KeyStone Training

Turbo Decoder (TCP3D)

Agenda

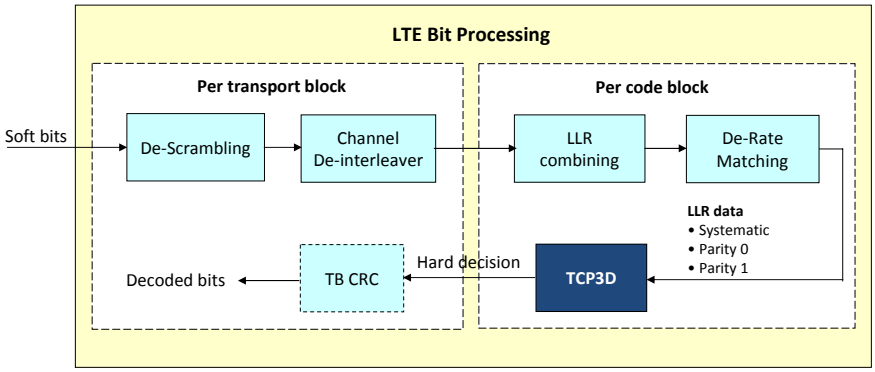
- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Example
- TCP3D Driver

TCP3D Architecture

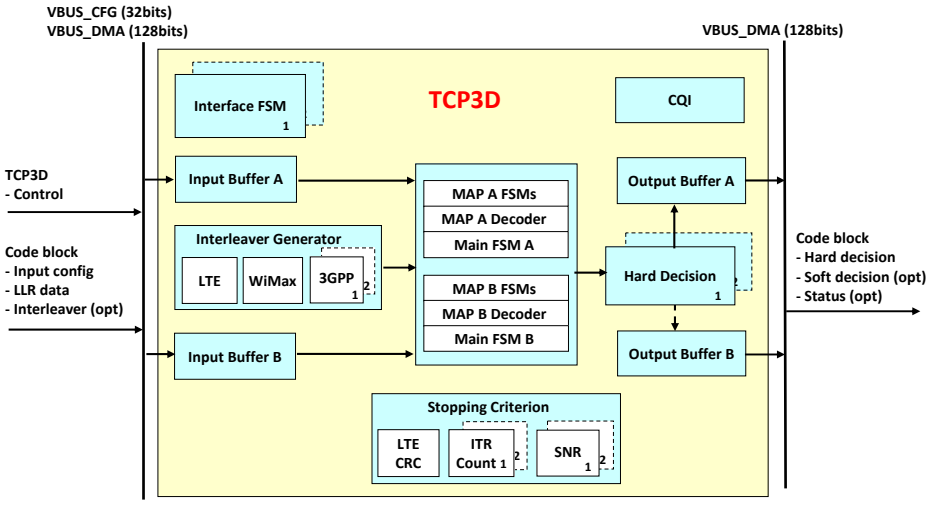
- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Example
- TCP3D Driver

Overview

- TCP3D is a programmable peripheral for decoding of 3GPP (WCDMA, HSUPA, HSUPA+, TD_SCDMA), LTE, and WiMax turbo codes.
- Turbo decoding is a part of bit processing.



TCP3D Architecture



Key Features

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Example
- TCP3D Driver

TCP3D Key Features (1/2)

- Supports 3GPP Rel-7 and older (WCDMA), LTE, and WiMAX turbo decoding
- Native Code Rate: 1/3
- Radix 4 Binary and Duo-Binary MAP Decoders
- Dual MAP decoders for non-contentious Interleavers
- Split decoder mode: TCP3D works as 2 independent, single MAP decoders
- Max Star and Max log-map algorithms
- Double Buffer input memory for lower latency transfers (except in split mode)
- 128 bit data bus for reduced latency transfers
- Input data bit width: 6 bits
- Programmable hard decision bit ordering within a 128-bit word: 0-127 or 127-0
- Soft output information for systematic and parity bits: 8 bits
- Extrinsic scaling per MAP for up to 8 iterations (Both Max and Max Star)

TCP3D Key Features (2/2)

- Block sizes supported: 40 to 8192
- Programmable Sliding window sizes {16, 32, 48, 64, 96, 128}
- Max number of iterations: 1 to 15
- Min number of iterations: 1 to 15
- SNR stopping criterion: 0 to 20 dB threshold
- LTE CRC stopping criterion
- LTE, WCDMA and WiMAX Hardware Interleaver Generators
- Channel Quality Indication
- Emulation support
- Low DSP pre-processing load
- Runs in parallel with DSP
- Targets base station environment

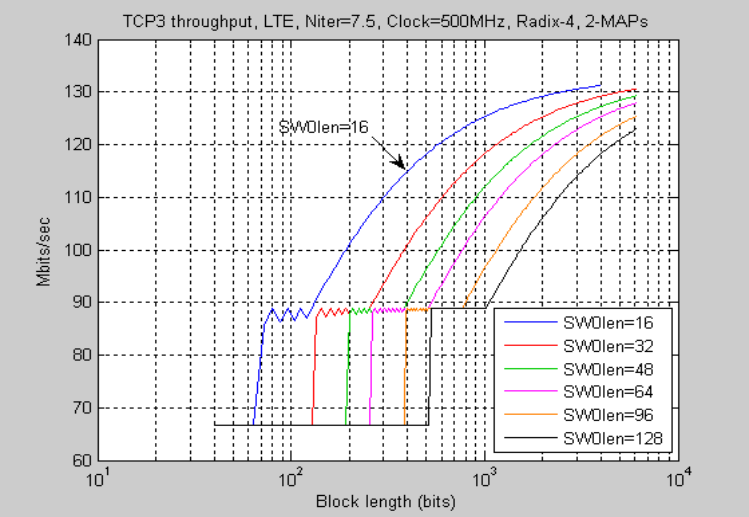
TCP2 vs. TCP3D

Feature	TCP2	TCP3D
Clock	333 MHz (CPU/3)nominal	500MHz (CPU/2) fully synchronous
Prolog Reduction	Yes	No latency – not used
Standalone Frame Size	20730	8192 (no CDMA2000 support)
Code Rates	All rates derived from 1/5	Native 1/3
Input Format	6-bit	6-bit
Input Sign	Programmable	Not Programmable
Stopping Criterion	SNR or CRC	SNR and CRC (for LTE only)
Re-encoding	Yes	No but CQI is supported
Log Equation	Max and Max*	Max and Max*
Interleaver Load	Concurrently with first MAP	ITG included- concurrently with 1st MAP
Hard Decision Ordering	Programmable	Programmable
Debugging Features	Pause after each MAP Emulation Support	Pause after each MAP Emulation Support
Extrinsic Scaling	Yes, used in max-log-map	Yes, used in both
HIU/EDMA Interface	VBUSP/DMA, 64bits@333 MHz	128bits@500MHz bridge
Memory Sleep Mode	Yes	No

TCP3D Throughput

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Example
- TCP3D Driver

Decoding Throughput for LTE



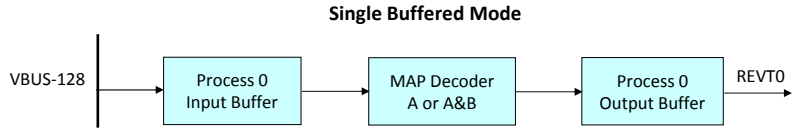
Niter = 6, Throughput from 83.3 Mbps up to 164 Mbps

TCP3D Processing Modes

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Examples
- TCP3D Driver

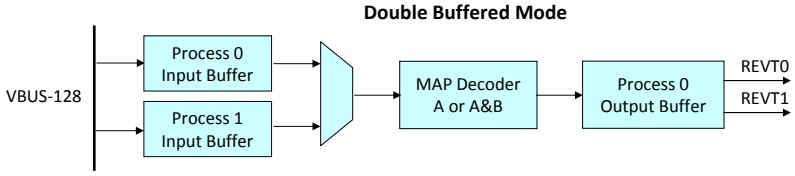
Single Buffered Mode

In the single buffered mode, the TCP3D operates as a single turbo-decoder with a single set of input buffers for the code block and a single set of input configuration registers.



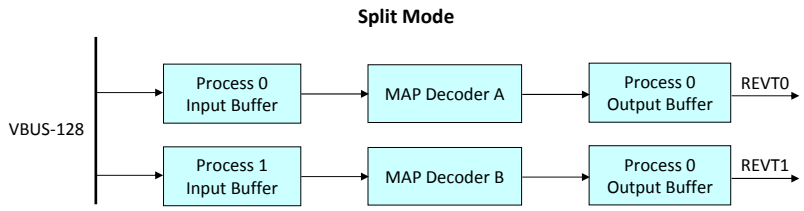
Double Buffered Mode

Double buffered mode (i.e. ping/pong mode) provides a very efficient system interface to the TCP3D. In this mode, the TCP3D operates as a single turbo-decoder with two complete sets of input configuration registers and input data buffers. The system can utilize this mode by loading the next code block to be decoded while the TCP3D is decoding the current code block. This approach can result in a reduction in the data transfer latency for the next code block. The two sets of resources are referred to as Process 0 and Process 1.



Split Mode

Split mode configures the TCP3D resources into two independent turbo decoder coprocessors that share a common VBUS interface. In this mode, the TCP3D can decode 2 code blocks in parallel. The two coprocessors are referred to as Process 0 and Process 1. Only for WCDMA.



Recommended Processing Modes

- Double buffer mode
  - LTE
  - WiMax
- Split mode
  - WCDMA
  - HSUPA+

Stopping Criteria

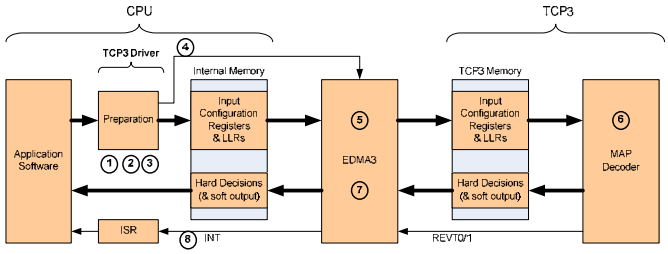
- Allows processing delay reduction.
- Three types of terminations:
  - **Iteration count** (a min and max number of iterations can be user specified).
  - **SNR Threshold** (SNR estimate of extrinsic information) – smaller threshold, fewer iterations.
  - **LTE CRC check** (after a min number of iterations and need to pass on a number of consecutive iterations) (for LTE only).

TCP3D Configuration

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Examples
- TCP3D Driver

Basic Data Flow Model

1. Formatting of the channel LLRs to match required TCP3 input format
2. Interleaver table generation (optional).
3. Input configuration parameters setup including the determination of the length and the number of block segments, and the calculation of initial beta states from tail
4. EDMA3 setup.
5. Data transfer using EDMA3 from DSP to TCP3 via a 128-bit VBUS. The transfer includes configuration registers, systematic and parity LLRs, and optionally interleaver table.
6. TCP3 decodes the code block.
7. TCP3 issues a receive event, REVT0/1, to EDMA3 causing the transfer of the decoded block from TCP3 to DSP.
8. EDMA3 generates the interrupt to DSP to signal that the decoding process is finished.



TCP3D Input LLR Formats

- On the DSP side soft bits are stored in lower 6 bits of the 8-bit words. The range of values is from -32 to 31 in Q0 format.
- During the EDMA transfer, only the lower 6 bits will be copied to TCP3 memory. It is user's responsibility to saturate the LLR values.

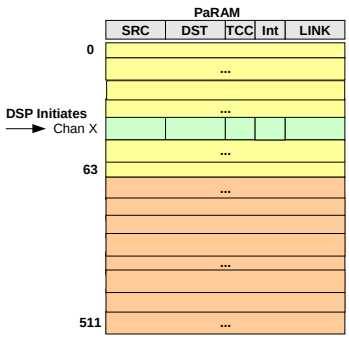
Input LLR interpretation	Integer range (Max-Log-MAP)		Fractional range (Max-star)					
			Q1		Q2		Q3	
	Binary	Decimal	Binary	Decimal	Binary	Decimal	Binary	Decimal
Most reliable "1"	011111	31	01111.1	15.5	0111.11	7.75	011.111	3.825
	011110	30	01111.0	15.0	0111.10	7.50	011.110	3.750
• • •	• • •		• • •	• • •	• • •	• • •	• • •	• • •
Least reliable "1"	000001	1	00000.1	0.5	0000.01	0.25	000.001	0.125
Neutral	000000	0	00000.0	0.0	0000.00	0.00	000.000	0.000
Least reliable "0"	111111	-1	11111.1	-0.5	1111.11	-0.25	111.111	-0.125
• • •	• • •		• • •	• • •	• • •	• • •	• • •	• • •
	100001	-31	10000.1	-15.5	1000.01	-7.75	100.001	-3.825
Most reliable "0"	100000	-32	10000.0	-16.0	1000.00	-8.00	100.000	-4.000

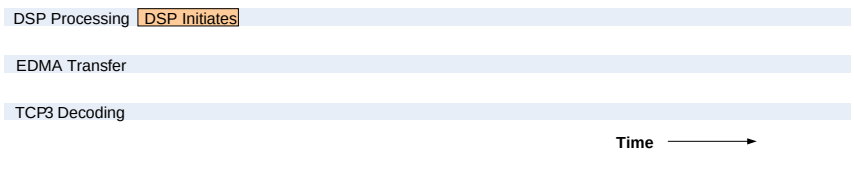
EDMA Setup Examples

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Examples
- TCP3D Driver

Single Block Decode w/ Optional Data

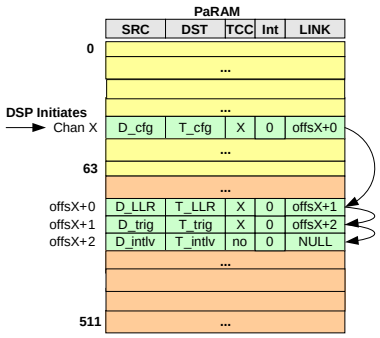
- DSP initiates process by setting the bit corresponding to EDMA channel X in the EDMA controller event set register.

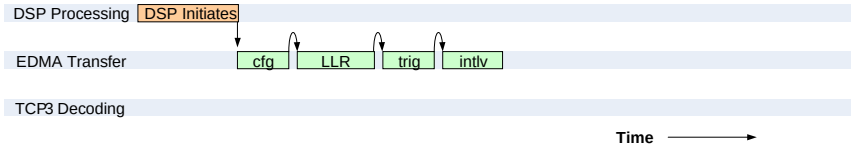




Single Block Decode w/ Optional Data

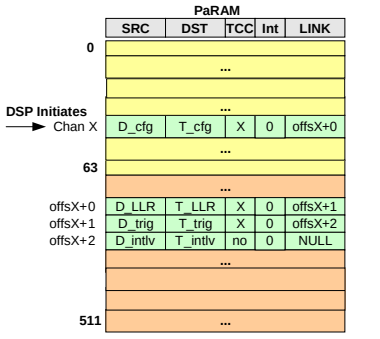
- DSP initiates process by setting the bit corresponding to EDMA channel X in the EDMA controller event set register.
- Using linking and self chaining approach, four chunks of data are transferred from DSP to TCP3D.
- The last transfer, the interleaver data, is linked to NULL.

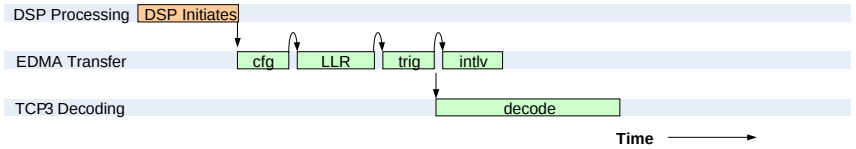




Single Block Decode w/ Optional Data

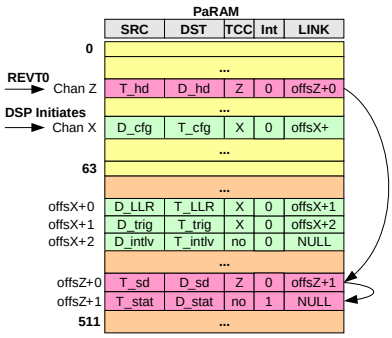
- DSP initiates process by setting the bit corresponding to EDMA channel X in the EDMA controller event set register.
- Using linking and self chaining approach, four chunks of data are transferred from DSP to TCP3D.
- The last transfer, the interleaver data, is linked to NULL.
- After receiving trigger register, TCP3D starts decoding the first half of the first iteration and waits for interleaver data.

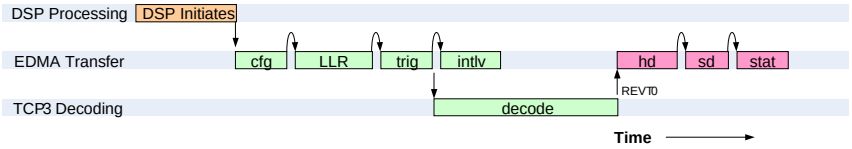




Single Block Decode w/ Optional Data

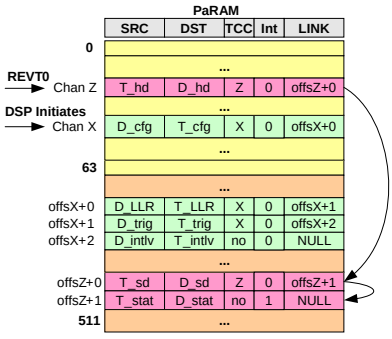
- DSP initiates process by setting the bit corresponding to EDMA channel X in the EDMA controller event set register.
- Using linking and self chaining approach, four chunks of data are transferred from DSP to TCP3D.
- The last transfer, the interleaver data, is linked to NULL.
- After receiving trigger register, TCP3D starts decoding the first half of the first iteration and waits for interleaver data.
- Once decoding is done, TCP3D sends REVT0 and sets the bit in the EDMA CC event register, which initiates the transfer of the results, hard decisions, soft decisions, and status register using linking and self chaining approach.

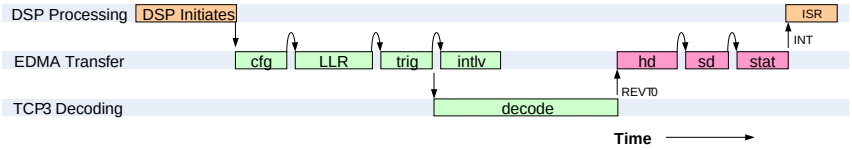




Single Block Decode w/ Optional Data

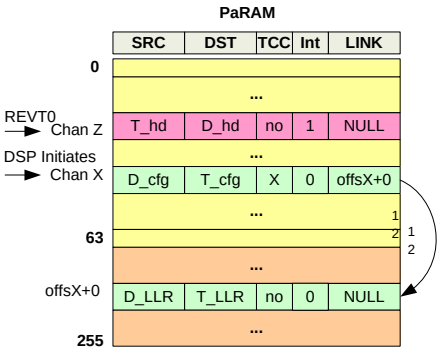
- DSP initiates process by setting the bit corresponding to EDMA channel X in the EDMA controller event set register.
- Using linking and self chaining approach, four chunks of data are transferred from DSP to TCP3D.
- The last transfer, the interleaver data, is linked to NULL.
- After receiving trigger register, TCP3D starts decoding the first half of the first iteration and waits for interleaver data.
- Once decoding is done, TCP3D sends REVT0 and sets the bit in the EDMA CC event register, which initiates the transfer of the results, hard decisions, soft decisions, and status register using linking and self chaining approach.
- After the last transfer is completed, EDMA generates an interrupt to DSP to indicate that decoding is completed.
- The transfer of the LLRs can be either 2D or 3D transfer.

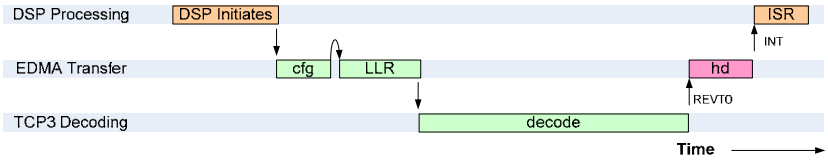




Single Block Decode w/ Necessary Data

- The decoding process can be reduced to only three transfers when the follow criteria are met:
  - TCP3D auto-trigger option is used, which automatically starts decoding once the LLRs are transferred. Thus, the trigger register does not need to be sent.
  - TCP3D is configured to generate the interleaver table internally.
  - The application only needs the hard decisions (not the soft decisions).

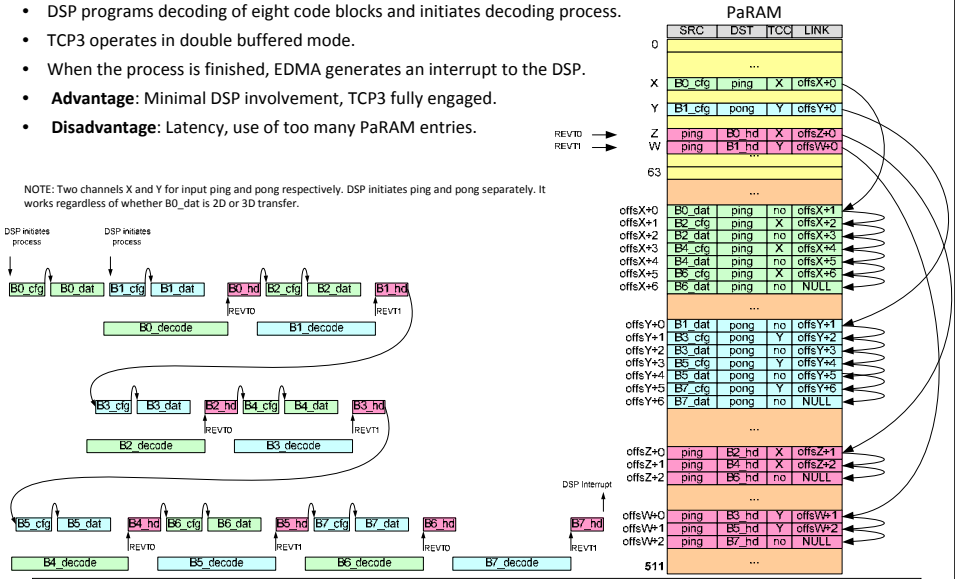




Decoding Process for Eight Blocks

- DSP programs decoding of eight code blocks and initiates decoding process.
- TCP3 operates in double buffered mode.
- When the process is finished, EDMA generates an interrupt to the DSP.
- **Advantage**: Minimal DSP involvement, TCP3 fully engaged.
- **Disadvantage**: Latency, use of too many PaRAM entries.

NOTE: Two channels X and Y for input ping and pong respectively. DSP initiates ping and pong separately. It works regardless of whether B0_dat is 2D or 3D transfer.



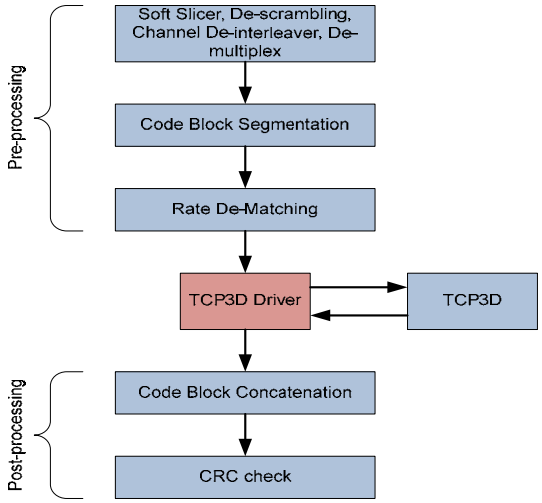
TCP3D Driver

- TCP3D Architecture
- TCP3D Throughput
- TCP3D Processing Modes
- TCP3D Configuration
- EDMA Setup Examples
- TCP3D Driver

Purpose of the TCP3D Driver

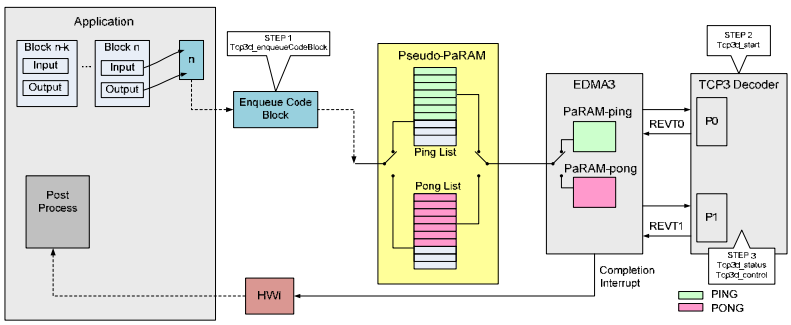
- To minimize the idle times of the TCP3D engine
- To achieve maximum throughput from the TCP3 decoder
- To minimize the CorePac intervention, minimum context switching
- Support multiple TCP3D instances to utilize each peripheral instance independently
- Support multicore input and multicore output destinations

TCP3D Driver in LTE Uplink Bit Processing Chain

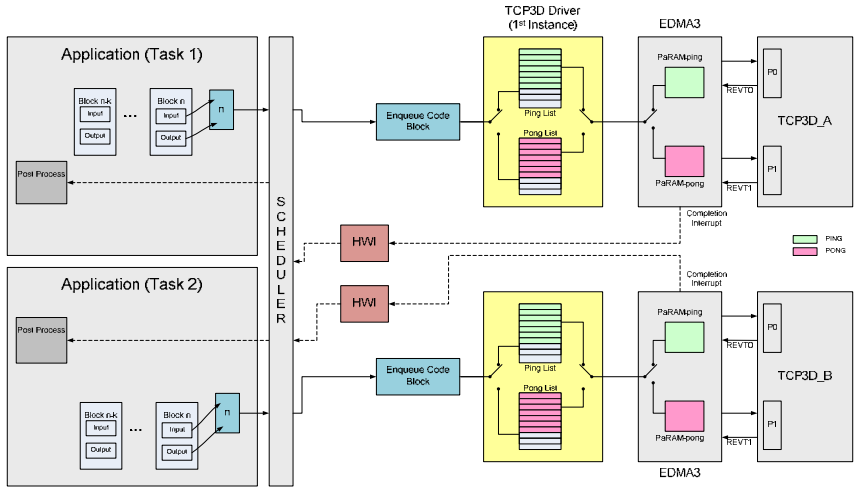


TCP3D Driver Functions

- After pre-processing tasks are complete, the soft bits (LLR) are ready for TCP3D.
- The TCP3D driver performs the following functions:
  - Enqueues code blocks to TCP3D input pseudo PaRAM list in L2 which holds the actual PaRAMs prepared.
  - TCP3D driver copies pseudo PaRAM sets to EDMA PaRAM space and executes them.
  - Starts/restart TCP3D if it is not running. This can be done by polling the stop flag or by using ISR function.
- After decoding any specified code block, the TCP3D driver can generate a complete interrupt to trigger the post-processing tasks.



Two TCP3 Decoder Instances



TCP3D Driver API

- Initialization
  - TCP3D_DRV_init()
- Run-time
  - TCP3D_DRV_enqueCodeBlock()
  - TCP3D_DRV_start()
- Utility
  - TCP3D_DRV_prepConfigRegs()

Summary

- For more information, please refer to the TCP3D User Guide.