25 Years of Turbo Codes: From Mb/s to beyond 100 Gb/s

Stefan Weithoffer*, Charbel Abdel Nour†, Norbert Wehn*, Catherine Douillard†, Claude Berrou†
*Department of Electrical and Computer Engineering, Technische Universität Kaiserslautern
Email: *{weithoffer, wehn}@eit.uni-kl.de, †{charbel.abdelnour, catherine.douillard, claude.berrou}@imt-atlantique.fr
†IMT Atlantique, Department of Electronics, Lab-STICC - UMR 6285

***Abstract*—In this paper, we demonstrate how the development of parallel hardware architectures for turbo decoding can be continued to achieve a throughput of more than 100 Gb/s. A new, fully pipelined architecture shows better error correcting performance for high code rates than the fully parallel approaches known from the literature. This is demonstrated by comparing both architectures for a frame size $K = 128$ LTE turbo code and a frame size $K = 128$ turbo code with parity puncture constrained interleaving. To the best of our knowledge, an investigation of the error correcting performance at high code rates of fully parallel decoders is missing from the literature. Moreover, place & route results for a case study implementation of the new architecture on 28 nm technology show a throughput of 102.4 Gb/s and an area efficiency of 4.34 Gb/s making it superior to reported implementations of other parallel decoder hardware architectures.**

***Keywords*—Forward Error Correction, Turbo decoder, LTE, High-throughput.**

I. Introduction

Turbo codes were not invented by an information theorist but by an electronic engineer who was wondering about the interest of implementing feedback – a fundamental concept in electronic design – in concatenated decoders. At that time (beginning of '90s), the most powerful channel coding scheme was actually given by the concatenation of a Reed-Solomon encoder and a convolutional encoder. Some researchers were already trying to improve the associated decoder performance by enabling bilateral exchanges between the component decoders (see [1] for instance with a claimed gain of about 0.5 dB on the signal to noise ratio). But due to the strong dissimilarity between the two coding principles, especially on the question of hard and soft decisions, it was not easy to take a large benefit from these back-and-forth exchanges.

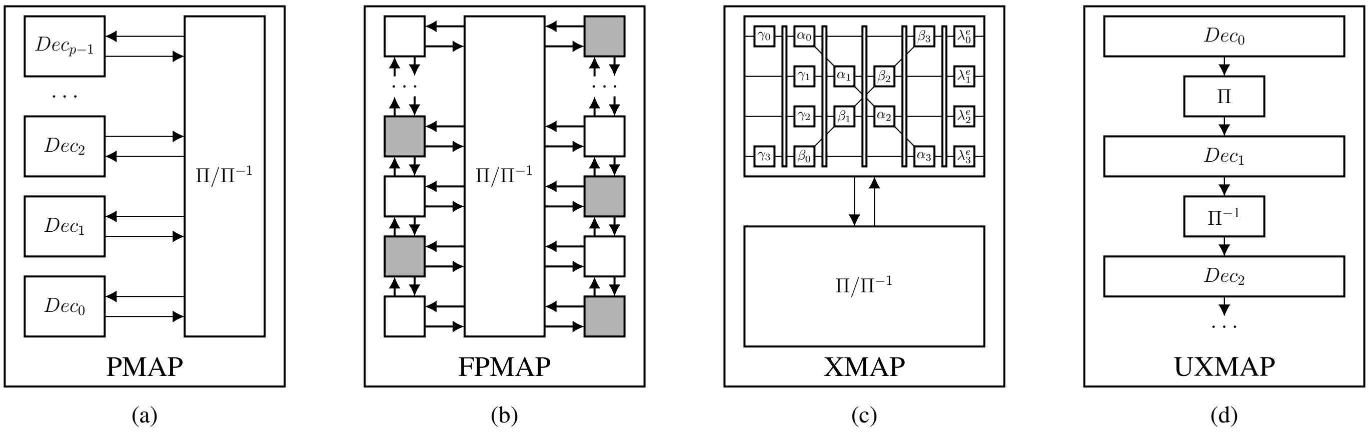
Meanwhile, interesting results were obtained on the soft-output Viterbi algorithm [2]–[4] which opened the way to efficient message passing between convolutional Viterbi-based decoders. It was then possible to associate two or more convolutional codes in original ways and eventually to introduce so-called parallel concatenation. The initial sought-for advantage of this parallel coding architecture was to simplify the clock management in the design of a possible demonstrator because the clocks of the outer and inner component decoders are not the same in serial concatenation, which is added complexity. It turned out that the performance was also greatly improved by this original construction, thanks to the additional introduction of extrinsic information, another new concept in the field of information theory. Actually, parallel concatenation offers better convergence than serial one in iterative decoding, but at the price of a reduced minimum Hamming distance (MHD), which was not detected at that time (bit error rates lower than 10^{-5} were not easy to simulate). Many works were then launched to obtain sufficient MHD for turbo codes.

Since the seminal work of Claude Shannon at the end of the '40s and the quasi-simultaneous invention of the transistor, information theory and electrical engineering have never ceased to work together to continuously improve information processing, including processes that have enabled the tremendous expansion of telecommunications. Today, microelectronic expertise is still a driver for the development of new architectures capable of supporting considerable throughputs. That's what this paper is about. The invention of turbo codes may also be seen as the illustration of a general principle in technology: the practical solution that is found to solve a complex problem (Shannon capacity in our case) is rarely given by the theory that was used to formalize it.

Turbo codes have, together with LDPC codes, played a most influential role in re-shaping our information centric society since their inception 25 years ago [5], [6]. They have been employed as channel codes for the 3G and 4G standards, which have brought forth the mobile internet, and will continue to be part of 5G through the evolution of LTE [7].

Along with the evolution of downlink data rates, the throughput requirements, i.e. the required amount of decoded bits per second, have evolved from less than 1 Mb/s for UMTS in 1999, 100 Mb/s for LTE in 2008 to Gb/s for current and planned releases of LTE-A Pro [7], which lead to new parallel architectures [8]–[10]. However, since the throughput requirements increased faster than the improvements provided by advances in microelectronics, a cross-layer approach which jointly considered code structure and implementation requirements was necessary.

For example, it became obvious, that, in order to allow higher throughputs, conflict-free interleavers were needed [11]. Thus, for LTE, *Quadrature Permutation Polynomial* (QPP) interleavers [12] were adopted, which allowed conflict free interleaving up to a parallelism of 64 for all LTE frame sizes [13]. Today, the highest throughputs in the order of tens of Gb/s are achieved by turbo decoder implementations with a *Fully Parallel MAP* (FPMAP) architecture which represents



Dec_{p-1}
Dec_2
Dec_1
Dec_0
Π/Π^{-1}
PMAP
FPMAP
γ_0 α_0 β_3 Λ_0^e
γ_1 α_1 β_2 Λ_1^e
γ_2 β_1 α_2 Λ_2^e
γ_3 β_0 α_3 Λ_3^e
XMAP
Π
Π^{-1}
UXMAP
(a)
(b)
(c)
(d)

Figure 1: Parallel Turbo Decoder Architectures

an extreme case of parallel processing of sub-blocks of size 1 [14], [15].

Increasing the parallelism at component decoder level, however, comes at the cost of a degradation in error correcting performance, especially when decoding at higher code rates. Moreover, employing multiple decoders to increase the throughput does not solve the additional challenge of tighter latency requirements [16].

Thus, even with the advances provided by microelectronics, achieving a throughput of more than 100 Gb/s with state-of-the-art turbo decoder hardware architectures while achieving good error correcting performance across a range of code rates is still an open question.

Early results of fully pipelined, iteration unrolled turbo decoding – a concept known from LDPC decoders [17], [18] – suggest that throughputs beyond 100 Gb/s are possible [19]. To this end, a investigation of the error correcting performance of FPMAP decoders at high code rates and a detailed discussion of a iteration unrolled, fully pipelined turbo decoder is missing from literature.

This work presents the first turbo decoder achieving a throughput of more than 100 Gb/s in 28 nm technology using an *Unrolled XMAP* (UXMAP) architecture (Section IV). The remainder of this paper is structured as follows: After giving an overview on state-of-the-art parallel turbo decoder hardware architectures in Section II, we compare the error correcting performance for the FPMAP and UXMAP architecture for different code rates in Section III and show how the error correcting performance for high code rates can be improved by employing carefully designed interleavers. In Section V, we give post place & route results for UXMAP and FPMAP decoders with frame size $K = 128$ and compare them with other high throughput implementations of different turbo decoder architectures. Section VI concludes the paper.

II. Turbo Decoder Hardware Architectures

The general turbo decoder structure consists of two component decoders connected through an interleaver/de-iterleaver [5]. They work cooperatively by exchanging extrinsic information Λ^e in an iterative loop. Each processing of one component decoder is counted as one *Half Iteration* (HI) and a complete run of the loop is counted as one *(full) ITeration* (IT). State-of-the-art parallel hardware architectures for turbo code decoding split the code blocks in smaller *sub-blocks* and employ either *spatial* or *functional* parallelization.

A. Parallel MAP Architecture

Turbo decoders with a *Parallel MAP* (PMAP) architecture use spatial parallelization and process sub-blocks on multiple sub-decoder cores as illustrated in Fig. 1 (a). Additionally, each sub-decoder core splits the sub-blocks further into smaller blocks called *windows*, to enable a parallel processing of the forward and backward recursions. Due to the splitting into sub-blocks, however, the state metrics at the sub-block and window borders need to be estimated to mitigate a decoding performance loss. With smaller sub-blocks and at higher code rates, the length of the necessary acquisition calculations is therefore increased, which in turn limits the maximum degree of parallelization [20]. Implementations with todays silicon technologies achieve a throughput in the order of single digit Gb/s [21]–[23].

B. Fully Parallel MAP Architecture

The FPMAP architecture [14] can be seen as an extreme case of the PMAP architecture, where the size of the sub-blocks is reduced to 1 and a shuffled decoding scheme [24] is used for an immediate exchange of the extrinsic information between the component decoders. This architecture is illustrated in Fig. 1 (b). For every trellis step and for both component decoders, there is a processing element which exchanges its state metrics and extrinsic information with the neighboring processing elements. Consequently, a complete turbo code iteration is processed in parallel in each clock cycle. In order to reduce the implementation complexity, it has been shown for LTE turbo codes that the processing can be divided into two groups of processing elements (grey and white in Fig. II (b)). By activating alternately the processing elements associated with odd and even bit positions, the area complexity

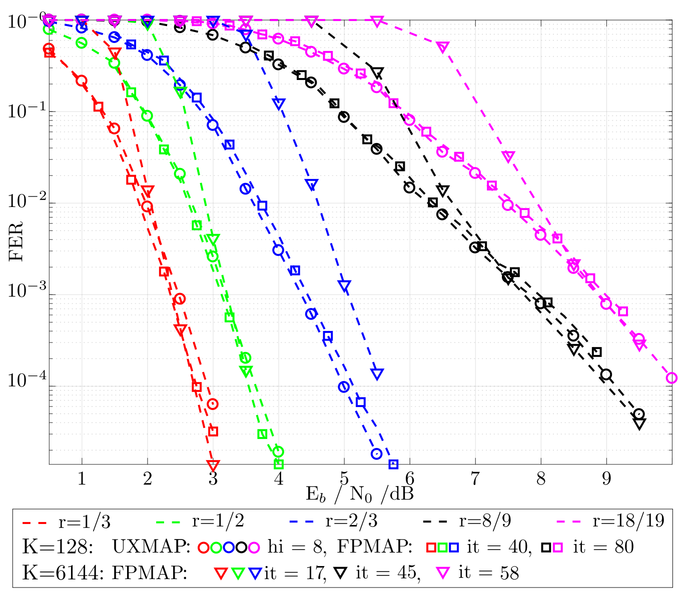


Figure 2: Comparison for LTE turbo code with $K = 128$.

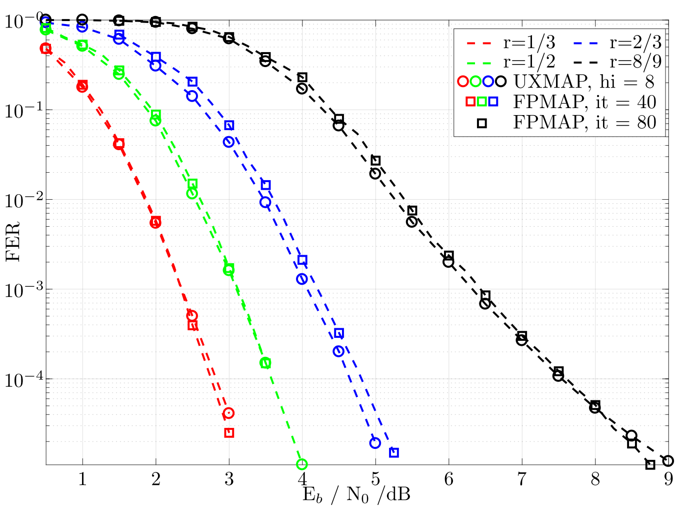


Figure 3: Comparison for a turbo code with PPC interleaver and frame size $K = 128$.

can be halved at the cost of an additional clock cycle per iteration. The FPMAP architecture has been demonstrated to allow a very high throughput of more than 15 Gb/s, however at the cost of a decreased area efficiency and an increased number of iterations for the same decoding performance [15], [25].

C. *Pipelined MAP Architecture*

A functional parallelization of the MAP algorithm by pipelining the recursive calculations of the state metric recursions leads to the XMAP architecture [9], [26]–[28]. It is named after the X-shaped decoder pipeline which is illustrated in Fig. 1 (c). The pipeline structure has been proven optimal with respect to the amount of state metric storage [29] and with current technologies, a throughput of over 1 Gb/s has been demonstrated [28]. However, the same limitations with respect to maximum degree of parallelization apply as for the PMAP architecture.

D. *Fully Pipelined Iteration Unrolled MAP Architecture*

Further pipelining of the decoding by unrolling the individual HI of the turbo decoding leads to the fully pipelined *Iteration Unrolled XMAP* (UXMAP) decoder architecture (Fig. 1 (d)), which processes complete code blocks in a pipeline. Note, that, in contrast to an early VLSI implementation of a turbo decoder, which also mentioned the concept of iteration unrolling [30], the UXMAP architecture is fully pipelined. Assuming a completely filled pipeline, this allows for the output of a complete decoded frame per clock cycle resulting in a very high throughput, which is only limited by the achievable clock frequency and frame size. The idea for this architecture was first presented in [19] but no detailed description, performance numbers or place & route results were given.

To achieve throughput in the order of 100 Gb/s, decoder architectures with extreme parallelism like the FPMAP and UXMAP are needed. Therefore, for the remainder of this paper, we will focus on those two architectures.

III. Decoding at different code rates

Applying different decoding architectures, the two considered decoders UXMAP and FPMAP do not achieve the same performance with the same set of simulation parameters. Any fair comparison should be performed at comparable performance levels. Fig. 2 shows the *Frame Error Rate* (FER) of UXMAP and FPMAP decoders for a LTE turbo code with frame size $K = 128$ and a quantization of 6 bits for the channel LLRs. When decoding for 4 iterations (UXMAP) and 40 iterations (FPMAP) respectively, both turbo decoders show similar performance for lower code rates. For the high code rates $8/9$ and $18/19$, however, the FER performance of the FPMAP decreases and the number of required iterations increases drastically (up to 80) which can be attributed to the scheduling of the decoding operations within the FPMAP algorithm. Indeed, for FPMAP, forward and backward state metrics exchange only a limited amount of information within one component decoder, largely penalizing the reliability of exchanged extrinsic information based on parity channel metrics. Consequently, for high code rates, the surviving (after puncturing) parity information from one trellis section needs several iteration steps to propagate to far away sections in the trellis. For the UXMAP decoder, this information influences a larger section of the trellis in each HI, because the processing is in essence that of a serial MAP processing.

This effect is still observed for larger frame sizes as seen also in Fig. 2, where the FER performance of a FPMAP with $K = 6144$ is added for comparison. Because of the larger frame size, only 17 FPMAP iterations achieve comparable FER performance to 4 UXMAP iterations at rate $1/3$. At high code rates, however, up to 58 FPMAP iterations are needed in order to match the UXMAP performance.

If high code rates are to be supported by the decoder, special care needs to be taken with respect to designing the puncturing patterns and interleaver.

A. *Parity Puncture Constrained (PPC) Interleavers*

The most widely-used interleaver families include the QPP interleaver [12], the *Dithered Relative Prime* (DRP) interleaver [31] and the *Almost Regular Permutation* (ARP) interleaver [32]. We will focus on ARP interleavers since it was shown in [33] that the ARP interleaver can provide the same interleaving properties as the other two interleaver types, guaranteeing minimum Hamming distance values at least as high as QPP and DRP interleavers.

Puncturing is generally used for code rate flexibility. Periodic puncturing patterns are favoured thanks to the offered advantages ranging from a simplified design process to the low complexity hardware implementation. In [34], it was observed that the puncturing of well-chosen systematic bits can improve the performance of turbo codes at high and low error rates especially for high coding rates. Lately in [35], it was shown that the reliability of extrinsic information related to an information bit depends on the position of the considered bit in the puncturing period and the puncturing or not of the corresponding parity. Moreover, it was observed that the extrinsic information computed from unpunctured parity positions is more reliable than the one generated from punctured parity positions. These observations led to the proposal of protograph-based interleavers [35], where a periodic strategy is applied through the interleaver, by connecting the positions with highly reliable extrinsic information to the positions with unreliable extrinsic information, more prone to errors. In addition to the associated error correction improvements, the introduced additional regularity through periodicity of both connection and puncturing patterns largely facilitates the support of code rate flexibility for hardware implementation. The FER performance of UXMAP and FPMAP for a turbo code with PPC interleaver and frame size $K = 128$ is shown in Fig. 3. In comparison to Fig. 2, both decoders show improvements for all investigated code rates. The largest gain in performance is observable for rate $8/9$, where there is an improvement of more than 1.5 dB at a FER of 10^{-3} for both decoders. However, the FPMAP decoder still requires 80 iterations to match the performance of the UXMAP decoder for rate $8/9$.

IV. Fully Pipelined Unrolled Decoder Architecture

In this section, we present a fully pipelined, iteration unrolled turbo decoder hardware architecture (UXMAP). Note that, in contrast to previously reported unrolled LDPC decoders [17], [18], our architecture is, without any extra effort, rate-compatible. Fig. 4 illustrates the pipelined architecture on the example of three X-shaped HI stages (*X-Stages*).

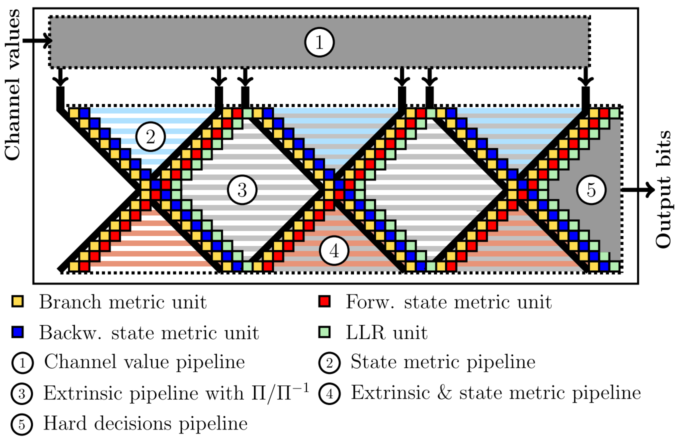

Channel values
1
2
3
4
5
Output bits
🟨 Branch metric unit
🟥 Forw. state metric unit
🟦 Backw. state metric unit
🟩 LLR unit
1 Channel value pipeline
2 State metric pipeline
3 Extrinsic pipeline with Π/Π^{-1}
4 Extrinsic & state metric pipeline
5 Hard decisions pipeline

Figure 4: Iteration unrolled pipelined decoder architecture.

A. *X-Stages*

The X-Stages realize a functionally parallelized processing of the serial MAP decoding of complete frames. For a block size K and radix 2, each X-Stage consists of $2 \cdot K$ branch metric units, K LLR units for the computation of the extrinsic information, forward and backward state metric units and state metric pipelines. To save area for the state metric pipeline, all state metrics are normalized to the state metric for state S_0 via subtractive normalization and only the remaining state metrics are sent to the state metric pipelines. Note, that K of the $2 \cdot K$ branch metric units are used for a re-computation of the branch metrics at the right side of each X-Stage which avoids area costly branch metric pipelines.

B. *Pipelines*

The different X-Stages are connected through the channel value and extrinsic pipelines which also realize the hardwired interleaved/de-interleaved information exchange between the X-Stages. Lastly, the hard decisions calculated in the last X-Stage are delayed by the hard decisions pipeline and given to the output so that one complete decoding result is generated per clock cycle once the pipeline is completely filled. Significant area saving in the pipelines - at the cost of an area increase in the X-Stages - is achieved by using a higher radix [36]. Increasing the number of trellis steps processed in parallel by the branch, state metric and LLR units from $k = 1$ to $k = 2$ (i.e. going from radix 2 to radix 4) leads to a reduction of the number of pipeline stages for all pipelines and thus translates to an area saving of about 50% and also halves the overall pipeline latency. However, investigations for radix 8 and radix 16 show, that there the area overhead in the computational units of the X-Stages supersedes the area savings in the pipelines.

V. Place & Route Results

We implemented the iteration unrolled architecture described in the previous section in VHDL and performed synthesis for a LTE turbo code with frame size $K = 128$ with a 28 nm Global Foundries FDSOI process. Since for this

	This Work		[23]	[37]	[38]	[21]	[15]	[22]	[28]
Architecture	UXMAP	FPMAP	PMAP				FPMAP	XMAP	
K	128		6144	6144	6144	6144	6144	6144	6144
Parallelism	128		64	32	64	6144	6144	64	32
n_{IT}	4	40	5.5	5.5	6	6	39	5.5	7
Technology	28 nm		90 nm$^{\natural\ \sharp}$	65 nm$^{\dagger\ \ddagger}$	65 nm$^{\dagger\ \ddagger}$	65 nm$^{\dagger\ \ddagger}$	65 nm$^{\dagger\ \ddagger}$	45 nm $^{\clubsuit\ \spadesuit}$	28 nm
Freq. $[MHz]$	800	500	625 (1000)	410 (1000)	400 (1000)	450 (1000)	100 (252)	600 (1000)	625
Throughput $[Gb/s]$	102.4	1.6	3.3 (5.29)	1.01 (2.47)	1.28 (4.78)	2.15 (2.78)	15.8 (39.86)	1.67 (3.2)	1.13
Area $[mm^2]$	23.61	1.04	19.75 (2.44)	2.49 (0.55)	8.3 (1.83)	7.7 (1.70)	109 (24.09)	2.43 (1.04)	0.49
Area Eff. $[Gb/s/mm^2]$	4.34	1.53	0.17 (2.17)	0.41 (4.49)	0.15 (1.74)	0.28 (2.81)	0.14 (1.65)	0.69 (2.68)	2.32

Table I: Comparison of implementation results for different turbo decoder architectures
Frequency scaling to 28 nm (Capped at 1000 MHz): $\natural$: 2.52; $\dagger$: 1.95; $\clubsuit$: 1.46; Area scaling to 28 nm: $\sharp$: 0.40; $\ddagger$:0.51; $\spadesuit$: 0.69

work the interleaving/de-interleaving is realized by hardwired connections in the pipeline, the area complexity can be expected to be virtually identical if a PPC interleaver is used. To allow a fair comparison, we furthermore re-implemented the FPMAP architecture reported in [15] for $K = 128$ and placed & routed both designs for the same process with worst case PVT (*Process/Voltage/Temperature*) constraints. The results for our re-implementation of the FPMAP architecture for $K = 128$ show an area consumption of ≈ 1 mm^2. While [15] used 4- and 6-bit quantization, respectively, we chose 6- and 8-bit for fair comparison. When compared to the UXMAP implementation, the FPMAP with $K = 128$ is outperformed by a factor of 64 in terms of throughput and a factor of 2.8 for the area efficiency in terms of Gb/s/mm^2.

Table I compares our place & route results for UXMAP and FPMAP implementations with implementation results for the PMAP, FPMAP and XMAP decoder architectures with a throughput of more than 1 Gb/s reported in the literature. Also included in Table I is a scaling to 28 nm technology. To this end, we cap the frequency scaling to a reasonable 1000 MHz, which allows to preserve single cycle accesses to SRAM.

All reference implementations feature a LTE turbo code with larger frame size $K = 6144$. The larger frame size leads to a much steeper slope in the waterfall region compared to the LTE turbo code with frame size $K = 128$. Consequently, all compared PMAP decoders and the XMAP decoder can be expected to achieve the same FER performance with less than the maximum number of iterations specified. However, even with a reduced number of iterations and considering a scaling to 28 nm technology, neither the PMAP decoders, not the XMAP decoder come close to even a throughput of 15 Gb/s. Only the FPMAP implementation from [15] is estimated to achieve a throughput of roughly 40 Gb/s when scaled to 28 nm technology. Nevertheless, a reduction of iterations leads to a considerable FER performance penalty for the FPMAP at high code rates (see Fig. 2).

With respect to area efficiency, our UXMAP implementation achieves a very good 4.34 Gb/s/mm^2, a value that is only matched by the full custom design from [37] when scaling it to 28 nm. Note, that the area efficiency in terms of Gb/s/mm^2 will be slightly higher for all reference implementations if the difference in FER performance due to the larger frame size is accounted for by performing less decoding iterations. On the other hand, and in contrast to the UXMAP architecture which supports a streaming approach due to its pipelined structure, additional area for buffering will be required for PMAP, XMAP and FPMAP. This buffering, necessary because of the latency of the iterative processing, will lower the architecture efficiency for very high throughputs. Fig. 5 shows the layout picture of the UXMAP decoder with 8 X-stages and frame size $K = 128$.

VI. Conclusion

In this work, we presented an implementation of a fully pipelined iteration unrolled turbo decoder. For a fair comparison, we re-implemented the VLSI implementation of a FPMAP architecture for the same frame size. With a throughput of 102.4 Gb/s and an area consumption of 23.61 mm^2, our rate flexible UXMAP clearly outperforms the re-implemented FPMAP as well as previously reported architectures in terms of throughput and area efficiency. We show, that the UXMAP architecture is superior to the FPMAP architecture for high code rates and demonstrate that FER performance can be further improved by employing PPC turbo codes. Future work will focus on re-introducing flexibility with respect to frame sizes to the architecture.

Acknowledgement

We gratefully acknowledge financial support by the DFG (project-ID: 2442/8-1) and the EU (project-ID: 760150-EPIC).

References

[1] E. Paaske. Improved decoding for a concatenated coding system recommended by CCSDS. *IEEE Transactions on Communications*, 38(8):1138–1144, Aug 1990.

[2] G. Battail. Coding for the Gaussian channel - The promise of weighted-output decoding. *International Journal of Satellite Communications*, 7:183–192, September 1989.

[3] J. Hagenauer and P. Hoeher. A Viterbi Algorithm with Soft-Decision Outputs and its Applications. In *Proc. 1989 Global Telecommunications Conference (GLOBECOM '89)*, pages 1680–1686, Dallas, Texas, USA, November 1989.

[4] C. Berrou, P. Adde, E. Angui, and S. Faudeil. A Low Complexity Soft-Output Viterbi Decoder Architecture. In *Proc. 1993 International Conference on Communications (ICC '93)*, pages 737–740, Geneva, Switzerland, May 1993.

[5] C. Berrou, A. Glavieux, and P. Thitimajshima. Near Shannon Limit Error-Correcting Coding and Decoding: Turbo-Codes. In *Proc. 1993 International Conference on Communications (ICC '93)*, pages 1064–1070, Geneva, Switzerland, May 1993.

[6] D.J.C. MacKay and R. Neal. Near Shannon limit performance of Low-Density Parity-Check Codes. *Electronic Letters*, 32:1645–1646, 1996.

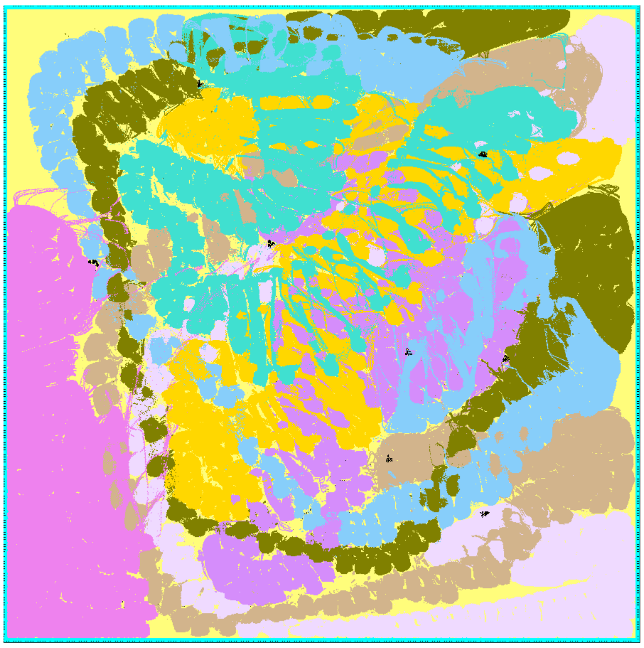


Figure 5: Layout of the UXMAP decoder. Colored clouds represent the different X-stages with the last X-stage notably in the lower left. The channel value pipeline is shown in yellow in the background.

[7] Third Generation Partnership Project. *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (3GPP TS 36.213 version 13.1.0 Release 13)* , April 2016.

[8] M. Bickerstaff, L. Davis, C. Thomas, D. Garrett, and C. Nicol. A 24Mb/s Radix-4 LogMAP Turbo Decoder for 3GPP-HSDPA Mobile Wireless. In *Proc. 2003 IEEE International Solid-State Circuits Conference (ISSCC '03)*, pages 150 – 151,484, San Francisco, CA, USA, February 2003.

[9] M. May, C. Neeb, and N. Wehn. Evaluation of High Throughput Turbo-Decoder Architectures. In *Proc. IEEE International Symposium on Circuits and Systems ISCAS 2007*, pages 2770–2773, New Orleans, USA, May 2007.

[10] J.-H. Kim and I.-C. Park. A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE. In *Proc. IEEE Custom Integrated Circuits Conference CICC '09*, pages 487–490, September 2009.

[11] M. J. Thul. Exploration of the Interleaver Bottleneck in Iterative Decoding using Parallel Architectures and a Proposal How to Overcome It. Technical report, Institute of Microelectronic Systems, Department of Electrical Engineering and Information Technology, University of Kaiserslautern, March 2001.

[12] J. Sun and O. Y. Takeshita. Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Transactions on Information Theory*, 51(1):101–119, January 2005.

[13] Third Generation Partnership Project. *3GPP TS 36.212 V8.5.0; 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 8)*, December 2008.

[14] R. G. Maunder. A Fully-Parallel Turbo Decoding Algorithm. *IEEE Transactions on Communications*, 63(8):2762–2775, Aug 2015.

[15] A. Li, L. Xiang, T. Chen, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo. VLSI Implementation of Fully Parallel LTE Turbo Decoders. *IEEE Access*, 4:323–346, 2016.

[16] G.P. Fettweis. The Tactile Internet: Applications and Challenges. *Vehicular Technology Magazine, IEEE*, 9(1):64–70, March 2014.

[17] P. Schläfer, N. Wehn, T. Lehnigk-Emden, and M. Alles. A New Dimension of Parallelism in Ultra High Throughput LDPC Decoding. In *IEEE Workshop on Signal Processing Systems (SIPS)*, Taipei, Taiwan, 2013.

[18] A. Balatsoukas-Stimming, M. Meidlinger, R. Ghanaatian, G. Matz, and A. Burg. A Fully-Unrolled LDPC Decoder Based on Quantized Message Passing. *arXiv preprint arXiv:1510.04589*, 2015.

[19] S. Weithoffer, M. Herrmann, C. Kestel, and N. Wehn. Advanced wireless digital baseband signal processing beyond 100 Gbit/s. In *2017 IEEE International Workshop on Signal Processing Systems (SiPS)*, pages 1–6, Oct 2017.

[20] S. Weithoffer, K. Kraft, and N. Wehn. Bit-level Pipelining for Highly Parallel Turbo-Code Decoders: A Critical Assessment. In *IEEE Africon 2017*, pages 138–143, 2017.

[21] T. Ilnseher, F. Kienle, C. Weis, and N. Wehn. A 2.12Gbit/s Turbo Code Decoder for LTE Advanced Base Station Applications. In *2012 7th International Symposium on Turbo Codes and Iterative Information Processing (ISTC) (ISTC 2012)*, Gothenburg, Sweden, August 2012.

[22] G. Wang, H. Shen, Y. Sun, J. R. Cavallaro, A. Vosoughi, and Y. Guo. Parallel Interleaver Design for a High throughput HSPA+/LTE Multi-Standard Turbo Decoder. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(5):1376–1389, May 2014.

[23] R. Shrestha and R. P. Paily. High-Throughput Turbo Decoder With Parallel Architecture for LTE Wireless Communication Standards. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(9):2699–2710, Sept 2014.

[24] Z. Juntan and M. P. C. Fossorier. Shuffled Iterative Decoding. *IEEE Transactions on Communications*, 53(2):209–213, February 2005.

[25] A. Li, P. Hailes, R. G. Maunder, B. M. Al-Hashimi, and L. Hanzo. 1.5 Gbit/s FPGA Implementation of a Fully-Parallel Turbo Decoder Designed for Mission-Critical Machine-Type Communication Applications. *IEEE Access*, 4:5452–5473, 2016.

[26] A. Worm, H. Lamm, and N. Wehn. Design of Low-Power High-Speed Maximum a Posteriori Decoder Architectures. In *Proc. Design, Automation and Test in Europe Conference and Exhibition 2001*, pages 258–265, Munich, Germany, March 2001.

[27] M. May, T. Ilnseher, N. Wehn, and W. Raab. A 150Mbit/s 3GPP LTE Turbo Code Decoder. In *Proc. Design, Automation and Test in Europe, 2010 (DATE '10)*, pages 1420–1425, March 2010.

[28] S. Weithoffer, F. Pohl, and N. Wehn. On the applicability of trellis compression to Turbo-Code decoder hardware architectures. In *2016 9th International Symposium on Turbo Codes and Iterative Information Processing (ISTC)*, pages 61–65, Sept 2016.

[29] M. M. Mansour and N. R. Shanbhag. VLSI architectures for SISO-APP decoders. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 11(4):627–650, August 2003.

[30] M. Jezequel and P. Penard. Turbo4: a high bit-rate chip for turbo code encoding and decoding. In *IEE Colloquium on Turbo Codes in Digital Broadcasting - Could It Double Capacity? (Ref. No. 1999/165)*, pages 4/1–4/5, 1999.

[31] S. Crozier and P. Guinand. Distance Upper Bounds and True Minimum Distance Results for Turbo-Codes Designed with DRP Interleavers. In *Proc. 3rd International Symposium on Turbo Codes & Related Topics*, Brest, France, September 2003.

[32] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jézéquel. Designing good permutations for turbo codes: towards a single model. In *Proc. IEEE International Conference on Communications*, volume 1, pages 341–345, June 2004.

[33] R. Garzón Bohórquez, C. A. Nour, and C. Douillard. On the Equivalence of Interleavers for Turbo Codes. *IEEE Wireless Communications Letters*, 4(1):58–61, Feb 2015.

[34] K. Gracie and S. Crozier. Convergence performance and EXIT analysis of 4-state partially-systematic Turbo codes. In *2008 5th International Symposium on Turbo Codes and Related Topics*, pages 414–419, Sept 2008.

[35] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard. Protograph-Based Interleavers for Punctured Turbo Codes. *IEEE Transactions on Communications*, 66(5):1833–1844, May 2018.

[36] Yuping Zhang and K. K. Parhi. High-Throughput Radix-4 logMAP Turbo Decoder Architecture. In *Proc. Fortieth Asilomar Conference on Signals, Systems and Computers ACSSC '06*, pages 1711–1715, October 2006.

[37] C. Roth, S. Belfanti, C. Benkeser, and Qiuting Huang. Efficient Parallel Turbo-Decoding for High-Throughput Wireless Systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 61(6):1824–1835, June 2014.

[38] Y. Sun and J.R. Cavallaro. Efficient hardware implementation of a highly-parallel 3GPP LTE/LTE-advance turbo decoder. *Integration VLSI Journal*, 2010.