**IN THE UNITED STATES DISTRICT COURT**
**FOR THE EASTERN DISTRICT OF TEXAS**
**SHERMAN DIVISION**

| | |
|---|---|
| **TURBOCODE LLC,**<br><br>Plaintiff,<br>v.<br><br>**ADVANTECH CO., LTD.,**<br><br>Defendant. | C.A. No. 4:24-cv-524<br><br>**JURY TRIAL DEMANDED**<br><br>**PATENT CASE** |

**ORIGINAL COMPLAINT FOR PATENT INFRINGEMENT**

Plaintiff TurboCode LLC files this Original Complaint for Patent Infringement against Advantech Co. Ltd, and would respectfully show the Court as follows:

## I. THE PARTIES

1. Plaintiff TurboCode LLC ("TurboCode" or "Plaintiff") is a Texas limited liability company with its address at 6000 Shepherd Mountain Cove, Suite #1604, Austin Texas 78730.

2. On information and belief, Defendant Advantech Co., Ltd. ("Defendant") is a corporation organized and existing under the laws of Taiwan with a place of business at No.1, Alley 20, Lane 26, Rueiguang Road, Neihu District, Taipei 114519, Taiwan. Defendant has a registered agent at CT Corporation System 1999 Bryan St., Ste 900, Dallas, TX 75201.

## II. JURISDICTION AND VENUE

3. This action arises under the patent laws of the United States, Title 35 of the United States Code. This Court has subject matter jurisdiction of such action under 28 U.S.C. §§ 1331 and 1338(a).

4. On information and belief, Defendant is subject to this Court's specific and general personal jurisdiction, pursuant to due process and the Texas Long-Arm Statute, due at least to its business in this forum, including at least a portion of the infringements alleged herein. Defendant

is subject to this Court's general personal jurisdiction pursuant to due process and/or the Texas Long Arm Statute, Tex. Civ. Prac. & Rem. Code § 17.042, due at least to its substantial business conducted in this District, including: (i) having solicited and transacted business in Texas, including benefits directly related to the instant patent infringement causes of action set forth herein; (ii) having placed its products and services into the stream of commerce throughout the United States and having been actively engaged in transacting business in Texas and in this District, and (iii) having committed the complained of tortious acts in Texas and in this District. Advantech has sold computer servers, modular computers, and network products to related Advantech entities and to third parties in the United States, and has shipped infringing computer servers, modular computers, and network products to the United States.

5. Without limitation, on information and belief, within this state, Defendant used the patented invention thereby committing, and continuing to commit, acts of patent infringement alleged herein. In addition, on information and belief, Defendant derived revenues from its infringing acts occurring within Texas. Further, on information and belief, Defendant is subject to the Court's general jurisdiction, including from regularly doing or soliciting business, engaging in other persistent courses of conduct, and deriving substantial revenue from goods and services provided to persons or entities in Texas. Further, on information and belief, Defendant is subject to the Court's personal jurisdiction at least due to its sale of products and/or services within Texas. Defendant committed such purposeful acts and/or transactions in Texas such that it reasonably should know and expect that it could be haled into this Court as a consequence of such activity.

6. Venue is proper in this district under 28 U.S.C. § 1400(b). On information and belief, from and within this District Defendant committed at least a portion of the infringements at issue in this case. Furthermore, venue is proper as to Defendant because 28 U.S.C. § 1391(c)(3) provides that "a defendant not resident in the United States may be sued in any judicial district, and

the joinder of such a defendant shall be disregarded in determining where the action may be brought with respect to other defendants."

7. For these reasons, personal jurisdiction exists and venue is proper in this Court under 28 U.S.C. § 1400(b).

## III. COUNT I
## (PATENT INFRINGEMENT OF UNITED STATES PATENT NO. 6,813,742)

8. Plaintiff incorporates the above paragraphs herein by reference.

9. On November 2, 2004, the USPTO duly and legally issued U.S. Patent No. 6,813,742 ("the '742 Patent" or "Patent-in-Suit"), entitled "High Speed Turbo Codes Decoder for 3G Using Pipelined SISO Log-Map Decoders Architecture." The '742 patent was the subject of a reexamination request filed on July 13, 2006. An *Ex Parte* Reexamination Certificate was issued for the '742 patent on February 10, 2009. A true and correct copy of the '742 Patent with its *Ex Parte* Reexamination Certificate is attached hereto as Exhibit 1.

10. TurboCode is the assignee of all right, title, and interest in the '742 patent, including all rights to enforce and prosecute actions for infringement and to collect damages for all relevant times against infringers of the '742 Patent. Accordingly, TurboCode possesses the exclusive right and standing to prosecute the present action for infringement of the '742 Patent by Defendant.

11. This case generally relates to decoder architectures and processes for receiving and decoding data in communications devices.

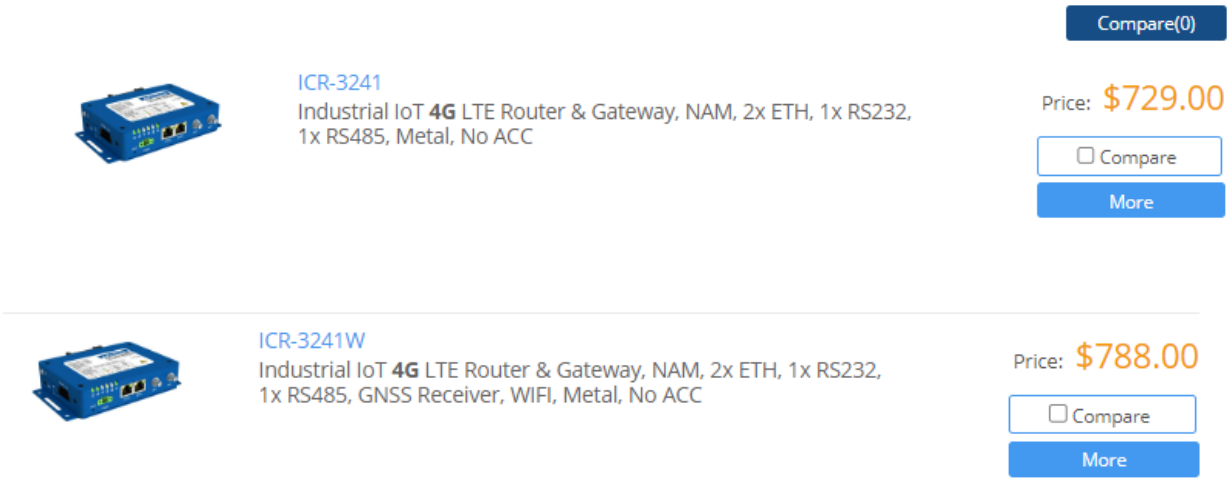
12. **Direct Infringement.** Upon information and belief, Defendant directly infringed claim 6 of the '742 Patent in Texas, and elsewhere in the United States, by performing actions comprising using or performing the claimed method of iteratively decoding a plurality of sequences of received baseband signals by using and/or testing the products, devices, systems, and components of systems that comply with the 3G and/or 4G/LTE standards as disclosed in the 3rd

Generation Partnership Project ("3GPP") Standard Specifications (releases 8-11) governing cellular wireless communications including, but not limited to, ICR-3241, ICR-3241W, TREK-773R-1, AIM-65, AIM-68CT-S4, AIM68CT-S2, PWS-870-9S53, PWS-870-9S54, FWA-3034, FWA-3034R-00A1R, ESRP-PCS-ECU1051, and TREK-733L-S ("Accused Instrumentalies").

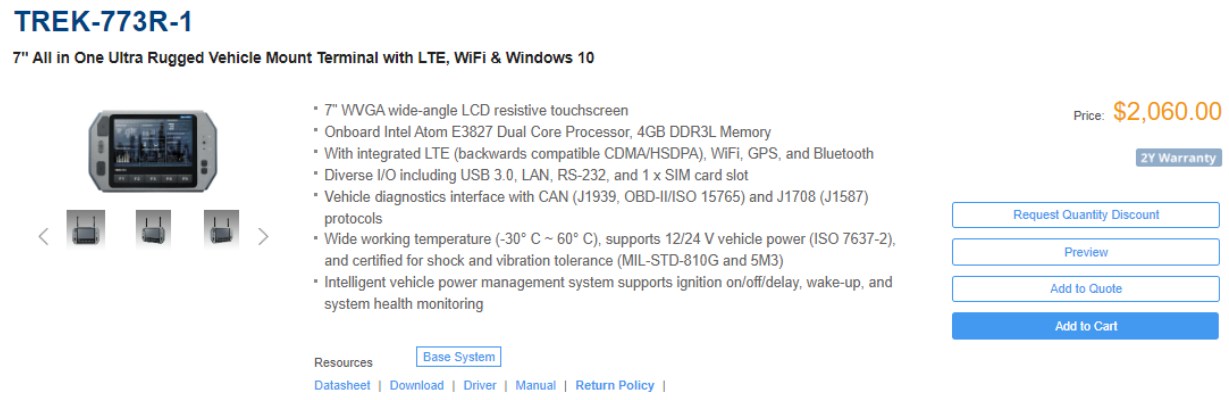
13. Claim 6 of the '742 Patent *Ex Parte* Reexamination Certificate states:

> A method of iteratively decoding a plurality of sequences of received baseband signals, the method comprising:
>
> providing an input buffer comprising at least three shift registers, for receiving an input signal and generating first, second, and third shifted input signals;
>
> providing first and second soft decision decoders serially coupled in a circular circuit, wherein each decoder processes soft decision from the preceding decoder output data, and wherein the first decoder further receives the first and second shifted input signals from the input buffer and the second decoder further receives the third shifted input signal from the input buffer;
>
> providing at least one memory module coupled to an output of each of the first and second soft decision decoders, wherein the output of the memory module associated with the second soft decision decoder is fed back as an input of the first soft decision decoder;
>
> processing systematic information data and extrinsic information data using the maximum a posteriori (AP) probability algorithm, and/or logarithm approximation algorithm;
>
> generating soft decision based on the maximum a posteriori (MAP) probability algorithm, and/or logarithm approximation algorithm;
>
> weighing and storing soft decision information into the corresponding memory module;
>
> performing, for a predetermined number of times, iterative decoding from the first to the last of multiple decoders, wherein an output from the last soft decision decoder is fed back as an input to the first soft decision decoder, then from the first to the second decoders, and propagate to the last decoder in a circular circuit.

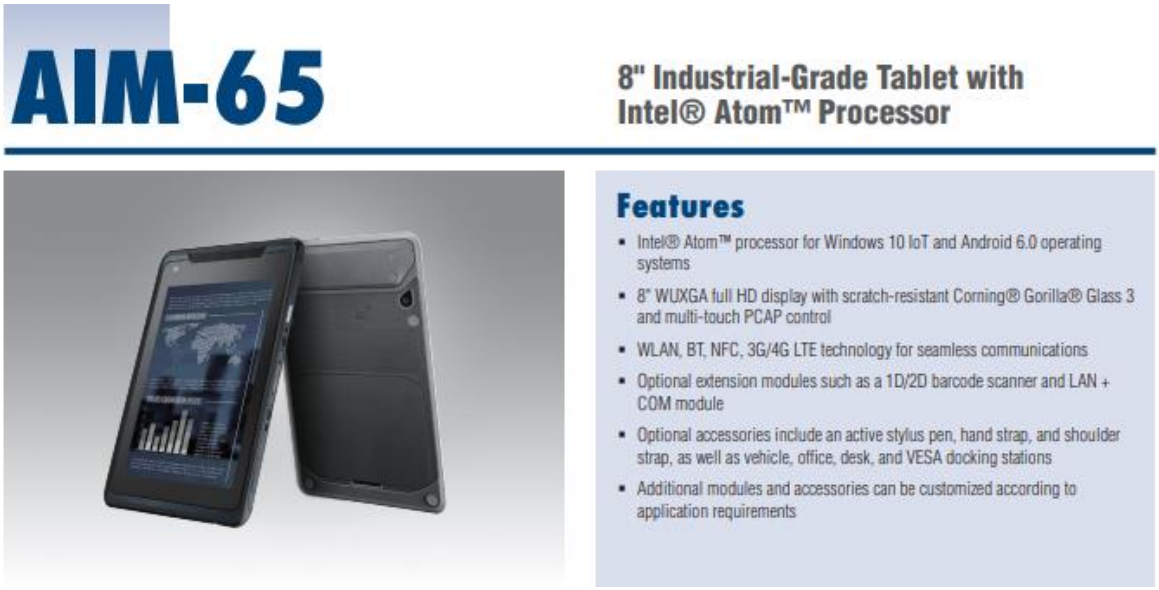
14. The Accused Instrumentalities provided or performed a method of iteratively decoding a plurality of sequences of received baseband signals, as shown below by their compliance with the 3G and/or 4G/LTE standards disclosed in the 3GPP Standard Specifications:



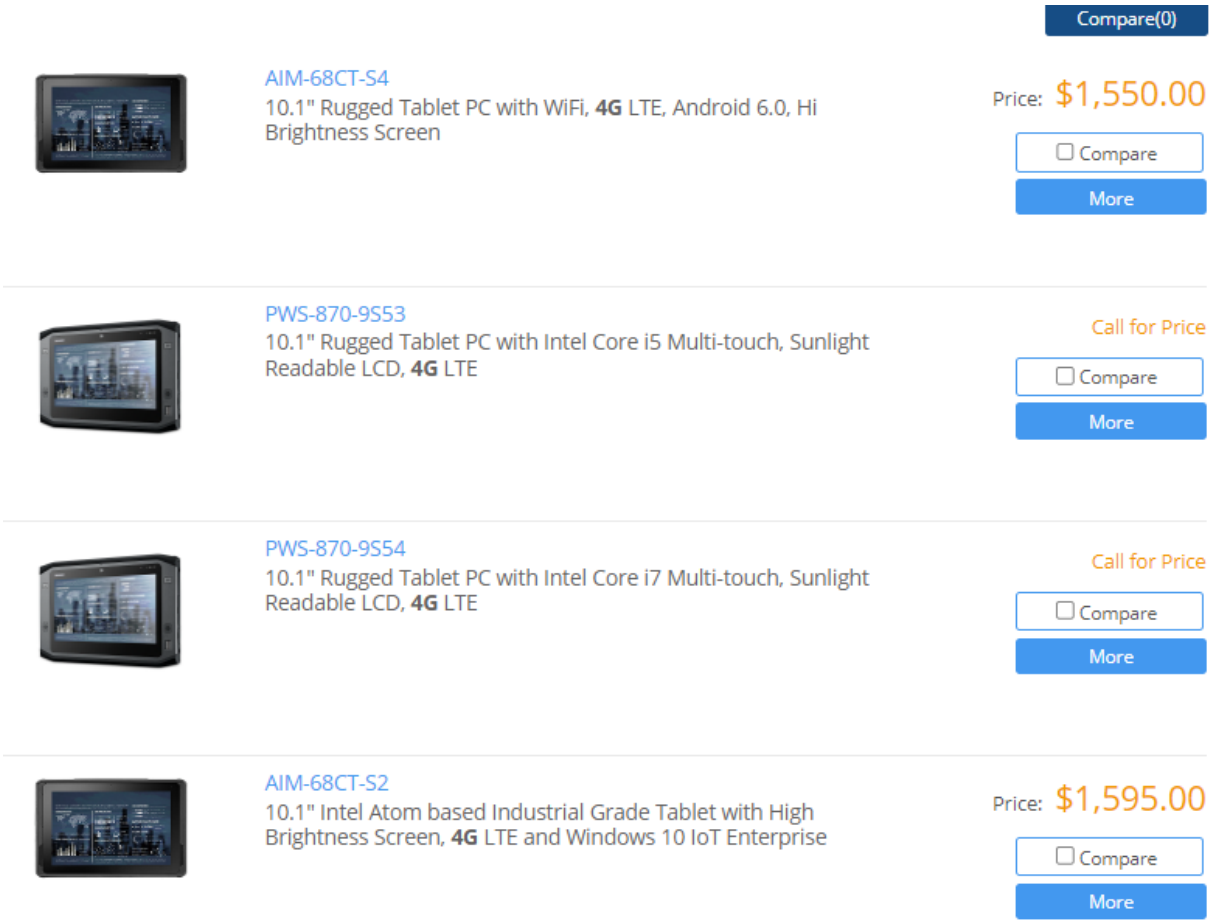

(*E.g.*, https://buy.advantech.com/Search.aspx?skey=4g).



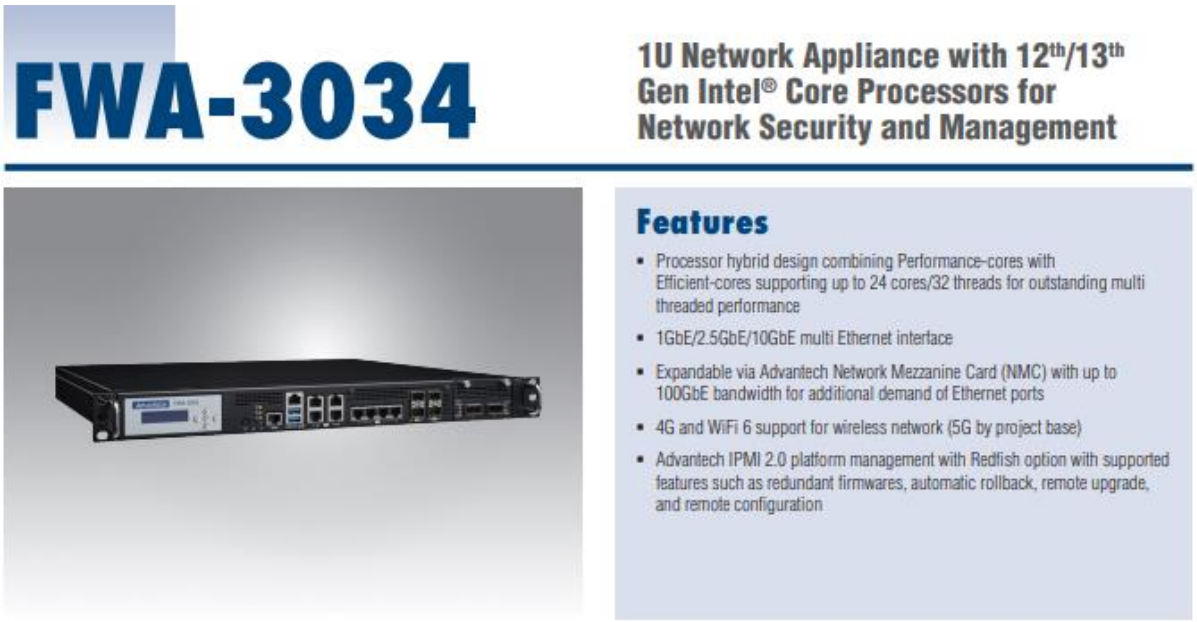

(*E.g.*, https://buy.advantech.com/Industry-Solutions/Vehicle-PC-Solutions-Vehicle-Mount-Computers/TREK-773R-1/system-22619.htm).



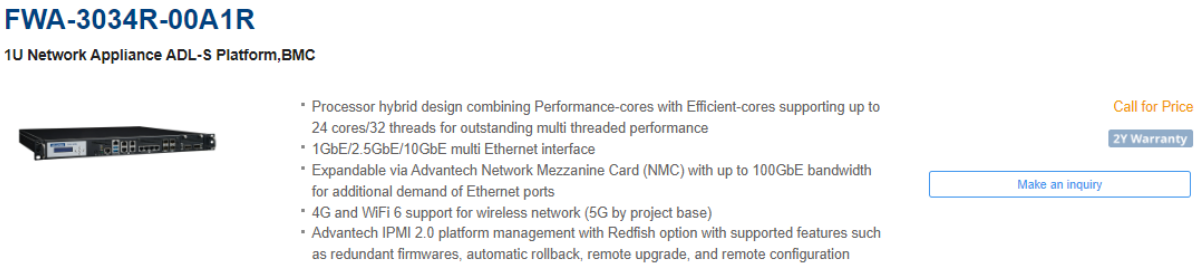

(*E.g.*, https://buy.advantech.com/resource/ProductResource/AUS/964f57bf-996e-4af9-bc95-cdb53b021b09.pdf?utm_source=eStore).



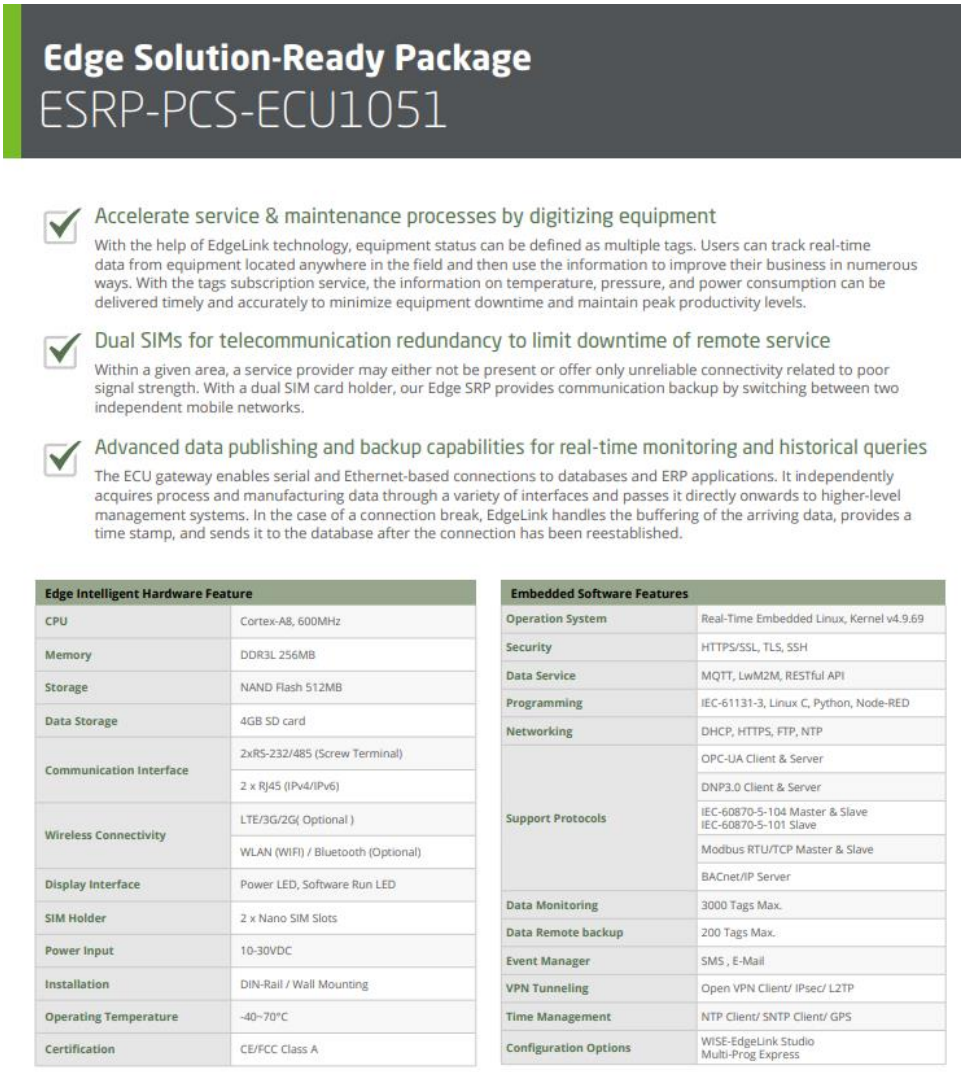

(*E.g.*, https://buy.advantech.com/Search.aspx?skey=4g).




(*E.g.*, https://advdownload.advantech.com/productfile/PIS/FWA-3034/file/FWA-3034_DS(032924)20240402133904.pdf?utm_source=eStore).




(*E.g.*, https://buy.advantech.com/Rackmount-Systems/Network-Appliance-Medium/model-FWA-3034R-00A1R.htm).



## Edge Solution-Ready Package
# ESRP-PCS-ECU1051

### Accelerate service & maintenance processes by digitizing equipment

With the help of EdgeLink technology, equipment status can be defined as multiple tags. Users can track real-time data from equipment located anywhere in the field and then use the information to improve their business in numerous ways. With the tags subscription service, the information on temperature, pressure, and power consumption can be delivered timely and accurately to minimize equipment downtime and maintain peak productivity levels.

### Dual SIMs for telecommunication redundancy to limit downtime of remote service

Within a given area, a service provider may either not be present or offer only unreliable connectivity related to poor signal strength. With a dual SIM card holder, our Edge SRP provides communication backup by switching between two independent mobile networks.

### Advanced data publishing and backup capabilities for real-time monitoring and historical queries

The ECU gateway enables serial and Ethernet-based connections to databases and ERP applications. It independently acquires process and manufacturing data through a variety of interfaces and passes it directly onwards to higher-level management systems. In the case of a connection break, EdgeLink handles the buffering of the arriving data, provides a time stamp, and sends it to the database after the connection has been reestablished.

| Edge Intelligent Hardware Feature | |
|---|---|
| CPU | Cortex-A8, 600MHz |
| Memory | DDR3L 256MB |
| Storage | NAND Flash 512MB |
| Data Storage | 4GB SD card |
| Communication Interface | 2xRS-232/485 (Screw Terminal) |
| | 2 x RJ45 (IPv4/IPv6) |
| Wireless Connectivity | LTE/3G/2G( Optional ) |
| | WLAN (WIFI) / Bluetooth (Optional) |
| Display Interface | Power LED, Software Run LED |
| SIM Holder | 2 x Nano SIM Slots |
| Power Input | 10-30VDC |
| Installation | DIN-Rail / Wall Mounting |
| Operating Temperature | -40~70°C |
| Certification | CE/FCC Class A |

| Embedded Software Features | |
|---|---|
| Operation System | Real-Time Embedded Linux, Kernel v4.9.69 |
| Security | HTTPS/SSL, TLS, SSH |
| Data Service | MQTT, LwM2M, RESTful API |
| Programming | IEC-61131-3, Linux C, Python, Node-RED |
| Networking | DHCP, HTTPS, FTP, NTP |
| Support Protocols | OPC-UA Client & Server |
| | DNP3.0 Client & Server |
| | IEC-60870-5-104 Master & Slave<br>IEC-60870-5-101 Slave |
| | Modbus RTU/TCP Master & Slave |
| | BACnet/IP Server |
| Data Monitoring | 3000 Tags Max. |
| Data Remote backup | 200 Tags Max. |
| Event Manager | SMS , E-Mail |
| VPN Tunneling | Open VPN Client/ IPsec/ L2TP |
| Time Management | NTP Client/ SNTP Client/ GPS |
| Configuration Options | WISE-EdgeLink Studio<br>Multi-Prog Express |

(*E.g.*, https://advdownload.advantech.com/productfile/PIS/ESRP-PCS-ECU1051/file/ESRP_ECU1051_en-en20221020162439.pdf?utm_source=eStore).

**TREK-733L-S** Phase Out

**7" All-in-One In-Vehicle Computer with Android Support, 4G LTE, WLAN**

- 7" WSVGA Touchscreen LCD with capacitive touch
- NXP Dual Core i.MX6DL ARM SOC, 1GB DRAM, Supports Android 4.4.2
- Built-in 4G LTE, WLAN, Bluetooth and GNSS. Antenna support via SMA
- Supports 12/24V vehicle power and shock/vibration tolerant (MIL STD-810G)
- Built-in battery pack & UPS with emergency alert notifications
- Features vehicle 12/24V power mngmt. system
- One analog video input to dedicated video processor & multi isolation Digital I/O
- Wide temp -20°C ~ 70°C

(*E.g.*, https://buy.advantech.com/7-Freescale-i-MX6D-RISC-All-in-One-Android-Mobile-Data-Terminal/TREK-733L-S/TREK-733L-S/system-21787.htm).

15. For example, each of the Accused Instrumentalities performed iterative decoding using at least the BCJR algorithm.

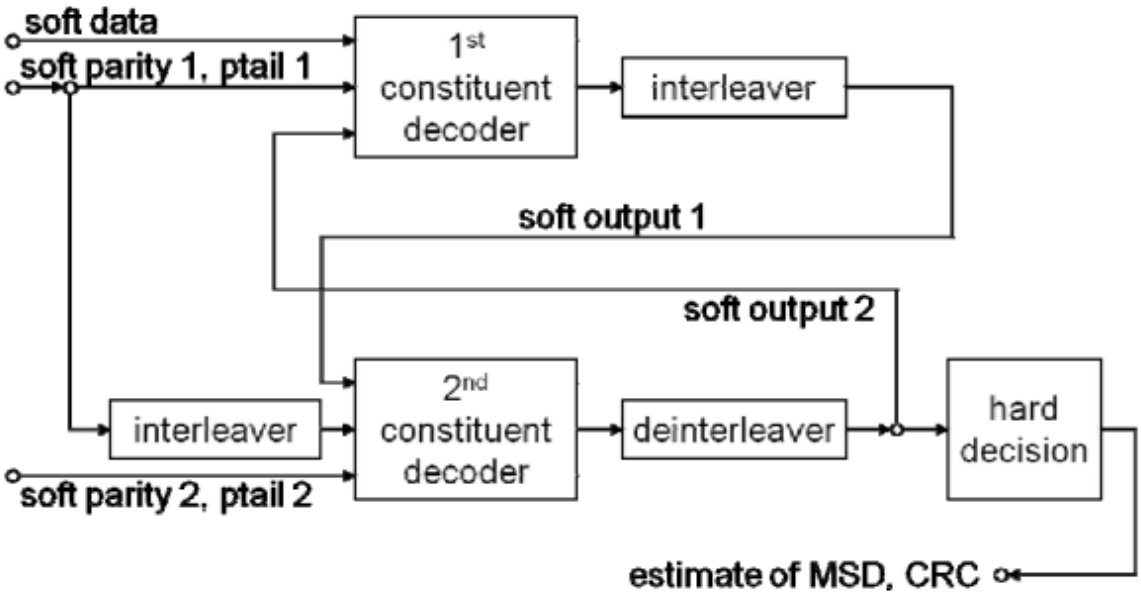



**Figure 18: Turbo decoder**

(*See* 3GPP TS 26.268 at 21 (v. 11), 17 (v. 8).).

*See* Mansour et al., "VLSI Architectures for SISO-APP Decoders," IEEE Transactions On Very Large Scale Integration ("VLSI") Systems, Vol. 11, No. 4 (Aug. 2003), at 627, *available at* http://shanbhag.ece.illinois.edu/publications/mansr-tvlsi-2003-2.pdf: "The BCJR algorithm was generalized in [S. Benedetto *et al.*, "A Soft-Input Soft-Output Maximum a posteriori (Map) Module to Decode Parallel and Serial Concatenated Codes," JPL, TDA Progress Report 42-127, Nov. 1996] into a soft-input soft-output a posteriori probability (SISO- APP) algorithm to be used as a building block for iterative decoding in code networks with generic topologies…." *See also Cheng et. al.* "A 0.077 to 0.168 nj/bit/iteration Scalable 3GPP LTE Turbo Decoder with an Adaptive Sub-Block Parallel Scheme and an Embedded DVFS Engine," 2010 IEEE Custom Integrated Circuits Conference (CICC) (19-22 Sept. 2010), at 3, *available at* https://dspace.mit.edu/bitstream/handle/1721.1/72198/Chandrakasan-a%200.077%20to%200.168.pdf?sequence=1&isAllowed=y (citation omitted): "Figure 3 shows the system architecture. The blocks in the dashed box handle the turbo decoding operations, and those outside the dashed box belong to the DVFS scheme. Turbo decoding is an iterative process with several turbo iterations. Each turbo iteration comprises two soft-in, soft-out (SISO) decoding processes using BCJR algorithm with the first one performed on the input code block in the original order and the second one in an order generated by the interleaver block." *See also* "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); eCall data transfer; In-band modem solution; ANSI-C reference code (3GPP TS 26.268 version 11.0.0 Release 11)," at 14, *available at* https://www.etsi.org/deliver/etsi_ts/126200_126299/126268/11.00.00_60/ts_126268v110000p.pdf ("3GPP TS 26.268 v. 11"); "Digital cellular telecommunications system (Phase 2+); Universal Mobile Telecommunications System (UMTS); eCall data transfer; In-band modem solution;

ANSI-C reference code (3GPP TS 26.268 version 8.0.0 Release 8)," at 17, *available at* https://www.etsi.org/deliver/etsi_ts/126200_126299/126268/08.00.00_60/ts_126268v080000p.pdf ("3GPP TS 26.268 v. 8"):

| Type/Constant | Dimension | Description |
|---|---|---|
| /* Synchronization */ | | |
| const Int16 wakeupSin500 | [16] | sine waveform at 500 Hz |
| const Int16 wakeupCos500 | [16] | cosine waveform at 500 Hz |
| const Int16 wakeupSin800 | [10] | sine waveform at 800 Hz |
| const Int16 wakeupCos800 | [10] | cosine waveform at 800 Hz |

*See also id.* at 20 (v. 11), 16-17 (v. 8) and source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268:

```
/*============================================================================*/
/* PSAP FUNCTION: PsapReceiver                                                */
/*----------------------------------------------------------------------------*/
/* Description: PSAP receiver function (decoding is done outside)             */
/*                                                                            */
/* In:     const ModState* ms        -> modulator struct                      */
/*         const Int16*    pcm       -> input data for demodulation           */
/* Out:    IntLLR*         softBits  <- demodulated soft bit sequence         */
/*----------------------------------------------------------------------------*/
void PsapReceiver(const ModState *ms, const Int16 *pcm, IntLLR *softBits)


/*============================================================================*/
/* PSAP FUNCTION: SymbolDemod                                                 */
/*----------------------------------------------------------------------------*/
/* Description: symbol demodulator                                            */
/*                                                                            */
/* In:     const ModState* ms        -> modulator struct                      */
/*         const Int16*    mPulse    -> received pulse train                  */
/* Out:    IntLLR*         softBits  <- demodulated soft bit sequence         */
/*----------------------------------------------------------------------------*/
void SymbolDemod(const ModState *ms, const Int16 *mPulse, IntLLR *softBits)
```

(*E.g.*, ecall_fec.c line 163 *Bool FecDecode(const IntLLR *in, Int16 rv, Ord1 *out)*).

```
/*============================================================================*/
/* DECODER FUNCTION: FecDecode                                                */
/*----------------------------------------------------------------------------*/
/* Description: decoding to find the MSD                                      */
/*                                                                            */
/* In:     const IntLLR* in   -> received soft bits                           */
/*         Int16         rv   -> redundancy version                           */
/* Out:    Ord1*         out  <- decoded MSD in binary representation         */
/* Return: Bool                <- result of CRC check                         */
/*----------------------------------------------------------------------------*/
Bool FecDecode(const IntLLR *in, Int16 rv, Ord1 *out)
```

(*E.g.*, ecall_fec.c line 240 */*iterative decoding*/ for (i = 0; i < FEC_ITERATIONS; i++)*).[1]

*See also* May et al., "A 150Mbit/s 3GPP LTE Turbo code decoder," 2010 Design, Automation & Test in Europe Conference & Exhibition (March 8-12, 2010), *available at* https://ieeexplore.ieee.org/document/5457035/authors#authors: "3GPP long term evolution (LTE) enhances the wireless communication standards UMTS and HSDPA towards higher throughput. A throughput of 150 Mbit/s is specified for LTE using 2×2 MIMO. For this, highly punctured Turbo codes with rates up to 0.95 are used for channel coding, which is a big challenge for decoder design. This paper investigates efficient decoder architectures for highly punctured LTE Turbo codes. We present a 150 Mbit/s 3GPP LTE Turbo code decoder, which is part of an industrial SDR multi-standard baseband processor chip."

16. Additionally, or alternatively, the relevant standards bodies, such as 3GPP, provided specifications that specify virtually all aspects of turbo encoders (*i.e.*, turbo-code transmitters), turbo decoders (*i.e.*, turbo-code receivers) and specify minimum performance requirements for the receivers and decoders (as well as encoders and transmitters) so that arbitrary transmitter-receiver pairs can communicate seamlessly. *See, e.g.*, ROHDE & SCHWARZ, "Radio fundamentals for cellular networks White paper," 19 (Jan. 2019), *available at* https://www.elektronikfokus.dk/wp-content/uploads/sites/5/WhitePaper_Radio-fundamentals-for-cellular-networks_wp_en_5216-0467-52_v0201.pdf.

17. On information and belief, each of the Accused Instrumentalities processed received baseband digital signals in an iterative manner. It was not commercially-feasible to

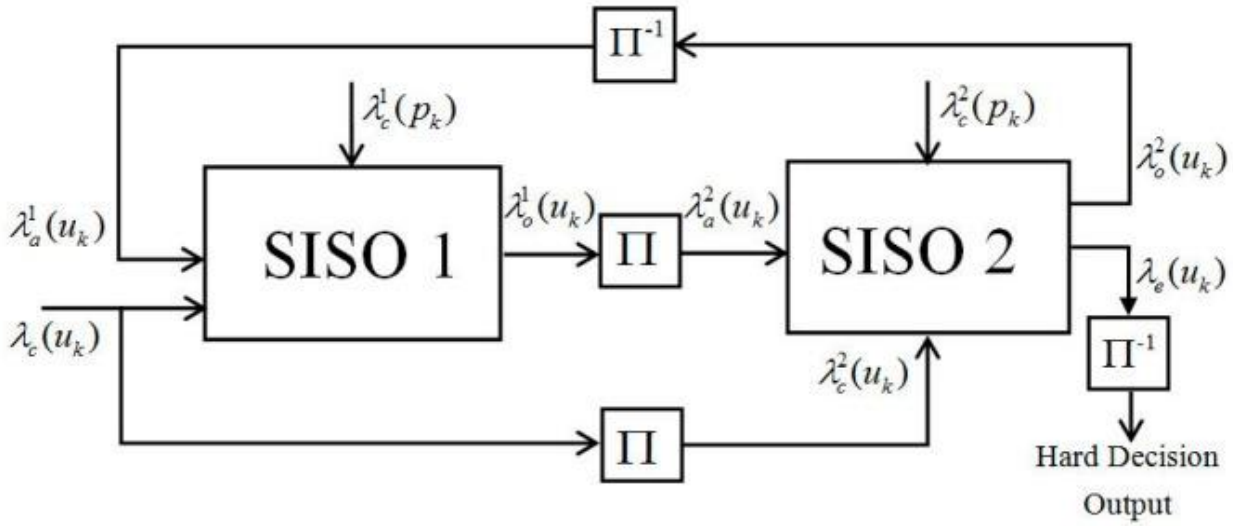
[1] All citations to source code refer to the source code release accompanying 3GPP TS 26.268, available at https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=1446." Specific line citations for this and other source code are to the version accompanying Release 11, but substantially identical code is present in the version accompanying Release 8, available from the same hyperlink.

implement turbo decoders in a non-iterative manner. No known commercial turbo decoder implementation used a pipeline of a SISO pair (with interleave/deinterleave operations) as replicated hardware blocks to fully replace iterations, and it is universally known in industry and academia that decoding of 3G/4G LTE turbo codes was explicitly and inherently iterative by both requirement and design. *See, e.g.*, Dejan Spasov, "Decoding of LTE Turbo Codes Initialized with the Two Recursive Convolutional Codes," 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO) (2020), *available at* https://ieeexplore.ieee.org/document/9245282 ("Turbo codes were the first error-correcting codes that demonstrated reliable communications near the channel capacity with practically feasible hardware. Due to their excellent error-correcting capability, they are part of many modern communication technologies, like 3G, 4G, LTE, etc. …. The decoding of LTE Turbo codes is iterative."); Altera Corporation, "3GPP LTE Turbo Reference Design" (Jan. 2020), *available at* https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/an/an505.pdf ("A Turbo decoder consists of two single soft-in soft-out (SISO) decoders, which work iteratively."); Shuai Shao *et al.*, "Survey of Turbo, LDPC and Polar Decoder ASIC Implementations," IEEE Communications Surveys & Tutorials, 1 (Jan. 17, 2019), *available at* https://eprints.soton.ac.uk/427712/1/Survey_of_Turbo_LDPC_and_Polar_Decoder_ASIC_Implementations.pdf ("Both the turbo and LDPC codes employ an iterative decoding process, in which each successive attempt at decoding the information block informs the next, until the process converges to a legitimate codeword.").

18. In iterative turbo decoder implementations, each iteration provides a decoding pass step. At each moment, a given error environment has an associated optimal number of decoding pass steps to produce the best results. Thus, a permanently-fixed number of decoding pass steps

incurs either wasted computation or degraded performance. (*See, e.g.,* A. Matache et al., "Stopping Rules for Turbo Decoders," TMO Progress Report 42-142 (Aug. 15, 2000), *available at* https://ipnpr.jpl.nasa.gov/progress_report/42-142/142J.pdf). Because the number of decoding pass steps can be arbitrary and dynamically varied, stopping rules (dependent on real-time computational measurements) are used to determine the number of decoding pass steps.

19. Were a non-iterative (pipeline) implementation employed, a fixed or maximum number of decoding pass steps would have to be implemented, with each decoding pass step requiring extensive hardware, and in practice yielding at almost every instant either too few stages or too many stages. Thus, non-iterative implementations suffered disadvantages of higher costs, high power consumption, and lower average performance. In practice any commercial 4G LTE turbo decoder is necessarily recursive/iterative. While lower-level pipelines were employed in many hardware implementations of MAP computations *within* SISO elements, this is different in scope and method than a (non-iterative) pipeline of *full-scale SISO pair blocks* (with interleave/deinterleave operations) of replicated hardware to replace (SISO pair) iterations. All iterative implementations of 4G LTE turbo decoders were functionally equivalent to the figures below. The top of the figures below illustrates the feedback loop that forced 3G/4G LTE turbo decoders to be iterative. The iterative loop traversed elements SISO 1, Π, SISO 2, $\Pi^{-1}$, and closed with the signal path return to the input of SISO 1.




(*E.g.*, *see* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/).

20. Upon information and belief, the Accused Instrumentalities provided an input buffer comprising at least three shift registers, for receiving an input signal and generating first, second, and third shifted input signals. The Accused Instrumentalities provided (*e.g.*, via an input buffer) input to the constituent decoders of the turbo decoder. The input buffer comprised at least three shift registers. The input buffer received an input signal, and first, second, and third shifted input signals were generated for input to a turbo decoder. The generated first, second, and third shifted input signals, shown as "soft data," "soft parity 1, ptail 1," and "soft parity 2, ptail 2," were input into the 1st constituent decoder and 2nd constituent decoder as shown below:

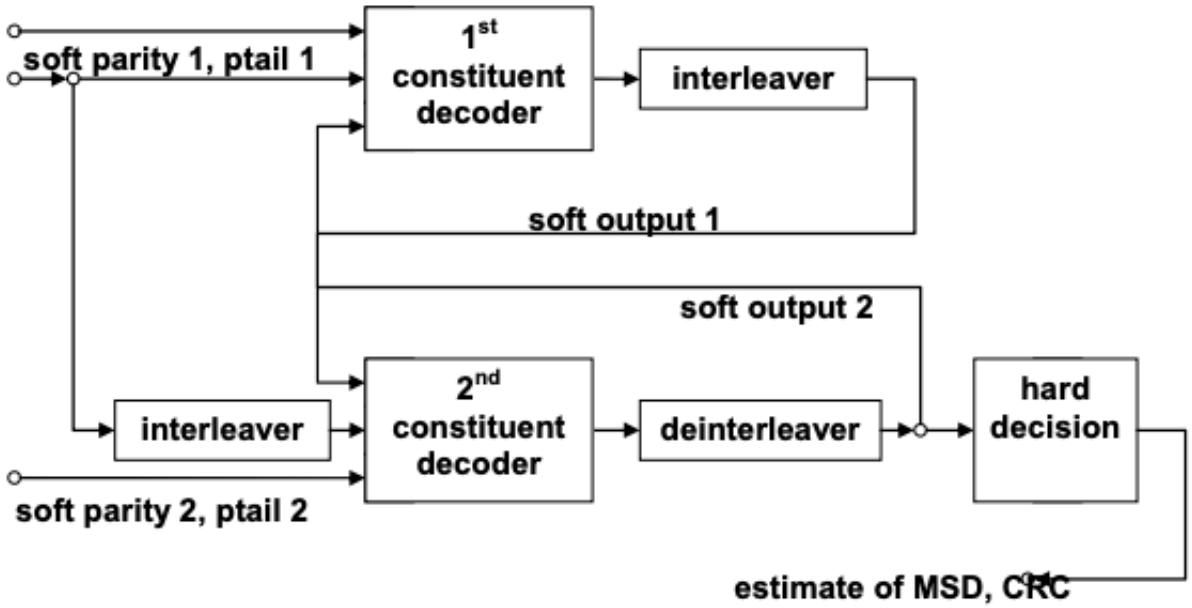



**Figure 18: Turbo decoder**

(*E.g.*, 3GPP TS 26.267, at 14 (v. 11, v. 8)).

21. Upon information and believe, in the Accused Instrumentalities, each such buffer provided three sections based on operations of a turbo encoder. For example, an input buffer is denoted as Figure 7, labeled as a "channel coded bit buffer."


**Figure 7: Channel coded bit buffer**

(*See* 3GPP TS 26.267, at 14 (v. 11, v. 8)). Note that the "soft data" of Figure 18 corresponds to the "MSD+CRC," "tail 1," and "tail 2" fields of Figure 7. Such a channel coded bit buffer could then be decoded using shifting to generate the first, second, and third shifted input signals, which were stored in registers for input into the turbo decoder: "6.1.3 Modulation: The encoded binary

data stream bits $b_i$ are grouped into symbols. Each symbol $d_j$ carries 4 bits of information and modulates one basic downlink waveform… Table 4 describes the symbol modulation mapping between symbol and the downlink waveform. The downlink waveform is derived from the basic downlink waveform $p_{DL}(n)$ by a cyclic right-shift by $k$ samples, denoted by (p → k), and multiplication with a sign $q$." (*See* 3GPP TS 26.267, at 22 (v. 11), 21 (v. 8)). *See also Cheng et. al.* "A 0.077 to 0.168 nj/bit/iteration Scalable 3GPP LTE Turbo Decoder with an Adaptive Sub-Block Parallel Scheme and an Embedded DVFS Engine," 2010 IEEE Custom Integrated Circuits Conference (CICC) (19-22 Sept. 2010), at Fig. 3, *available at* https://dspace.mit.edu/bitstream/handle/1721.1/72198/Chandrakasan-a%200.077%20to%200.168.pdf?sequence=1&isAllowed=y:

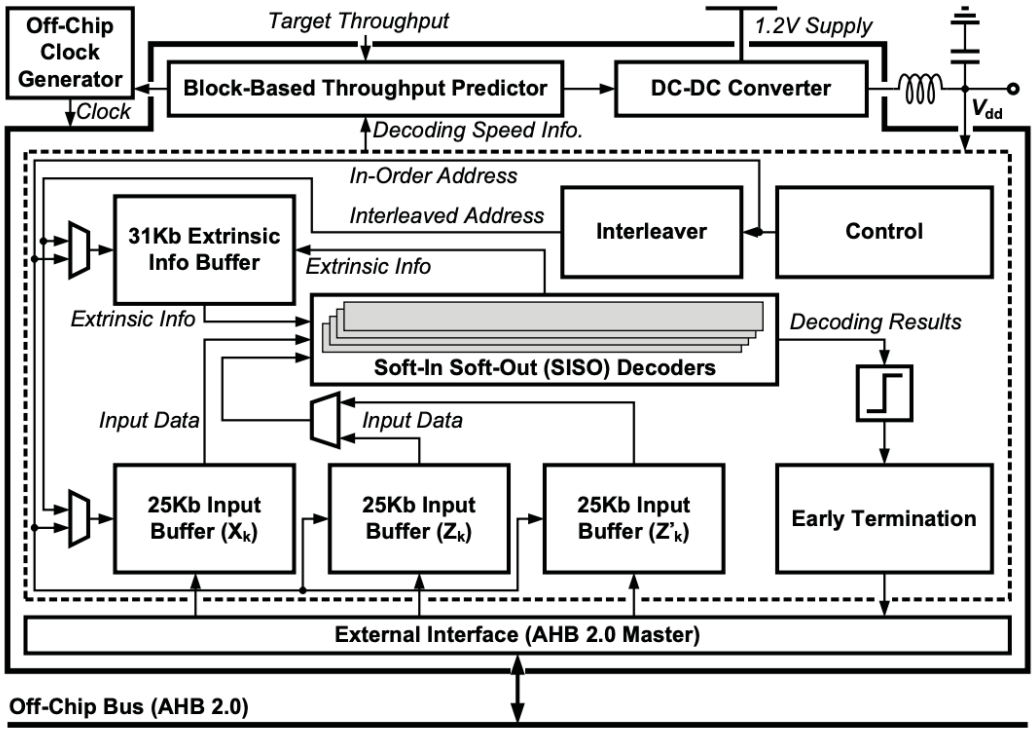



Fig. 3. The system architecture.

*See also* 3GPP TS 26.268 at 21 (v. 11), 17 (v. 8) and source code release accompanying 3GPP TS 26.268 ecall_fec.c line 200 *void UpdateBuffer(IntLLR *chLLRbuffer, const IntLLR *softInBits,*

*Int16 rv)* ecall_fec.c line 225 *void DecodeBuffer(const IntLLR *syst1, const IntLLR *syst2, const IntLLR *parity1, const IntLLR *parity2, Ord1 *decBits)*).

22. Additionally, and alternatively, as discussed above, each of the Accused Instrumentalities processed received baseband digital signals in an iterative manner through its implementation of turbo decoding. Any turbo decoder implementation receiving and operating on an incoming input signal included an input buffer structure and three component shift registers to provide time-aligned values (*i.e.*, first, second, and third shifted input signals) needed for each SISO computation. To the extent that any of the Accused Instrumentalities did not implement shift register functions in hardware, it was a well-established convention to implement shift register functions via software. *See, e.g.*, Wikipedia, "Shift register," *available at* https://en.wikipedia.org/wiki/Shift_register ("Many computer languages include instructions to 'shift right' and 'shift left' the data in a register, effectively dividing by two or multiplying by two for each place shifted."); GeeksforGeeks, "Left Shift and Right Shift Operators in C/C++," *available at* https://www.geeksforgeeks.org/left-shift-right-shift-operators-c-cpp/; mbedded.ninja, "Shift Registers," §5 (Apr. 8, 2020), *available at* https://blog.mbedded.ninja/electronics/components/shift-registers/.

23. Upon information and belief, the Accused Instrumentalities provided first and second soft decision decoders serially coupled in a circular circuit, wherein each decoder processed soft decision from the preceding decoder output data, and wherein the first decoder further received the first and second shifted input signals from the input buffer and the second decoder further received the third shifted input signal from the input buffer.

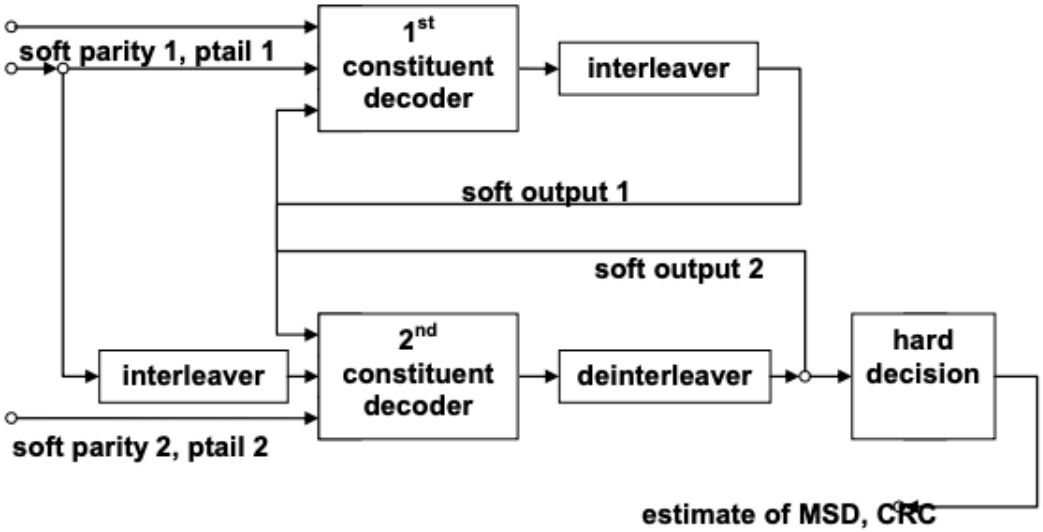



**Figure 18: Turbo decoder**

(*E.g.*, 3GPP TS 26.267, at 14 (v. 11, v. 8)).

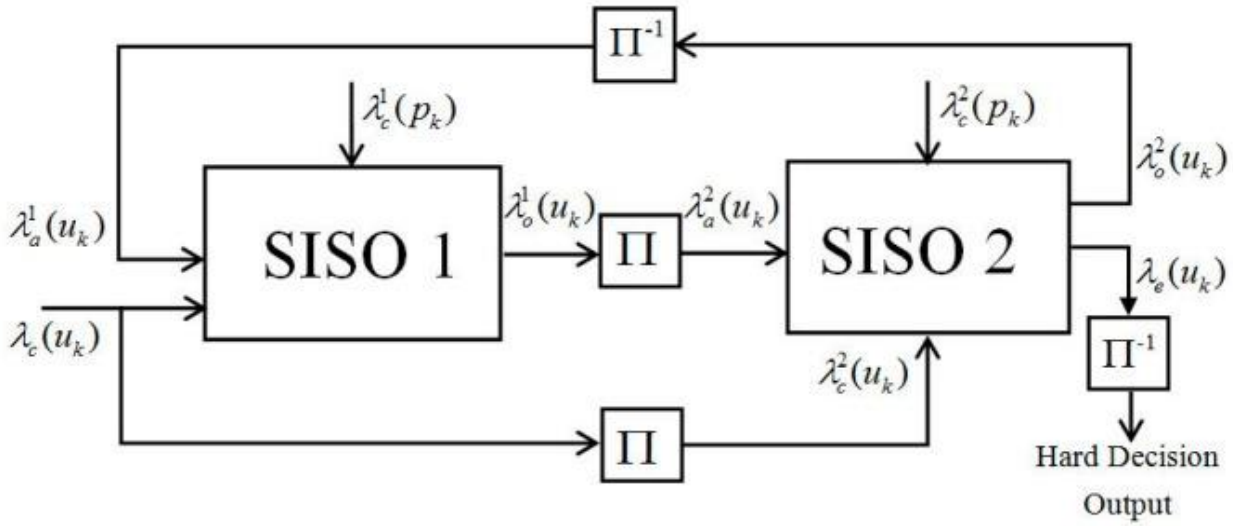
24. In the example, the first soft decision decoder output soft decision that became "soft output 1" after exiting "interleaver." This "soft output 1" was fed as input into the second soft decision decoder, "2nd constituent decoder," for the second decision decoder to process. The second soft decision decoder also received the "soft parity 2, ptail 2" input signal. The second soft decision decoder output soft decision that became "soft output 2" after exiting "deinterleaver." This "soft output 2" was fed as input into the first soft decision decoder for the first soft decision decoder to process. The first soft decision decoder, "1st constituent decoder," also received the "soft data" and "soft parity 1, ptail 1" input signals. *See also Cheng et. al.* "A 0.077 to 0.168 nj/bit/iteration Scalable 3GPP LTE Turbo Decoder with an Adaptive Sub-Block Parallel Scheme and an Embedded DVFS Engine," 2010 IEEE Custom Integrated Circuits Conference (CICC) (19-22 Sept. 2010), at 3, *available at* https://dspace.mit.edu/bitstream/handle/1721.1/72198/Chandrakasan-a%200.077%20to%200.168.pdf?sequence=1&isAllowed=y (citation removed): "Figure 3 shows

the system architecture. The blocks in the dashed box handle the turbo decoding operations, and those outside the dashed box belong to the DVFS scheme. Turbo decoding is an iterative process with several turbo iterations. Each turbo iteration comprises two soft-in, soft-out (SISO) decoding processes using BCJR algorithm with the first one performed on the input code block in the original order and the second one in an order generated by the interleaver block." *See also* Valenti et al., "UMTS Turbo Code and an Efficient Decoder," International Journal of Wireless Information Networks, Vol. 8, No. 4 (Oct. 2001), at 206, *available at* https://community.wvu.edu/~mcvalenti/documents/valenti01.pdf at Page 206, section 5. THE MAX* OPERATOR, 5.1. Log-MAP Algorithm, 5.2. Max-log-MAP Algorithm.

```
/*============================================================================*/
/* DECODER FUNCTION: Bcjr                                                     */
/*----------------------------------------------------------------------------*/
/* Description: BCJR algorithm                                                */
/*                                                                            */
/* In:     const IntLLR* parity      -> received parity soft bits             */
/* InOut:  IntLLR*       extrinsic  <-> extrinsic information                 */
/*----------------------------------------------------------------------------*/
void Bcjr(const IntLLR *parity, IntLLR *extrinsic)
```

(*E.g.*, 3GPP TS 26.268 at 21 (v. 11), 17 (v. 8)).

25. Additionally or alternatively, as explained above, all known commercial implementations of 4G LTE turbo decoders were iterative and functionally equivalent to the figures below. The figures below illustrate soft decision from the preceding decoder output (a posteriori information) being fed as an input (a priori information) in an iterative mode. The top of the figures below illustrates the feedback loop that forced 3G/4G LTE turbo decoders to be iterative. The iterative loop traversed elements SISO 1, Π, SISO 2, $\Pi^{-1}$, and closed with the signal path return to the input of SISO 1 to form a circular circuit.




*See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/ ("The SISO decoder consists of three input ports, which are system information $\lambda_c(u_k)$, parity information $\lambda_c(p_k)$, and a priori information $\lambda_a(u_k)$ which is computed by another SISO decoder. Two output ports of the SISO decoder generate external information $\lambda_e(u_k)$ and posteriori information $\lambda_o(u_k)$. The different superscripts represent information corresponding to different SISO decoders, and subscript *k* denotes *k*-th bit information of the current variable.").

26. As explained above, each of the Accused Instrumentalities processed received baseband digital signals in an iterative manner through its implementation of turbo decoding. Any turbo decoder implementation receiving and operating on an incoming input signal included an input buffer structure and three component shift registers to provide time-aligned values (*i.e.*, first, second, and third shifted input signals) needed for each SISO computation. To the extent that any of the Accused Instrumentalities did not implement shift register functions in hardware, it was a well-established convention to implement shift register functions via software. *See, e.g.*, Wikipedia, "Shift register," *available at* https://en.wikipedia.org/wiki/Shift_register ("Many

computer languages include instructions to 'shift right' and 'shift left' the data in a register, effectively dividing by two or multiplying by two for each place shifted."); GeeksforGeeks, "Left Shift and Right Shift Operators in C/C++," *available at* https://www.geeksforgeeks.org/left-shift-right-shift-operators-c-cpp/; mbedded.ninja, "Shift Registers," §5 (Apr. 8, 2020), *available at* https://blog.mbedded.ninja/electronics/components/shift-registers/.

27. Upon information and belief, the Accused Instrumentalities provided at least one memory module coupled to an output of each of the first and second soft decision decoders, wherein the output of the memory module associated with the second soft decision decoder was fed back as an input of the first soft decision decoder. For example, the Accused Instrumentalities included at least one memory module (*e.g.*, "interleaver" to the right of the "1st constituent decoder" and "deinterleaver" in the figures below), that was electrically coupled to an output of a corresponding soft decision decoder (*e.g.*, "1st constituent decoder" and "2nd constituent decoder")), wherein the output of the memory module associated with the second soft decision decoder ("deinterleaver") was fed back as an input of the first soft decision decoder, as shown below:

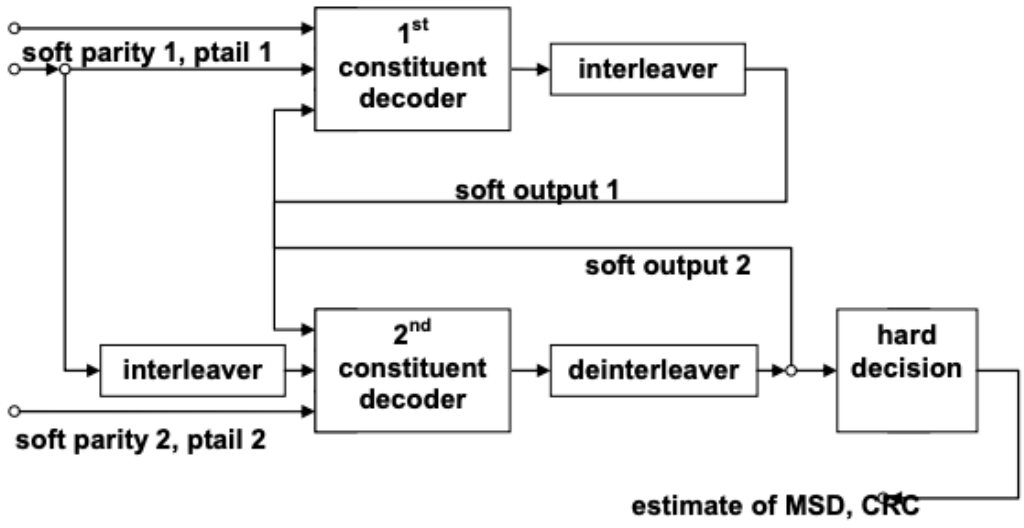



**Figure 18: Turbo decoder**

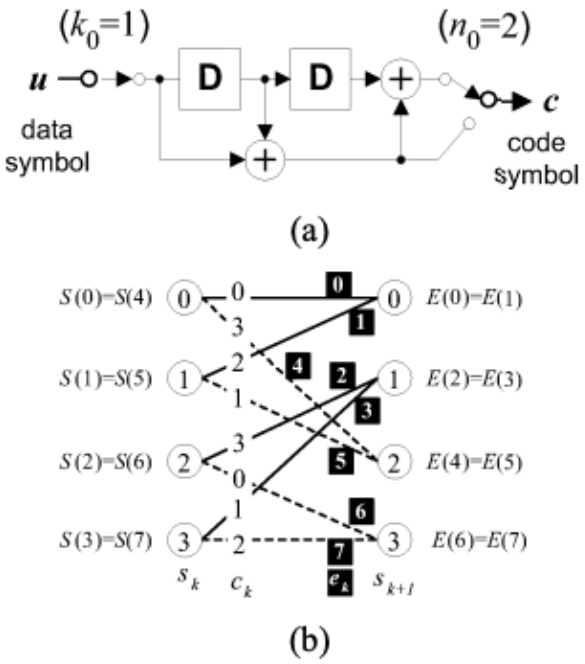
(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)).

28. In the example, the "deinterleaver" memory module was associated with the second soft decision decoder, "2nd constituent decoder." The output of the deinterleaver, "soft output 2," was fed back as an input of the first soft decision decoder, "1st constituent decoder." Additional evidence that the "interleaver" and "deinterleaver" comprised memory modules is shown in source code associated with the figure:

```
/* initialize memory */
  Le12 = (IntLLR*)&decBits[0];
  Le21 = (IntLLR*)&decBits[sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL)];

  memset(Le12, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));
  memset(Le21, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

  /* iterative decoding */
  for (i = 0; i < FEC_ITERATIONS; i++) {
    memcpy(Le12, Le21, sizeof(IntLLR)*(NRB  INFO  CRC + NRB  TAIL));

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC+NRB_TAIL; j++) {
      temp = (Int32)Le12[j] + (Int32)syst1[j];
      Le12[j] = (ABS(temp) < LLR_MAX) ?
        (IntLLR)temp : (IntLLR)(SIGN(temp)*LLR_MAX);
    }
    /* decode code one (produces Le12) */
    Bcjr(parity1, Le12);

    /* interleave extrinsic information (produces interleaved Le12) */
    Interleave(Le12, Le21);

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC; j++) {
      temp = (Int32)Le21[j] + (Int32)syst1[interleaverSeq[j]];
      Le21[j] = (ABS(temp) < LLR_MAX) ?
        (IntLLR)temp : (IntLLR)((SIGN(temp))*LLR_MAX);
    }
    for (j = 0; j < NRB_TAIL; j++) {
      Le21[j+NRB_INFO_CRC] = syst2[j];
```

```
  }
  /* decode code two (produces interleaved Le21) */
  Bcjr(parity2, Le21);

  /* deinterleave extrinsic information (produces Le21) */
  Deinterleave(Le21);
}
```

(*E.g.*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268). *See also* Mansour et al., "VLSI Architectures for SISO-APP Decoders," IEEE Transactions On Very Large Scale Integration ("VLSI") Systems, Vol. 11, No. 4 (Aug. 2003) "Fig. 1. A (2, 1, 3) convolutional code. (a) An encoder with 2 memory delay elements (D) and modulo 2 adders, data symbol alphabet {0, 1}, code symbol alphabet {0, 1, 2, 3}, memory states {0, 1, 2, 3}, and code rate R = (1=2). (b) A trellis section where solid edges correspond to u = 0, and dashed edges correspond to u = 1. The output code symbols c are shown on the edges. The edges are numbered with black squares, and the edge starting and ending states are shown on the left and right, respectively." *See also id.* at FIG. 1:



*See also* Valenti et al., "UMTS Turbo Code and an Efficient Decoder," International Journal of Wireless Information Networks, Vol. 8, No. 4 (Oct. 2001), at 207, *available at* https://community.wvu.edu/~mcvalenti/documents/valenti01.pdf: "Two key observations should be pointed out before going into the details of the algorithm: (1) It does not matter whether the forward sweep or the reverse sweep is performed first; and (2) while the partial path metrics for the entire first sweep (forward or backward) must be stored in memory, they do not need to be stored for the entire second sweep. This is because the LLR values can be computed during the second sweep, and thus partial path metrics for only two stages of the trellis (the current and previous stages) must be maintained during the second sweep." *See also Cheng et. al.* "A 0.077 to 0.168 nj/bit/iteration Scalable 3GPP LTE Turbo Decoder with an Adaptive Sub-Block Parallel Scheme and an Embedded DVFS Engine," 2010 IEEE Custom Integrated Circuits Conference (CICC) (19-22 Sept. 2010), at Fig. 3, *available at* https://dspace.mit.edu/bitstream/handle/1721.1/72198/Chandrakasan-a%200.077%20to%200.168.pdf?sequence=1&isAllowed=y:

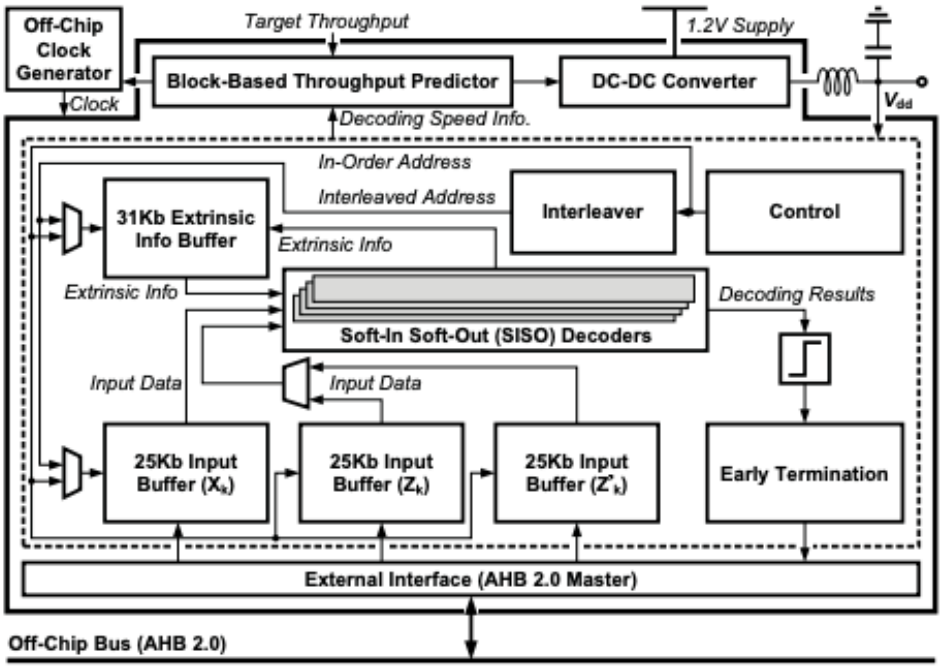
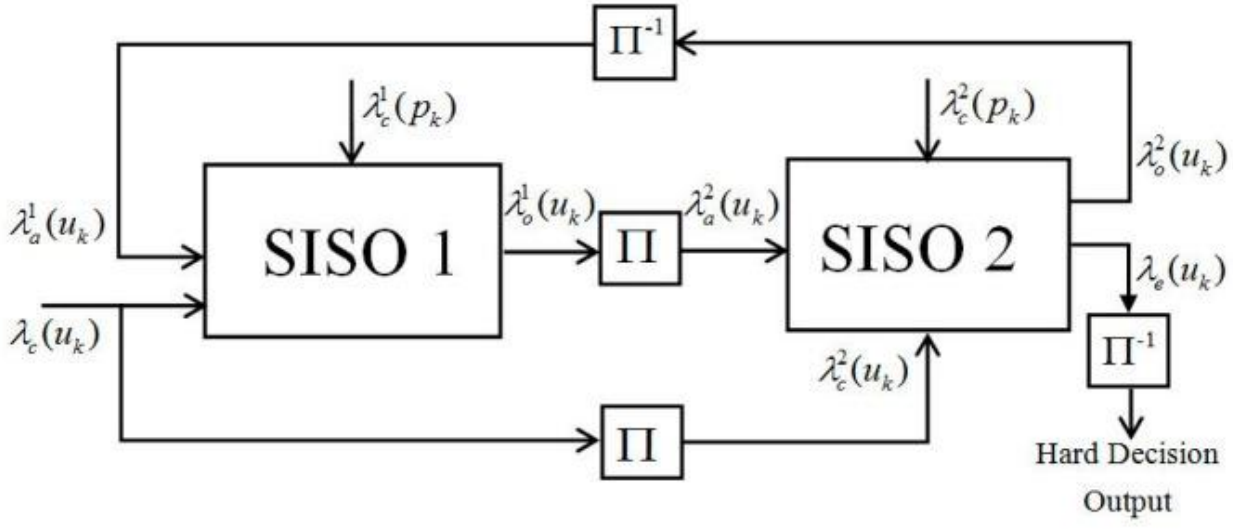



Fig. 3. The system architecture.

*See also*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268:

```
ecall_fec.c line 232
 /* initialize memory */
 Le12 = (IntLLR*)&decBits[0];
 Le21 = (IntLLR*)&decBits[sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL)];
 memset(Le12, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));
 memset(Le21, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));
ecall_fec.c line 250
 Bcjr(parity1, Le12); /*corresponding memory module Le12*/
ecall_fec.c line 265
 Bcjr(parity2, Le21); /*corresponding memory module Le21*/
```

29. Additionally, or alternatively, as explained above, all known commercial implementations of 4G LTE turbo decoders were iterative and functionally equivalent to the figures below. The figures below illustrate output of the memory module associated with a last soft decision decoder fed back as an input to the first soft decision decoder via interleaving and/or de-interleaving (*e.g.*, the a posteriori output of SISO 2 was de-interleaved with the associated memory module and fed back as a priori input of SISO 1).



*See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/) ("Π and Π−1 denote interleaving and de-interleaving, respectively.").

30. Upon information and belief, the Accused Instrumentalities processed systematic information data and extrinsic information data using the maximum a posteriori (MAP) probability algorithm, and/or logarithm approximation algorithm. For example, the Accused Instrumentalities used at least the BCJR algorithm for turbo decoding in accordance with the figures below:

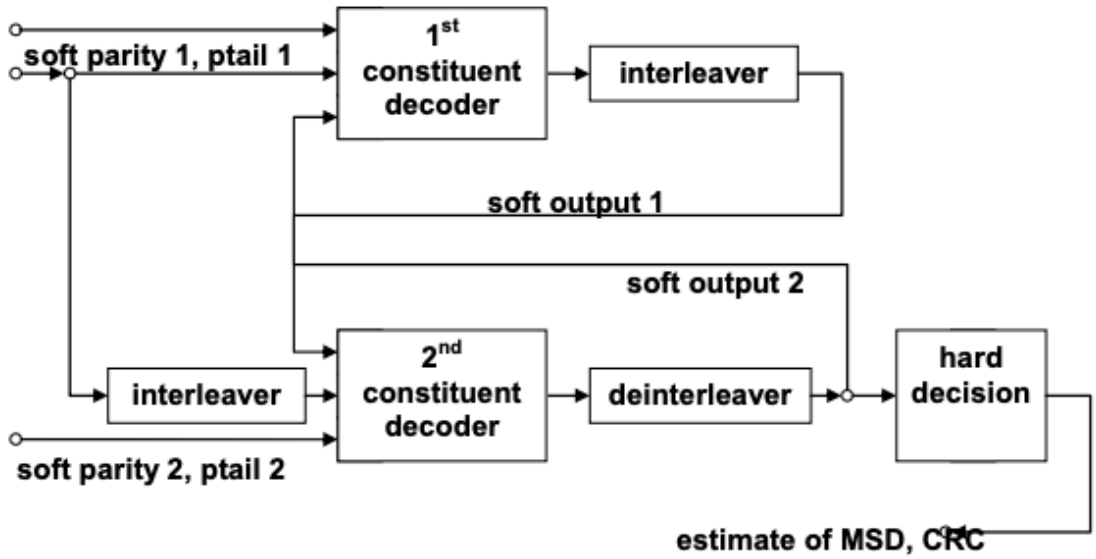



**Figure 18: Turbo decoder**

(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)).

31. The source code associated with the figure indicates that it processes systematic information data and extrinsic information data using the BCJR algorithm:

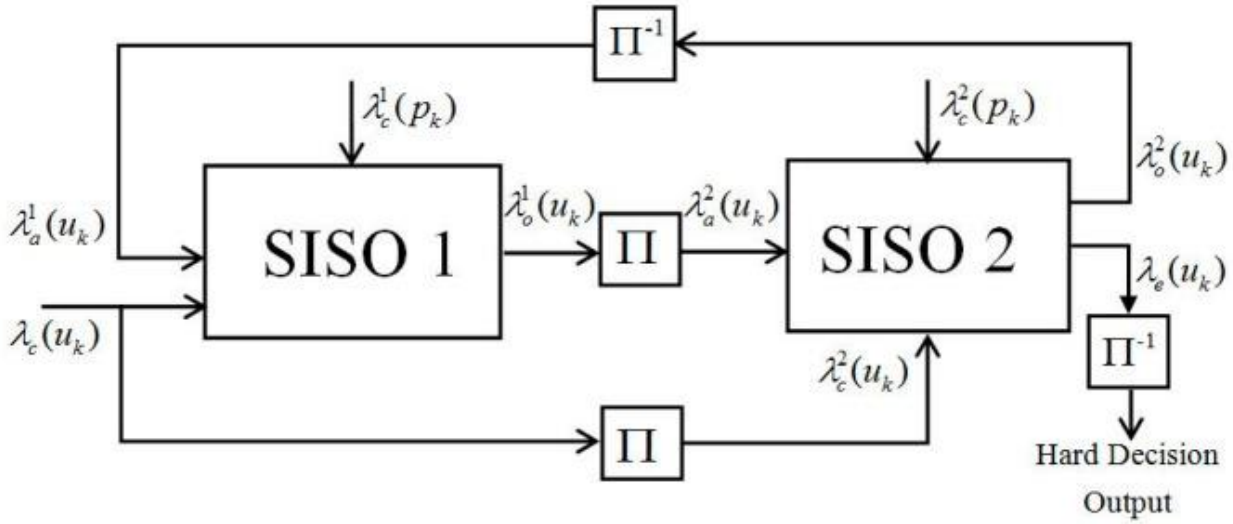
```
/* initialize memory */
  Le12 = (IntLLR*)&decBits[0];
  Le21 = (IntLLR*)&decBits[sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL)];
  memset(Le12, 0, sizeof(IntLLR)*(NRB  INFO  CRC + NRB  TAIL));
  memset(Le21, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

  /* iterative decoding */
  for (i = 0; i < FEC_ITERATIONS; i++) {
    memcpy(Le12, Le21, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC+NRB_TAIL; j++) {
      temp = (Int32)Le12[j] + (Int32)syst1[j];
      Le12[j] = (ABS (temp) < LLR  MAX)?
        (IntLLR)temp : (IntLLR)(SIGN(temp)*LLR_MAX);
    }
    /* decode code one (produces Le12) */
    Bcjr(parity1, Le12);

    /* interleave extrinsic information (produces interleaved Le12) */
    Interleave(Le12, Le21);

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC; j++) {
      temp = (Int32)Le21[j] + (Int32)syst1[interleaverSeq[j]];
      Le21[j] = (ABS(temp) < LLR  MAX) ?
        (IntLLR)temp : (IntLLR)((SIGN(temp))*LLR_MAX);
    }
    for (j = 0; j < NRB  TAIL; j++) {
      Le21[j+NRB_INFO_CRC] = syst2[j];
    }

  /* decode code two (produces interleaved Le21) */
  Bcjr(parity2, Le21);

  /* deinterleave extrinsic information (produces Le21) */
  Deinterleave(Le21);
}
```

(*E.g.*, *see*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268).

32. The BCJR algorithm is a MAP probability algorithm that processes systematic information data and extrinsic information data: "Turbo codes are composed of an interconnection of component codes through interleavers, typically convolutional codes, and their decoders consist of an equal number of component decoders each of which operates on its corresponding codeword and shares information with other component decoders iteratively according to the topology of the encoder. The decoding algorithm in the component decoders is the maximum a-posteriori probability (MAP) algorithm typically implemented in the form known as the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm. The main advantage of a MAP decoding algorithm over a maximum likelihood decoding algorithm such as the Viterbi algorithm is that it produces optimum soft information which is crucial to the operation of these decoders. The BCJR algorithm was generalized in [S. Benedetto *et al.*, "A Soft-Input Soft-Output Maximum a posteriori (Map) Module to Decode Parallel and Serial Concatenated Codes," JPL, TDA Progress Report 42-127, Nov. 1996.] into a soft-input soft-output a posteriori probability (SISO-APP) algorithm to be used as a building block for iterative decoding in code networks with generic topologies. The advantages of the SISO-APP algorithm over other forms of the MAP algorithm is that it is independent of the code type (systematic/nonsystematic, recursive/nonrecursive, trellis with multiple edges), and it generates reliability information for code symbols as well as message symbols which makes it applicable irrespective of the concatenation scheme (parallel/serial/hybrid), and hence will be considered in this paper." *See* Mansour et al., "VLSI Architectures for SISO-APP Decoders," IEEE Transactions On Very Large Scale Integration ("VLSI") Systems, Vol. 11, No. 4 (Aug. 2003), at 627, *available at* http://shanbhag.ece.illinois.edu/publications/mansr-tvlsi-2003-2.pdf (citations removed). *See also*

*id.* at 629 ("The decoding problem can now be defined as follows: given a noisy version of $\underline{c}$ denoted by $\underline{y}=\Delta(y_1,\cdots,y_k,\cdots,y_L)$, find the data sequence $\underline{u}$. There are two probabilistic solutions to this decoding problem. Maximum likelihood (ML) decoding determines the most likely connected path $\underline{s}$ through the trellis that maximizes the probability $P(\underline{y}|\underline{s})$. From $\underline{s}$, the most likely data sequence $\underline{u}$ is easily determined using (1). On the other hand, MAP decoding, which we consider here, determines $\underline{u}$ by estimating each of the symbols $u_k$ independently using the observations $\underline{y}$. The kth estimated symbol $u_k$ is the one that maximizes the posterior probability $P(u_k|\underline{y})$, and hence the name symbol-by-symbol MAP. The SISO-APP algorithm, a generalized version of the BCJR-APP algorithm, is a probabilistic algorithm that solves the MAP decoding problem.") (citations removed).

33. Additionally, or alternatively, as explained above, all known commercial implementations of 4G LTE turbo decoders were iterative and functionally equivalent to the figures below. The figures below illustrate that each decoder processed systemic information data and extrinsic information data—system information $\lambda_c(u_k)$, parity information $\lambda_c(p_k)$, a priori information $\lambda_a(u_k)$ and external information $\lambda_e(u_k)$:



*See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/. *See also, e.g.*, Guohui Wang *et al.*, "High-throughput Contention-Free Concurrent Interleaver Architecture for Multi-standard Turbo Decoder," ASAP 2011—22nd IEEE International Conference on Application-specific Systems, Architectures and Processors 113 (Sept. 2011); Cristian Anghel et al., "CTC Turbo Decoding Architecture for LTE Systems Implemented on FPGA," ICN 2012: The Eleventh International Conference on Networks 199, 199-200 (2012).

34. Each soft decision decoder on all known commercial implementations of any turbo decoder processed systematic information data and extrinsic information data using a maximum a posteriori (MAP) probability algorithm and/or a logarithm approximation algorithm to yield posteriori information $\lambda_o(u_k)$. *See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/ (discussing use of a log maximum a posteriori decoding algorithm). *See also, e.g.*, Guohui Wang *et al.*, "High-throughput Contention-Free Concurrent Interleaver Architecture for Multi-standard Turbo Decoder," ASAP 2011—22nd IEEE International Conference on Application-specific Systems, Architectures and Processors 113 (Sept. 2011) (noting that MAP decoders are used as the component SISO decoders); Cristian Anghel et al., "CTC Turbo Decoding Architecture for LTE Systems Implemented on FPGA," ICN 2012: The Eleventh International Conference on Networks 199, 200 (2012) (describing ideal use of classic MAP algorithm and practical implementation of log-MAP algorithms).

35. Upon information and belief, the Accused Instrumentalities generated soft decision based on the maximum a posteriori (MAP) probability algorithm and/or logarithm approximation algorithm. For example, the Accused Instrumentalities used at least the BCJR algorithm for decoding in accordance with the figures below:

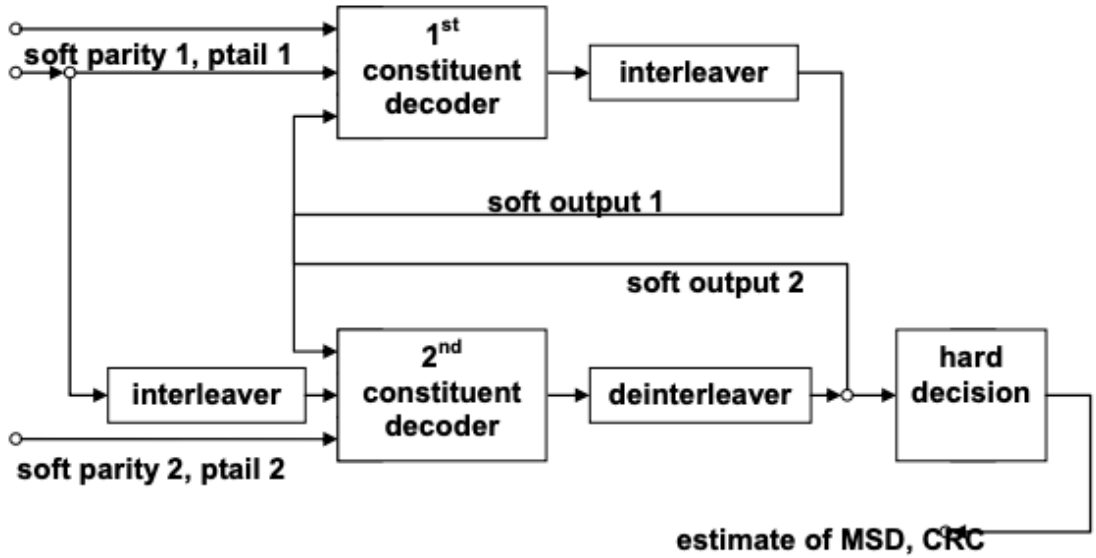



Figure 18: Turbo decoder

(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)). The source code associated with the figure, for example, indicates that it generated soft decision based on the BCJR algorithm. Data processed by the BCJR algorithm was interleaved or deinterleaved, resulting in "soft output 1" or "soft output 2," respectively, as shown in the figure:

```
/* initialize memory */
 Le12 = (IntLLR*)&decBits[0];
 Le21 = (IntLLR*)&decBits[sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL)];
 memset(Le12, 0, sizeof(IntLLR)*(NRB  INFO  CRC + NRB  TAIL));
 memset(Le21, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

 /* iterative decoding */
 for (i = 0; i < FEC_ITERATIONS; i++) {
  memcpy(Le12, Le21, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

  /* add received systematic bits to extrinsic information */
  for (j = 0; j < NRB_INFO_CRC+NRB_TAIL; j++) {
   temp = (Int32)Le12[j] + (Int32)syst1[j];
   Le12[j] = (ABS (temp) < LLR  MAX)?
    (IntLLR)temp : (IntLLR)(SIGN(temp)*LLR_MAX);
  }
  /* decode code one (produces Le12) */
  Bcjr(parity1, Le12);

  /* interleave extrinsic information (produces interleaved Le12) */
  Interleave(Le12, Le21);

  /* add received systematic bits to extrinsic information */
  for (j = 0; j < NRB_INFO_CRC; j++) {
   temp = (Int32)Le21[j] + (Int32)syst1[interleaverSeq[j]];
   Le21[j] = (ABS(temp) < LLR  MAX) ?
    (IntLLR)temp : (IntLLR)((SIGN(temp))*LLR_MAX);
  }
  for (j = 0; j < NRB  TAIL; j++) {
   Le21[j+NRB_INFO_CRC] = syst2[j];
  }

 /* decode code two (produces interleaved Le21) */
 Bcjr(parity2, Le21);

 /* deinterleave extrinsic information (produces Le21) */
 Deinterleave(Le21);
}
```

(*E.g.*, *see*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268). The BCJR algorithm included a MAP probability algorithm which processed soft input to generate soft decision: "Turbo codes are composed of an interconnection of component codes through interleavers, typically convolutional codes, and their decoders consist of an equal number of component decoders each of which operates on its corresponding codeword and shares information with other component decoders iteratively according to the topology of the encoder. The decoding algorithm in the component decoders is the maximum a-posteriori probability (MAP) algorithm typically implemented in the form known as the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm. The main advantage of a MAP decoding algorithm over a maximum likelihood decoding algorithm such as the Viterbi algorithm is that it produces optimum soft information which is crucial to the operation of these decoders. The BCJR algorithm was generalized in [S. Benedetto et al., "A Soft-Input Soft-Output Maximum a posteriori (Map) Module to Decode Parallel and Serial Concatenated Codes," JPL, TDA Progress Report 42-127, Nov. 1996.] into a soft-input soft-output a posteriori probability (SISO-APP) algorithm to be used as a building block for iterative decoding in code networks with generic topologies. The advantages of the SISO-APP algorithm over other forms of the MAP algorithm is that it is independent of the code type (systematic/nonsystematic, recursive/nonrecursive, trellis with multiple edges), and it generates reliability information for code symbols as well as message symbols which makes it applicable irrespective of the concatenation scheme (parallel/serial/hybrid), and hence will be considered in this paper." *See* Mansour et al., "VLSI Architectures for SISO-APP Decoders," IEEE Transactions On Very Large Scale Integration ("VLSI") Systems, Vol. 11, No. 4 (Aug. 2003), at 627, *available at* http://shanbhag.ece.illinois.edu/publications/mansr-tvlsi-2003-2.pdf. *See also id.* at 629 ("The decoding problem can now be defined as follows: given a noisy version of $\underline{c}$ denoted by

$\underline{y}=\Delta(y_1,\cdots,y_k,\cdots,y_L)$, find the data sequence $\underline{u}$. There are two probabilistic solutions to this decoding problem. Maximum likelihood (ML) decoding determines the most likely connected path $\underline{s}$ through the trellis that maximizes the probability $P(\underline{y}|\underline{s})$. From $\underline{s}$, the most likely data sequence $\underline{u}$ is easily determined using (1). On the other hand, MAP decoding, which we consider here, determines $\underline{u}$ by estimating each of the symbols $u_k$ independently using the observations $\underline{y}$. The kth estimated symbol $u_k$ is the one that maximizes the posterior probability $P(u_k|\underline{y})$, and hence the name symbol-by-symbol MAP. The SISO-APP algorithm, a generalized version of the BCJR-APP algorithm [6], is a probabilistic algorithm that solves the MAP decoding problem.").

36. Upon information and belief, the Accused Instrumentalities weighed and stored soft decision information into the corresponding memory module (*e.g.*, "interleaver" or "deinterleaver") as shown in the figures below:

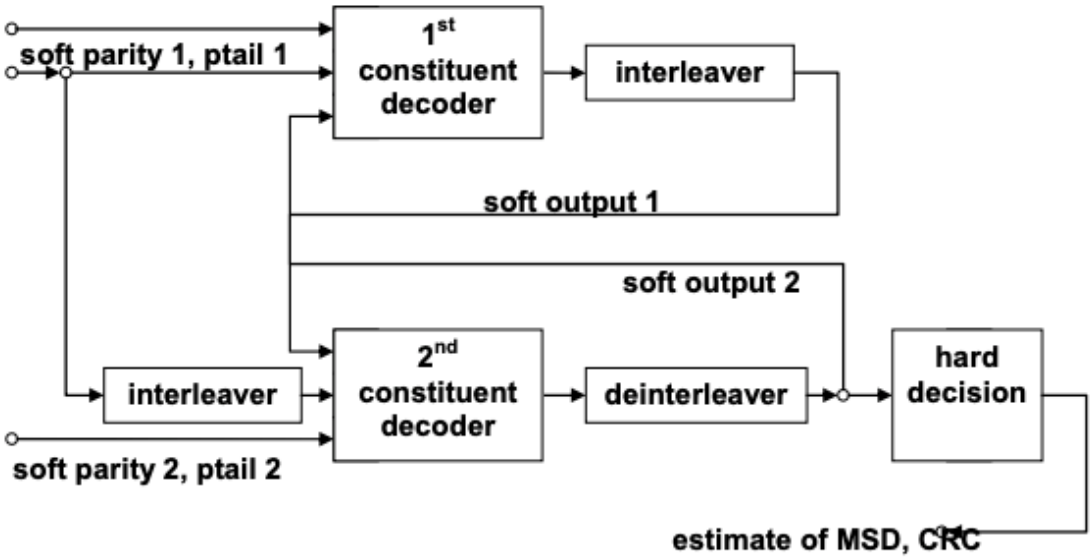



**Figure 18: Turbo decoder**

(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)). The source code associated with the figure indicates use of the BCJR algorithm as shown below:

```
/* initialize memory */
  Le12 = (IntLLR*)&decBits[0];
  Le21 = (IntLLR*)&decBits[sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL)];
  memset(Le12, 0, sizeof(IntLLR)*(NRB  INFO  CRC + NRB  TAIL));
  memset(Le21, 0, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

  /* iterative decoding */
  for (i = 0; i < FEC_ITERATIONS; i++) {
    memcpy(Le12, Le21, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC+NRB_TAIL; j++) {
      temp = (Int32)Le12[j] + (Int32)syst1[j];
      Le12[j] = (ABS (temp) < LLR  MAX)?
        (IntLLR)temp : (IntLLR)(SIGN(temp)*LLR_MAX);
    }
    /* decode code one (produces Le12) */
    Bcjr(parity1, Le12);

    /* interleave extrinsic information (produces interleaved Le12) */
    Interleave(Le12, Le21);

    /* add received systematic bits to extrinsic information */
    for (j = 0; j < NRB_INFO_CRC; j++) {
      temp = (Int32)Le21[j] + (Int32)syst1[interleaverSeq[j]];
      Le21[j] = (ABS(temp) < LLR  MAX) ?
        (IntLLR)temp : (IntLLR)((SIGN(temp))*LLR_MAX);
    }
    for (j = 0; j < NRB  TAIL; j++) {
      Le21[j+NRB_INFO_CRC] = syst2[j];
    }

  /* decode code two (produces interleaved Le21) */
  Bcjr(parity2, Le21);

  /* deinterleave extrinsic information (produces Le21) */
  Deinterleave(Le21);
}
```

(*E.g.*, *see*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268).

```
/* normalization of betaQ */
for (i = 0; i < FEC_STATES; i++) {
temp = (Int32)bTemp1[i] -  (Int32)norm;
bTemp1[i] = (temp < (-LLR  MAX)) ? (IntLLR)(-LLR  MAX) :  (IntLLR)temp;
}

...

/* normalization of alphaQ */
for (i = 0; i < FEC  STATES; i++) {
temp = (Int32)alpha2[i] - (Int32)norm;
alpha2[i] = (temp < (-LLR_MAX)) ? (IntLLR)(-LLR_MAX) : (IntLLR)temp;
}
```

(*E.g.*, *see,* source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 318-352). Data processed by the BCJR algorithm, for example, generates soft decision information. "Turbo codes are composed of an interconnection of component codes through interleavers, typically convolutional codes, and their decoders consist of an equal number of component decoders each of which operates on its corresponding codeword and shares information with other component decoders iteratively according to the topology of the encoder. The decoding algorithm in the component decoders is the maximum a-posteriori probability (MAP) algorithm typically implemented in the form known as the Bahl–Cocke–Jelinek–Raviv (BCJR) algorithm. The main advantage of a MAP decoding algorithm over a maximum likelihood decoding algorithm such as the Viterbi algorithm is that it produces optimum soft information which is crucial to the operation of these decoders. The BCJR algorithm was generalized in [S. Benedetto et al., "A Soft-Input Soft-Output Maximum a posteriori (Map) Module to Decode Parallel and Serial Concatenated Codes," JPL, TDA Progress Report 42-127, Nov. 1996.] into a soft-input soft-output a posteriori probability (SISO-APP) algorithm to be used as a building block for iterative decoding in code networks with generic topologies. The advantages of the SISO-APP algorithm over other forms of

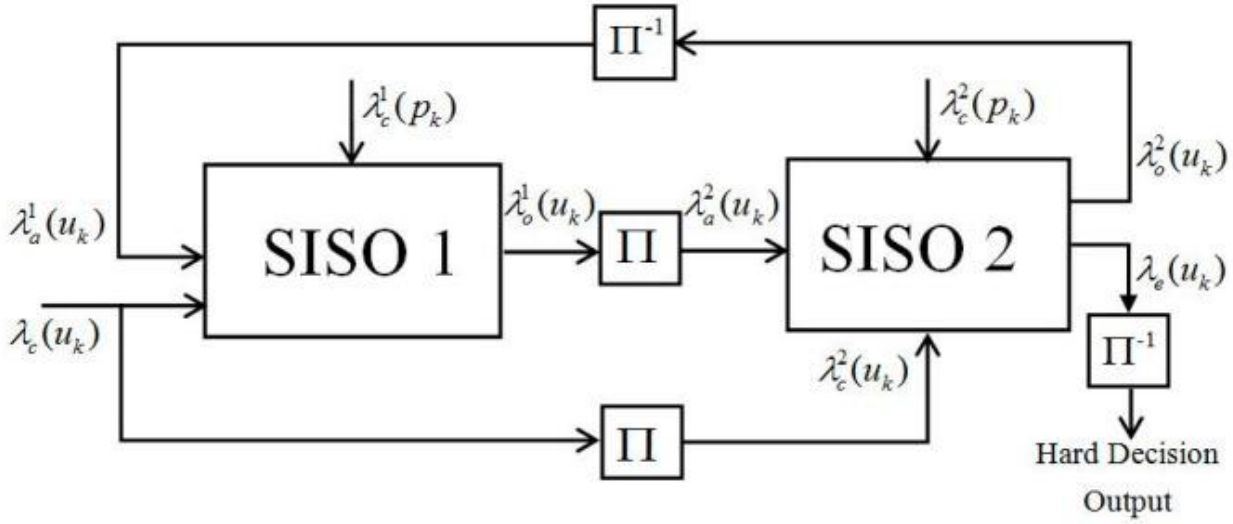
the MAP algorithm is that it is independent of the code type (systematic/nonsystematic, recursive/nonrecursive, trellis with multiple edges), and it generates reliability information for code symbols as well as message symbols which makes it applicable irrespective of the concatenation scheme (parallel/serial/hybrid), and hence will be considered in this paper." *See* Mansour et al., "VLSI Architectures for SISO-APP Decoders," IEEE Transactions On Very Large Scale Integration ("VLSI") Systems, Vol. 11, No. 4 (Aug. 2003), at 627, *available at* http://shanbhag.ece.illinois.edu/publications/mansr-tvlsi-2003-2.pdf (citations removed).

37. As shown by the source code associated with the figure, the soft decision information was then normalized, or "weighted," and then output to either (1) the "interleaver" memory module if performed at the first soft decision decoder or (2) the "deinterleaver" memory module if performed at the second soft decision decoder shown in the figure. Note that the variables "betaQ" and "alphaQ" comprise examples of such soft decision information in the source code:

```
/* normalization of betaQ */
for (i = 0; i < FEC_STATES; i++) {
temp = (Int32)bTemp1[i] -  (Int32)norm;
bTemp1[i] = (temp < (-LLR  MAX)) ? (IntLLR)(-LLR  MAX) :  (IntLLR)temp;
}

...

/* normalization of alphaQ */
for (i = 0; i < FEC  STATES; i++) {
temp = (Int32)alpha2[i] - (Int32)norm;
alpha2[i] = (temp < (-LLR_MAX)) ? (IntLLR)(-LLR_MAX) : (IntLLR)temp;
}
```

(*E.g.*, *see*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 318-352).

38. Additionally, or alternatively, as explained above, all known commercial implementations of 4G LTE turbo decoders were iterative and functionally equivalent to the figures below. The figures below illustrate that soft decision information was stored in the corresponding memory module (*e.g.*, the a posteriori output of SISO 1 is stored in the associated interleaver memory module Π, while the a posteriori output of SISO 2 is stored in the associated de-interleaver memory module $\Pi^{-1}$)




*See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/.

39. On information and belief, use of any viable maximum a posteriori (MAP) probability algorithm and/or logarithm approximation algorithm necessarily required weighing (or "normalization"). *See* Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/ (discussing use of a log maximum a posteriori decoding algorithm). *See also, e.g.,* Guohui Wang *et al.*, "High-throughput Contention-Free Concurrent Interleaver Architecture for Multi-standard Turbo Decoder," ASAP 2011—22nd

IEEE International Conference on Application-specific Systems, Architectures and Processors 113 (Sept. 2011) (noting that MAP decoders are used as the component SISO decoders); Cristian Anghel et al., "CTC Turbo Decoding Architecture for LTE Systems Implemented on FPGA," ICN 2012: The Eleventh International Conference on Networks 199, 200 (2012) (describing ideal use of classic MAP algorithm and practical implementation of log-MAP algorithms).

40. The Accused Instrumentalities performed, for a predetermined number of times, iterative decoding from the first to the last of multiple decoders, wherein an output from the last soft decision decoder was fed back as an input to the first soft decision decoder, then from the first to the second decoders, and propagated to the last decoder in a circular circuit, as shown in the figures below:

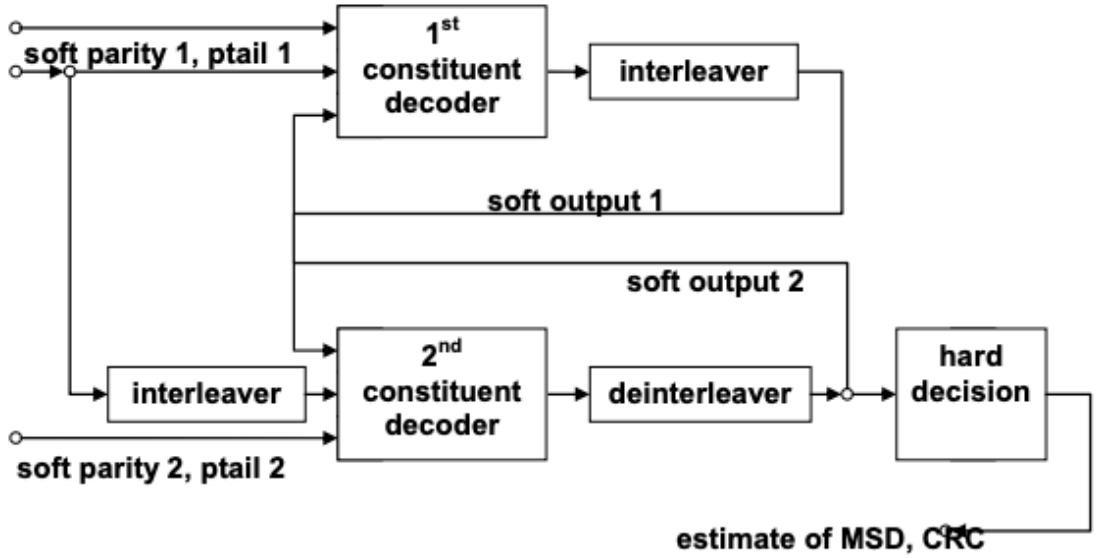



**Figure 18: Turbo decoder**

(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)). As shown in the example figure, decoding occurred from the first soft decision decoder, "1st constituent decoder," to the second, or last, soft decision decoder, "2nd constituent decoder." The second soft decision decoder outputs "soft output 2," which is fed as back as an input to the first soft decision decoder. The first soft decision

decoder outputs "soft output 1." This "soft output 1" is fed as input into the second soft decision decoder. The second soft decision decoder is the last soft decision decoder in a circular circuit propagating from the first soft decision decoder to the second soft decision decoder to the first … to the second to the first to the second, etc. This process was performed a predetermined number of times as defined by the software governing the turbo decoding process. For example, the default number of iterations defined in the source code associated with the figure is 8 iterations, stored in the variable FEC_ITERATIONS:

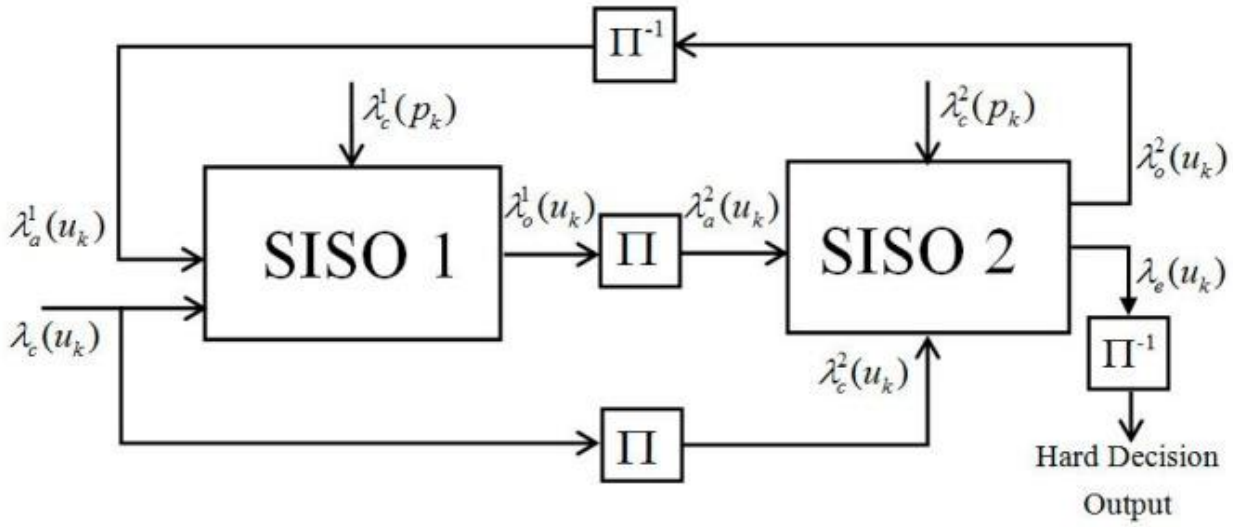
```
#define FEC_VAR                 (30206)         variance: 1/4550000 in Q37
#define FEC_MEAN                (0xB9999A)      mean: 5.8 in Q21
#define FEC_ITERATIONS          (8)             number of decoder iterations
#define FEC_STATES              (8)             number of decoder states
```

(*E.g.*, 3GPP TS 26.267, at 25 (v. 11), 24 (v. 8)). As another example, FEC_ITERATIONS is used during decoding process:

```
/* iterative decoding */
for (i = 0; i < FEC_ITERATIONS; i++) {
memcpy(Le12, Le21, sizeof(IntLLR)*(NRB_INFO_CRC + NRB_TAIL));
/* add received systematic bits to extrinsic information */
        for (j = 0; j < NRB_INFO_CRC+NRB_TAIL; j++) {
                temp = (Int32)Le12[j] + (Int32)syst1[j];
                Le12[j] = (ABS(temp) < LLR_MAX) ?
                        (IntLLR)temp : (IntLLR)(SIGN(temp)*LLR_MAX);
        }
        /* decode code one (produces Le12) */
        Bcjr(parity1, Le12);
        /* interleave extrinsic information (produces interleaved Le12) */
        Interleave(Le12, Le21);
        /* add received systematic bits to extrinsic information */
        for (j = 0; j < NRB_INFO_CRC; j++) {
                temp = (Int32)Le21[j] + (Int32)syst1[interleaverSeq[j]];
                Le21[j] = (ABS(temp) < LLR_MAX) ?
                        (IntLLR)temp : (IntLLR)((SIGN(temp))*LLR_MAX);
        }
        for (j = 0; j < NRB_TAIL; j++) {
                Le21[j+NRB_INFO_CRC] = syst2[j];
        }
        /* decode code two (produces interleaved Le21) */
        Bcjr(parity2, Le21);
        /* deinterleave extrinsic information (produces Le21) */
        Deinterleave(Le21);
}
```

(*E.g.*, *see*, source code release accompanying 3GPP TS 26.268, ecall_fec.c, lines 232-268).

41. Additionally, or alternatively, as above, all known commercial implementations of 4G LTE turbo decoders were iterative and functionally equivalent to the figures below. The figures below illustrates an output from the last soft decision decoder was fed back as an input to the first soft decision decoder, then from the first to the second decoders, and propagate to the last decoder in a circular circuit (*e.g.*, the a posteriori output of SISO 2 is de-interleaved with the associated memory module and fed as a priori input of SISO 1, while the a posteriori output of SISO 1 is interleaved with the associated memory module and fed as a priori input of SISO 2).




(*E.g.*, Jun Li et al., "Turbo Decoder Design based on an LUT-Normalized Log-MAP Algorithm," Entropy (Basel) (Aug. 20, 2019), *available at* https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7515343/). Iterative turbo decoder implementations required completing a certain number of iterations to complete decoding a frame with a satisfactory degree of confidence. Iterative decoding must have therefore been performed a predetermined number of times according to a stopping rule. *See, e.g.*, A. Matache et al., "Stopping Rules for Turbo Decoders," TMO Progress Report 42-142 (Aug. 15, 2000), available at https://ipnpr.jpl.nasa.gov/progress_report/42-142/142J.pdf.

42. Plaintiff has been damaged as a result of Defendant's infringing conduct. Defendant is thus liable to Plaintiff for damages in an amount that adequately compensates Plaintiff for such Defendant's infringement of the '742 Patent, *i.e.*, in an amount that by law cannot be less than would constitute a reasonable royalty for the use of the patented technology, together with interest and costs as fixed by this Court under 35 U.S.C. § 284.

## IV. JURY DEMAND

Plaintiff, under Rule 38 of the Federal Rules of Civil Procedure, requests a trial by jury of any issues so triable by right.

## V. PRAYER FOR RELIEF

WHEREFORE, Plaintiff respectfully requests that the Court find in its favor and against Defendant, and that the Court grant Plaintiff the following relief:

a. Judgment that one or more claims of United States Patent No. 6,813,742 have been infringed, either literally and/or under the doctrine of equivalents, by Defendant;

b. Judgment that Defendant account for and pay to Plaintiff all damages to and costs incurred by Plaintiff because of Defendant's infringing activities and other conduct complained of herein;

c. That Plaintiff be granted pre-judgment and post-judgment interest on the damages caused by Defendant's infringing activities and other conduct complained of herein; and

d. That Plaintiff be granted such other and further relief as the Court may deem just and proper under the circumstances.

June 7, 2024

DIRECTION IP LAW

*/s/ David R. Bennett*

David R. Bennett (IL Bar No.: 6244214)
P.O. Box 14184
Chicago, Illinois 60614-0184
Telephone: (312) 291-1667
dbennett@directionip.com

*Attorneys for Plaintiff TurboCode LLC*