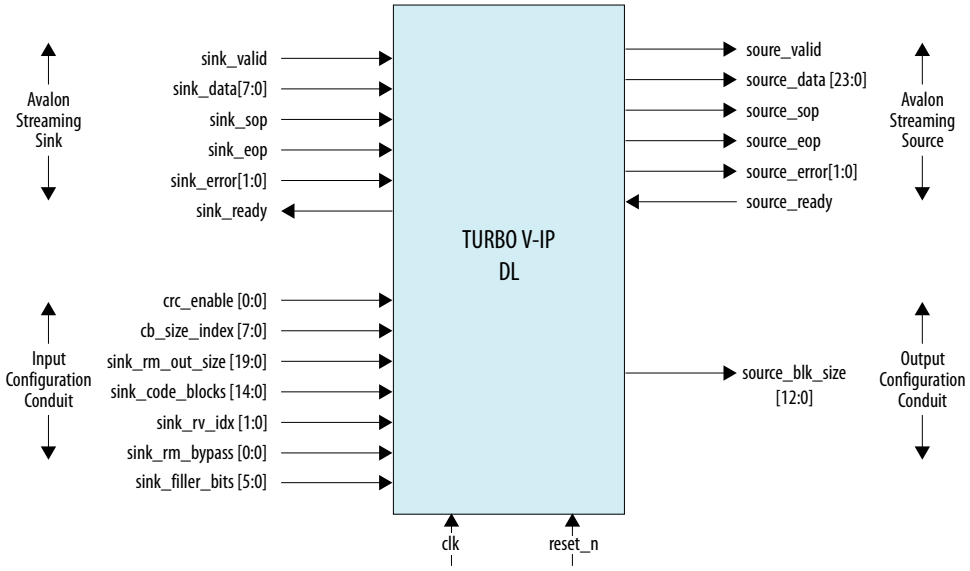


Table 5. Downlink Accelerator Signals

Signal Name	Direction	Bit Width	Description
clk	Input	1	300 MHz 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. When sink_valid is not asserted, the IP strops processing until sink_valid is reasserted.
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
sink_ready	Output	1	Indicates when the IP can accept data
Sink_error	Input	2	Two-bit mask to indicate errors affecting the data transferred in the current cycle.
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
sink_filler_bits	Input	6	The number of filler bits the IP inserts at the transmitter when the IP performs code block segmentation.
source_valid	Output	1	Asserted by the IP when there is valid data to output.

continued...

Signal Name	Direction	Bit Width	Description
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_error	Output	2	Error signal propagated from Turbo Encoder indicating Avalon-ST protocol violations on source side • 00: No error • 01: Missing start of packet • 10: Missing end of packet • 11: Unexpected end of packet Other types of errors may also be marked as 11.
Source_blk_size	Output	13	Output code block size K

Uplink Accelerator

Figure 8. Uplink Accelerator Interfaces

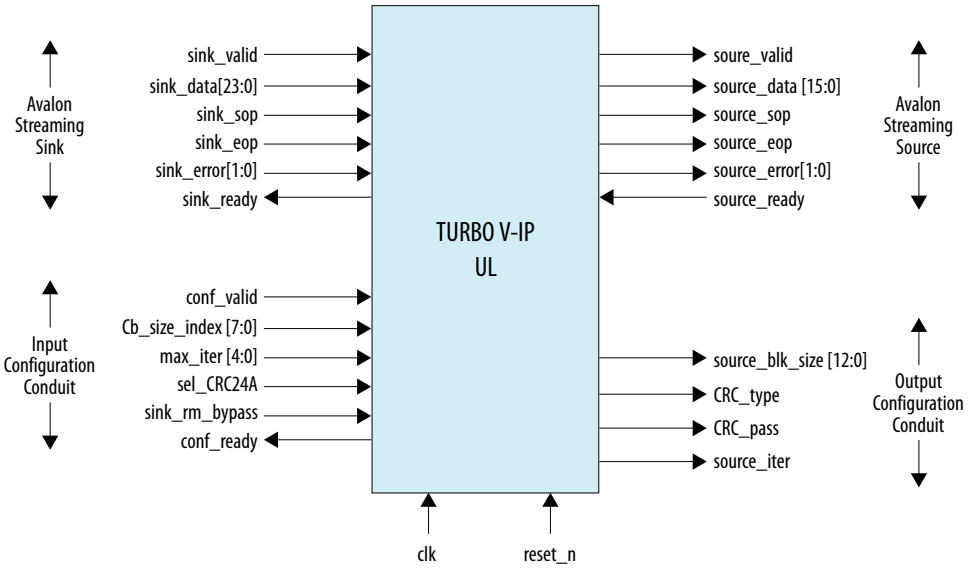


Table 6. Uplink Accelerator Signals

Signal	Direction	Bit Width	Description
clk	Input	1	300 MHz 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
sink_sop	Input	1	Avalon streaming input start of packet
sink_eop	Input	1	Avalon streaming input end of packet

continued...

Signal	Direction	Bit Width	Description
sink_ready	Input	1	Avalon streaming input ready
conf_valid	Input	1	Input configuration conduit valid
cb_size_index	Input	8	Block size iteration index
max_iteration	Input	5	Maximum iteration
rm_bypass	Input	1	Enables bypass mode
sel_CRC24A	Input	1	Specifies the type of CRC that you need for the current data block: • 0: CRC24A • 1: CRC24B
conf_ready	Input	1	Input configuration conduit ready
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_error	Output	2	Error signal indicating Avalon streaming protocol violations on source side: • 00: No error • 01: Missing start of packet • 10: Missing end of packet • 11: Unexpected end of packet Other types of errors may also be marked as 11.
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.

3.2.1. 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

3.3. 4G Turbo-V Timing Diagrams

Downlink Accelerator

Figure 9. Timing Diagram for Write Logic with Codeblock 40

The IP:

- Places null 20 bits in column 0 to 19 and writes the data bits from column 20.
- Writes all 44 bits to memory in 6 clock cycles.
- Writes trellis termination bits into column 28 to 31.
- Increments write address for each row.
- Generates write enable signal for 8 individual RAM at a time.

The IP does not write filler bits into RAM. Instead, the IP leaves the place holder for filter bits in the RAM and inserts the NULL bits into the output during the read process. The first write starts from column 20.

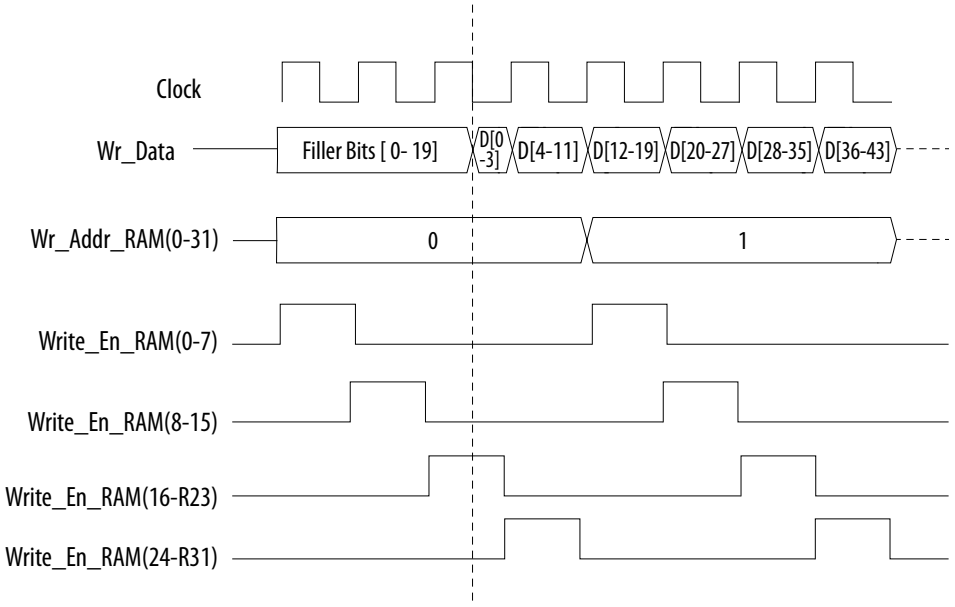


Figure 10. Timing Diagram for Read Logic with Codeblock 40

For each read, you see 8 bits in one clock cycle but only two bits are valid. The IP writes these two bits into the shift register. When the IP forms 8 bits it sends them to the output interface.

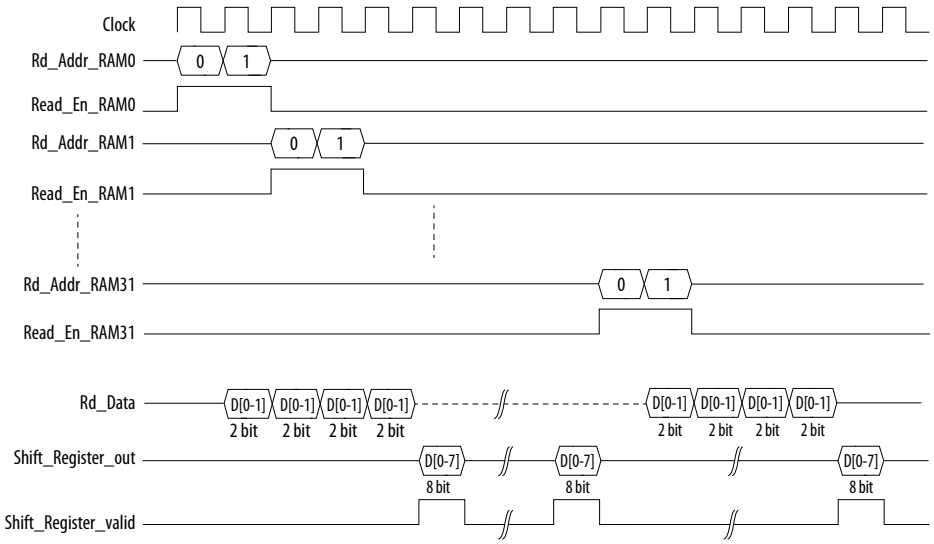


Figure 11. Timing Diagram for Write Logic with Codeblock 6144

The filler bits are from column 0 to 27 and the data bits are from column 28. The IP:

- Writes all 6,148 bits to memory in 769 clock cycles.
- Writes trellis termination bits into column 28 to 31.
- Increments write address for each row.
- Generates write enable signal generated for 8 individual RAM at a time.

The IP does not write filler bits into RAM. Instead the IP leaves the place holder for filter bits over in the RAM and inserts the NULL bits into output during the read process. The first write starts from column 28.

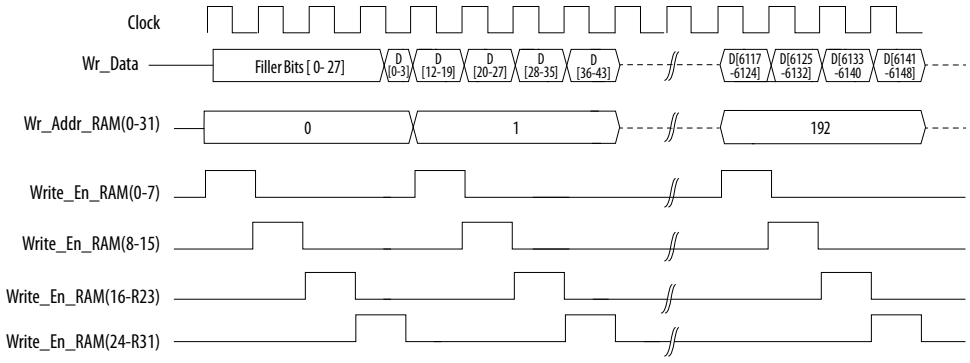
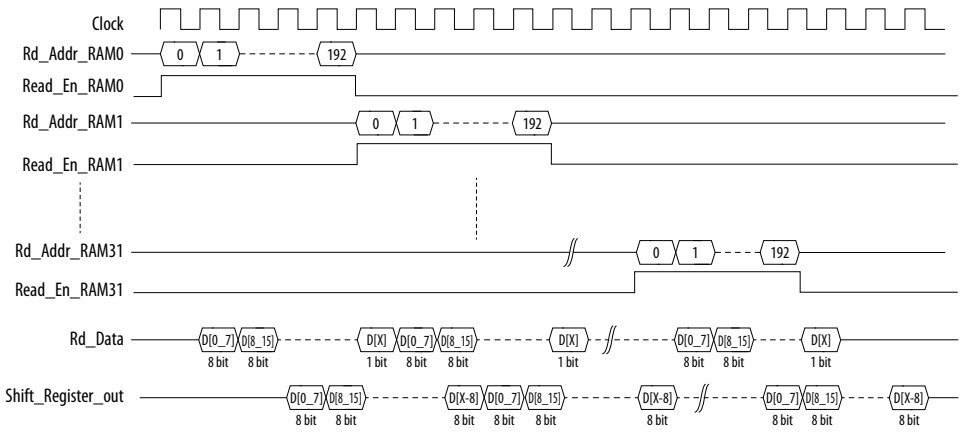


Figure 12. Timing Diagram for Read Logic with Codeblock 6144

On the read side, each read gives 8 bits. While reading the 193rd row, the IP read 8 bits, but only one bit is valid. The IP forms eight bits with shift registers and sends them out by reading from the next column.



Uplink Accelerator

Figure 13. Input Timing Diagram

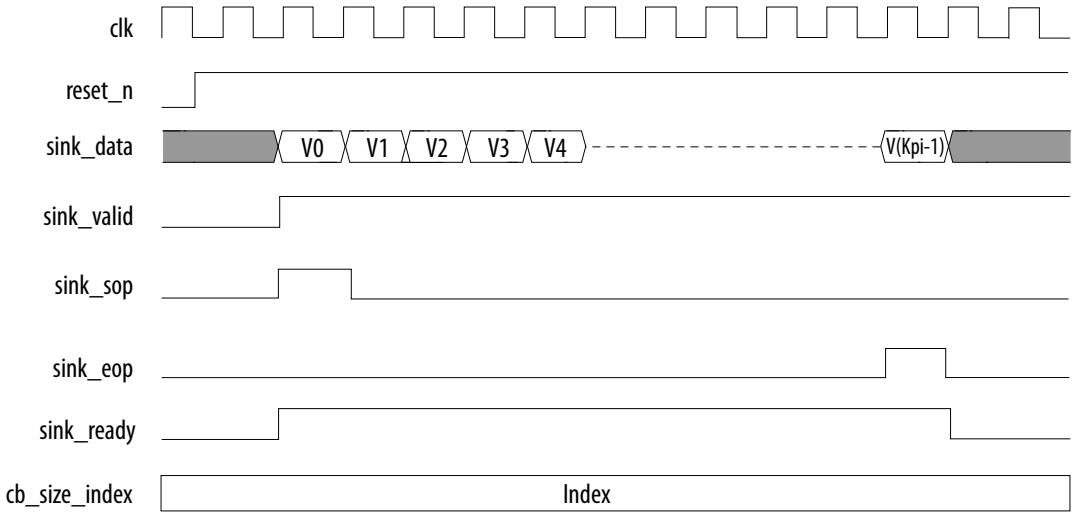
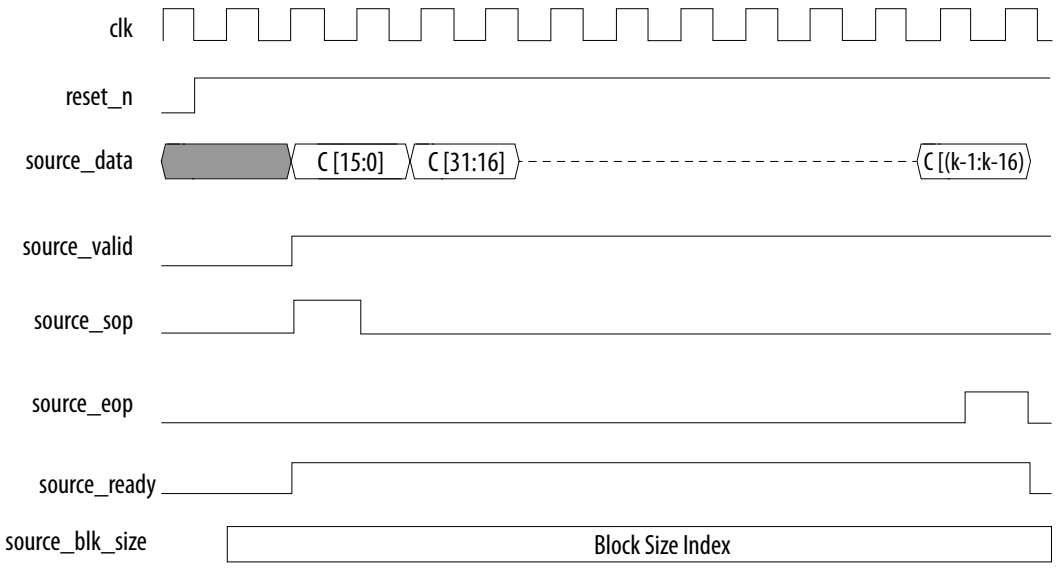


Figure 14. Output Timing Diagram



3.4. 4G Turbo-V Latency and Throughput

The latency is measured between input first packet SOP to output first packet SOP. The processing time is measured between input first packet SOP to output last packet EOP.

Downlink accelerator

The throughput is the rate at which the IP can pump the input into the downlink accelerator as it is ready.

Table 7. Downlink Accelerator Latency, Processing Time, and Throughput

With the maximum K size of 6,144 and E size of 11,522. Processing time measured for 13 code blocks. Clock speed is 300 MHz.

K	E	Latency		Processing time		Input Throughput
		(cycles)	(us)	(cycles)	(us)	(%)
6,144	11,552	3,550	11.8	14,439	48.13	95

Figure 15. Latency and Processing Time Calculation

The figure shows the procedure to calculate latency, processing time, and throughput.

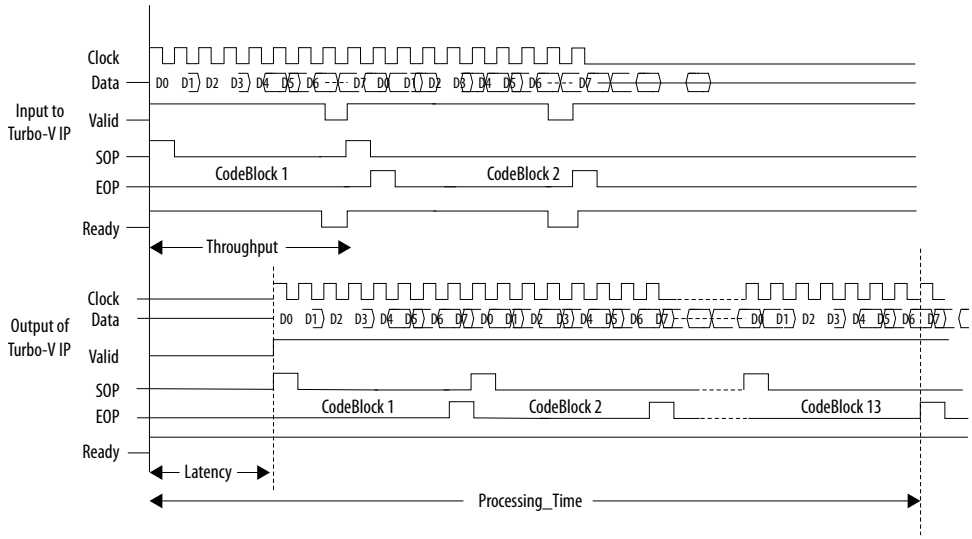


Figure 16. K Size versus Latency

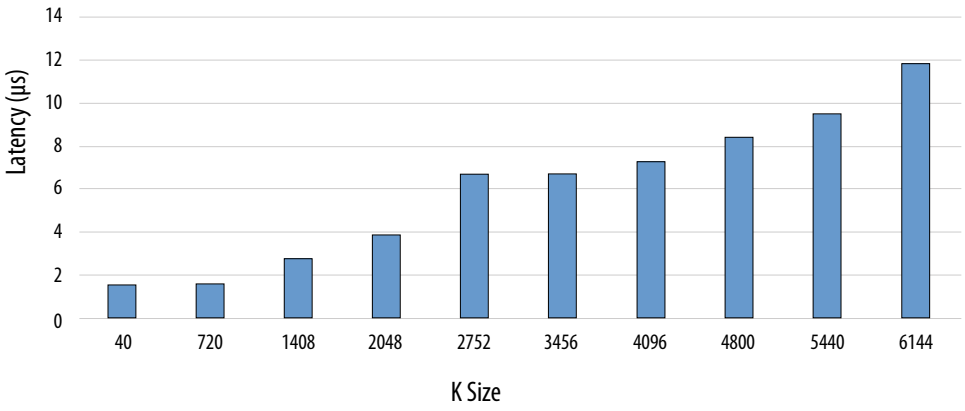
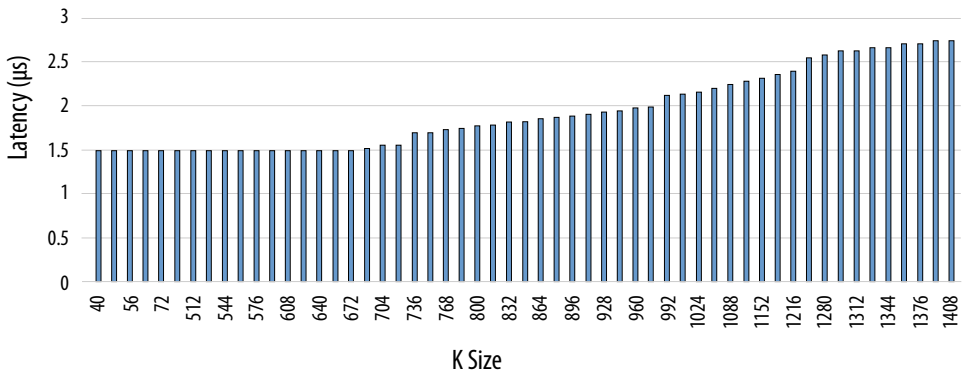


Figure 17. K Size versus Latency

k=40 to 1408



Uplink Accelerator

Table 8. Uplink Accelerator Latency and Processing Time

With max iteration number = 6. Clock speed is 300 MHz.

K	E	Latency		Processing time	
		(cycles)	(us)	(cycles)	(us)
86	40	316	1.05	318	1.06
34,560	720	2,106	7.02	2,150	7.16
34,560	1,408	3,802	12.67	3,889	12.96
34,560	1,824	4,822	16.07	4,935	16.45
28,788	2,816	7,226	24.08	7,401	24.67
23,742	3,520	8,946	29.82	9,165	30.55
34,560	4,032	10,194	33.98	10,445	34.81
26,794	4,608	11,594	38.64	11,881	39.60
6,480	5,504	13,786	45.95	14,129	47.09
12,248	6,144	15,338	51.12	15,721	52.40

Table 9. Uplink Accelerator Latency and Processing Time

With max iteration number = 8

K	E	Latency		Processing time	
		(cycles)	(us)	(cycles)	(us)
86	40	366	1.22	368	1.22
34,560	720	2,290	7.63	2,334	7.78
34,560	1,408	4,072	13.57	4,159	13.86
34,560	1,824	5,144	17.14	5,257	17.52
28,788	2,816	7,672	25.57	7,847	26.15

continued...

23,742	3,520	9,480	31.6	9,699	32.33
34,560	4,032	10,792	35.97	11,043	36.81
26,794	4,608	12,264	40.88	12,551	41.83
6,480	5,504	14,568	48.56	14,911	49.70
12,248	6,144	16,200	54	16,583	55.27

Figure 18. K Size vs Latency

For max_iter=6

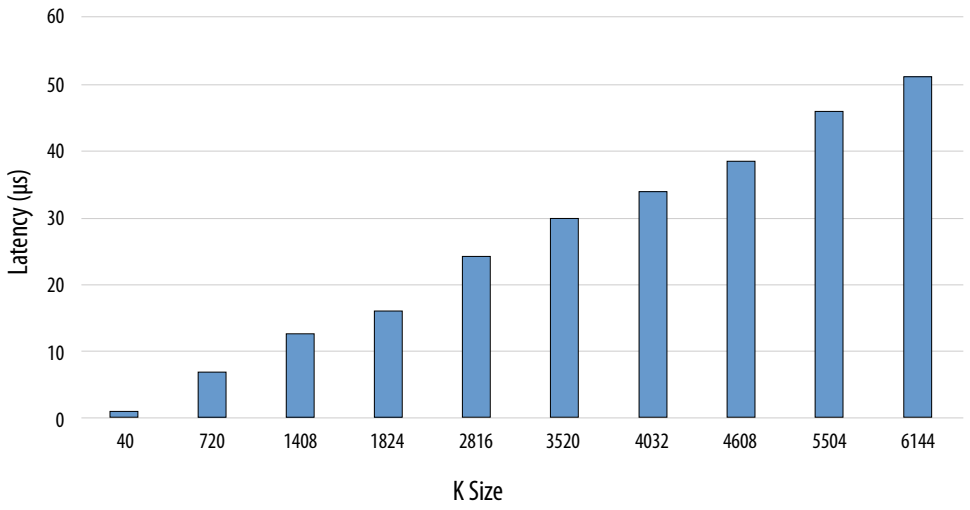


Figure 19. K Size vs Processing Time

For max_iter=6

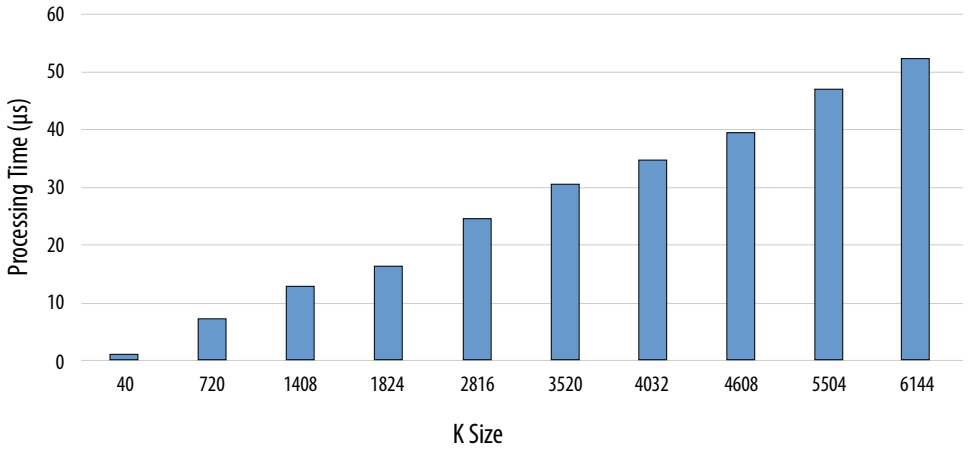


Figure 20. K Size vs Latency

For max_iter=8

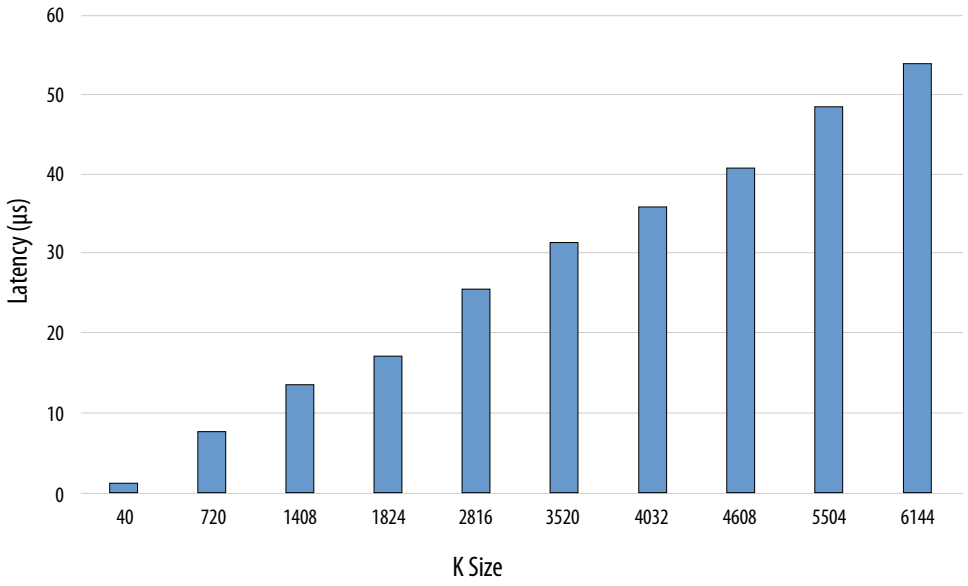
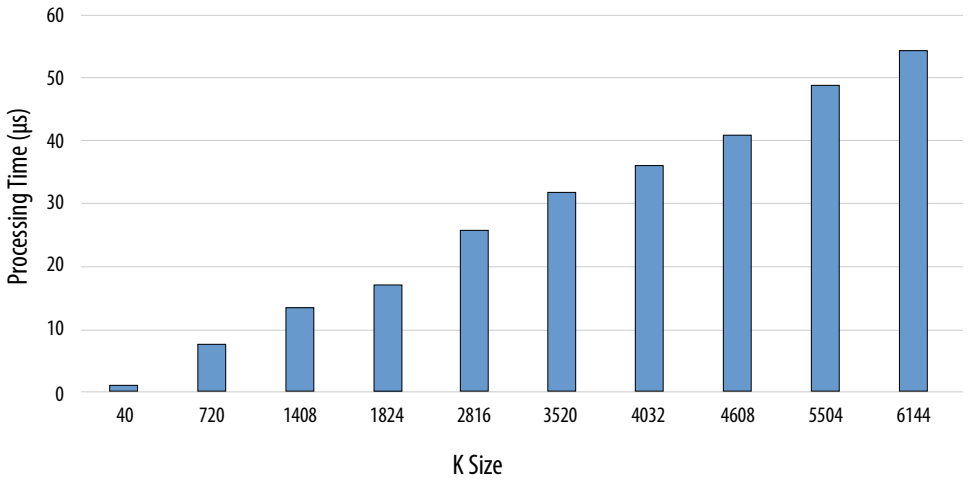


Figure 21. K Size vs Processing Time

For max_iter=8



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

Date	IP Version	Intel Quartus Prime Software Version	Changes
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.