Implementation Trade-offs of Soft-Input Soft-Output MAP Decoders for Convolutional Codes

Christoph Studer, *Member, IEEE*, Schekeb Fateh, *Student Member, IEEE*,
Christian Benkeser, *Member, IEEE*, and Qiuting Huang, *Fellow, IEEE*

***Abstract*—Soft-input soft-output (SISO) maximum a-posteriori (MAP) decoders for convolutional codes (CCs) are an integral part of many modern wireless communication systems. Specifically, SISO-MAP decoding forms the basis for turbo decoders, as, e.g., specified for HSDPA or 3GPP-LTE, or for iterative detection and decoding in multiple-input multiple-output wireless systems, such as IEEE 802.11n. In this paper, we investigate the silicon-area, throughput, and energy-efficiency trade-offs associated with SISO-MAP decoders based on the algorithm developed by Bahl, Cocke, Jelinek, and Raviv (BCJR). To this end, we develop radix-2 and radix-4 architectures for high-throughput SISO-MAP decoding of CCs having 4, 8, 16, 32, and 64 states and present corresponding implementation results in 180 nm, 130 nm, and 90 nm CMOS technology. We validate technology-scaling rules and finally demonstrate the use of the presented trade-off analysis by identifying the key design parameters for parallel turbo-decoder implementations.**

***Index Terms*—Very-large scale integration (VLSI), wireless communication, soft-input soft-output (SISO) maximum a-posteriori (MAP) decoding, convolutional codes, turbo codes and decoding, iterative decoding.**

I. Introduction

RELIABLE data transmission in wireless communication systems requires sophisticated channel coding schemes and corresponding high-throughput, low-area, and energy-efficient decoder implementations. Convolutional codes (CCs), used in stand-alone form [3] or as part of turbo codes [4], are among the most popular codes used in current and next-generation wireless communication standards, such as HSDPA [5], 3GPP-LTE [6], LTE-Advanced [7], or IEEE 802.11n [8]. CCs and turbo codes are of significant practical interest due to the fact that they offer excellent error-correction performance and can be implemented to achieve high throughput, while being efficient in terms of silicon area and power consumption [9]–[16]. Since the advent of turbo codes [4], *iterative decoding* algorithms relying on CCs, became a key enabler for wireless communication systems operating close to the Shannon limit. CCs have also been considered for wireless communication systems employing iterative detection and decoding, i.e., where reliability information is exchanged iteratively between a detector and the channel decoder. In particular, iterative detection and decoding in multiple-input multiple-output (MIMO) wireless systems [17] or systems exhibiting inter-symbol interference [18] is becoming an integral part of future transceiver designs because it is an efficient means to substantially improve the throughput and quality-of-service (i.e., link reliability, coverage, and range) compared to non-iterative decoding schemes (see, e.g., [19] and the references therein).

A. The need for a systematic trade-off analysis

Decoding of CCs is usually performed by the Viterbi algorithm (VA) [20], which is able to efficiently compute the most likely transmitted data sequence. Corresponding VLSI implementations have been shown to achieve high throughput at low silicon area [21]–[24]. However, for iterative decoding schemes, reliability information in the form of log-likelihood ratios (LLRs) for the transmitted bits needs to be computed. This requirement inhibits the use of Viterbi algorithm and necessitates soft-input soft-output (SISO) maximum a-posteriori (MAP) decoding algorithms. The algorithm that has become de-facto standard for SISO-MAP decoding was proposed by Bahl, Cocke, Jelinek, and Raviv (BCJR) [25] and was shown to be well-suited for the implementation in VLSI [10], [12], [16]. While a large number of BCJR decoders have been developed for turbo decoding [12], [16], most publications do *not* provide corresponding implementation results and lack in-depth investigation into the underlying silicon area, throughput, and energy efficiency trade-offs. A systematic analysis of these trade-offs is, however, of high practical benefit for the development of turbo decoders in order to identify the optimum architectural choices for a given throughput, area, or power constraint. Furthermore, architectures and implementations of BCJR decoders supporting a large number of states (i.e., 32-states and beyond) have been ignored in the literature,

The 180 nm CMOS implementation results have been presented in part in the Ph.D. Theses of C. Studer [1] and C. Benkeser [2].

C. Studer was with the Dept. Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland, and is now with the Dept. of Electrical and Computer Engineering, Rice University, Houston, TX 77005, USA (e-mail: studer@rice.edu).

S. Fateh, C. Benkeser, and Q. Huang are with the Dept. Information Technology and Electrical Engineering, ETH Zurich, 8092 Zurich, Switzerland (e-mail: fateh@iis.ee.ethz.ch; benkeser@iis.ee.ethz.ch; huang@iis.ee.ethz.ch).

The authors would like to thank S. Belfanti, M. Brändli, F. Gürkaynak, B. Muheim, D. Riha, and S. Schläpfer for their assistance during the ASIC designs. We gratefully acknowledge the support of A. Burg, N. Felber, W. Fichtner, and H. Kaeslin, and we would like to thank the anonymous reviewers for their valuable comments, which helped to improve the exposition of our results.

This project was supported in part by the Swiss National Science Foundation (SNSF) under Grant PA00P2-134155 and the Hasler Stiftung.

Digital Object Identifier XXX-XXX-XXX

TABLE I
MAXIMUM FREE-DISTANCE CODES AND REDUCTION OF 2-INPUT MAX UNITS IN THE LCU USING PARTIAL MAXIMUM SHARING (PMS)

K	S	d_{min}	Gen. poly. $\mathbf{p}$	Initial #	PMS #	Reduction
3	4	5	$[5_o\ 7_o]$	18	14	22.2%
4	8	6	$[15_o\ 17_o]$	42	22	47.6%
5	16	7	$[23_o\ 35_o]$	90	38	57.7%
6	32	8	$[53_o\ 75_o]$	186	70	62.4%
7	64	10	$[133_o\ 171_o]$	378	134	64.6%

even though the implementation of iterative detection and decoding, e.g., for IEEE 802.11n, necessitates 64-state SISO-MAP decoder circuits [19].

B. Contributions

Inspired by the trade-off analysis for turbo-decoder circuits in [26], we perform an in-depth investigation of the implementation trade-offs associated with high-throughput SISO-MAP decoders and explore their suitability for codes having a large number of states. To this end, we develop reference radix-2 and radix-4 decoder architectures for CCs having 4, 8, 16, 32, and 64 states based on the BCJR algorithm. To achieve high throughput at low silicon area and power consumption, we employ windowing, modulo normalization, and introduce a novel method referred to as partial maximum sharing, which is the key for efficient implementation of BCJR decoders supporting 32 states and beyond. We provide silicon area, throughput, and energy-efficiency results for 180 nm, 130 nm, and 90 nm CMOS technology and systematically investigate the associated implementation trade-offs. We furthermore present first-order models for the throughput, silicon area, and energy efficiency to validate technology-scaling rules. Finally, we show how the results presented in this paper enable us to identify the key design parameters of parallel turbo-decoder designs for modern and next-generation wireless systems.

C. Outline of the paper

The remainder of the paper is organized as follows. Section II summarizes the BCJR algorithm for SISO-MAP decoding. The VLSI architecture along with the corresponding optimizations is described in Section III. Implementation results and a comparison to existing SISO-MAP decoder designs can be found in Section IV. The implementation trade-offs and technology-scaling rules are investigated in Section V. A case study that makes use of our results is provided in Section VI. We conclude in Section VII.

II. SISO-MAP DECODING FOR CONVOLUTIONAL CODES

In this section, we briefly summarize the key properties of CCs and detail the SISO-MAP decoding algorithm which builds the foundation of the decoder implementations used for the trade-off analysis provided in Section V.

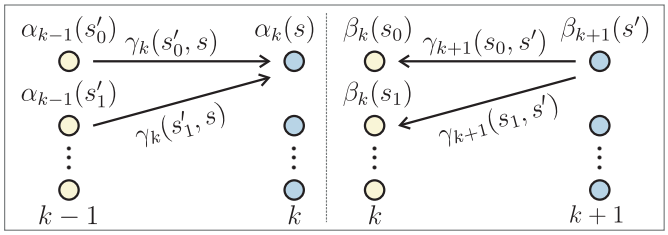


Fig. 1. Radix-2 state-metric recursions (left: forward; right: backward).

A. Convolutional Codes and Trellis Representation

Convolutional codes [3] are generated by feeding L binary-valued information bits $x_k \in \{0, 1\}$, $k = 1, \ldots, L$, into a shift register of length ν. The coded bits $c_{k,b} \in \{0, 1\}$, $\forall k$ and $b = 1, 2$, of a rate-1/2 CC[1] are generated by summations in GF(2) over well-defined values contained in the shift-register. The coded bits are then transmitted over a (wireless) communication channel and the receiver computes reliability information in form of log-likelihood ratios (LLRs) $L(c_{k,b})$, which represent the likelihood of $c_{k,b}$ being a binary one or zero [28].

1) Maximum free-distance codes: The error-rate performance of CCs is determined by the constraint length $K = \nu+1$ and the generator polynomial $\mathbf{p} = [g_1\ g_2]$ defining the connections to the GF(2) adders [27]. Minimization of the error-rate performance for a given constraint length K is achieved by selecting $\mathbf{p}$ such that minimum free Hamming distance d_{min} between the all-zero codeword and any other codeword is maximized. Table I shows the rate-1/2 maximum free distance codes from [27] investigated in the remainder of the paper.

2) Trellis representation: Convolutional codes can be represented by a *trellis diagram* (see Fig. 1) consisting of states and branches. States, denoted by $s \in \{0, \ldots, S-1\}$, correspond to the state of the shift register and the number of states is given by $S = 2^\nu$. Each state is associated with so-called state metrics $\alpha_k(s)$ and $\beta_k(s)$; branches are associated with the branch metrics $\gamma_k(s', s)$, where s' and s corresponds to a given state at trellis step $k-1$ and k, respectively.

B. The max-log M-BCJR algorithm

The BCJR algorithm [25], which builds upon the Viterbi algorithm [20], is the de-facto standard method for SISO-MAP decoding, as it allows for the efficient computation of the LLRs in hardware. The algorithm traverses the trellis (see Fig. 1) in both forward and backward directions to compute the state metrics $\alpha_k(s)$ and $\beta_k(s)$ in a recursive way. The original BCJR algorithm [25] requires i) the computation of transcendental functions and ii) a large amount of memory. Therefore, a hardware-friendly variant known as the *max-log M-BCJR algorithm* is typically employed in practical systems.

1) Computation of max-log LLRs: To avoid the evaluation of transcendental functions in VLSI, the max-log approxima-

[1]We exclusively consider rate-1/2 codes; higher code rates can, e.g., be obtained by means of puncturing [27].

tion to the forward state-metric recursion is applied [29]

$$\alpha_k(s) = \max\left\{\alpha_{k-1}(s'_0) + \gamma_k(s'_0, s), \alpha_{k-1}(s'_2) + \gamma_k(s'_2, s)\right\} \quad (1)$$

where s'_0 and s'_1 correspond to the two predecessor states at trellis-step $k-1$ of the state s (cf. Fig. 1). The backward state metrics $\beta_k(s')$ are computed similarly to (1) but in the opposite direction. After the computation of *all* forward and *all* backward state metrics, the LLRs associated with the transmitted information bits can be computed using the max-log approximation:

$$L(x_k) \approx \max_{\mathcal{S}(s',s;0)} \left\{\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)\right\} - \max_{\mathcal{S}(s',s;1)} \left\{\alpha_{k-1}(s') + \gamma_k(s', s) + \beta_k(s)\right\}. \quad (2)$$

Here, the sets $\mathcal{S}(s', s; 0)$ and $\mathcal{S}(s', s; 1)$ contain all state transitions (s', s) for which $x_k = 0$ and $x_k = 1$, respectively. Iterative MIMO decoding [17] and turbo equalization [18] additionally require the computation of the LLRs associated with the coded bits $L(c_{k,b})$, which is achieved analogously to (2) by maximizing over the sets $\mathcal{S}'(s', s; b; 0)$ and $\mathcal{S}'(s', s; b; 1)$ containing the state transitions for which $c_{k,b} = 0$ and $c_{k,b} = 1$, respectively.

2) Windowing: The algorithm outlined above requires storage of either *all* forward or *all* backward state metrics, which typically results in an excessive amount of memory. To significantly reduce the memory requirements, *windowing* is employed [30]. With this approach, the trellis is processed in small windows of M trellis-steps and the LLRs are computed only on the basis of the state and branch metrics within this window.[2] Specifically, the forward recursion computes and stores the M forward state metrics $\alpha_k(s)$ belonging to the mth window. For the backward recursion, which progresses from the end of a window to its beginning, suitable initial values need to be generated. To this end, a *dummy* backward recursion is carried out in the successor window $m+1$. The LLR-values are then computed simultaneously with the backward state metric recursion and with the aid of the buffered forward state metrics. An additional benefit of windowing is the fact that it enables parallel processing of the trellis, which is commonly used to achieve high decoding throughput (see Section VI for the details).

III. VLSI Architecture

In this section, we describe a parametrizable max-log M-BCJR architecture, which is used to investigate the trade-off analysis and to validate technology scaling rules in Section V. The architecture corresponds to an area- and throughput-optimized version of the M-BCJR decoder architectures integrated in [12], [16] for high-throughput and low-power decoding of turbo codes.

A. High-level VLSI architecture

Fig. 2 depicts the high-level architecture of the max-log M-BCJR decoder. The input buffers 1 and 2 serve as a temporary cache for the incoming LLRs. Since only $2 \times M$ LLRs need to be stored in total, these buffers are realized by arrays of latches (offering one simultaneous read and write access per clock cycle) as they i) require a similar amount of area and power consumption compared to dual-port S-RAM macrocells [31] and ii) simplify placing-and-routing during the back-end design. The branch-metric computation units (denoted by G in Fig. 2) are used to compute the $\gamma_k(s', s)$. To achieve high throughput, the forward, backward, and dummy-backward state metric recursions are implemented in three parallel units, which contain one add-compare-select (ACS) circuit, either radix-2 or radix-4, for each of the S trellis states. Windowing is implemented as follows: The forward state metric recursion unit computes the $\alpha_k(s)$ for each window m, which are buffered in the α-memory.[3] This memory stores $S \times M$ branch metrics and is realized by either one or several two-port S-RAM macrocells instances. For decoders supporting a large number of states (i.e., $S \geq 16$), we partition the α-memory into multiple instances (see Table II for the details). The backward state-metric recursion unit computes the $\beta_k(s)$ in window $m-1$. Initial values required at the beginning of each window are generated by the dummy state-metric recursion unit in window $m+1$. The buffered forward state metrics, intermediate results obtained in the backward recursion, and the branch metrics are used to compute the LLR values $L(c_{k,b})$, $b = 1, 2$, and $L(x_k)$ in the LLR computation unit (LCU).

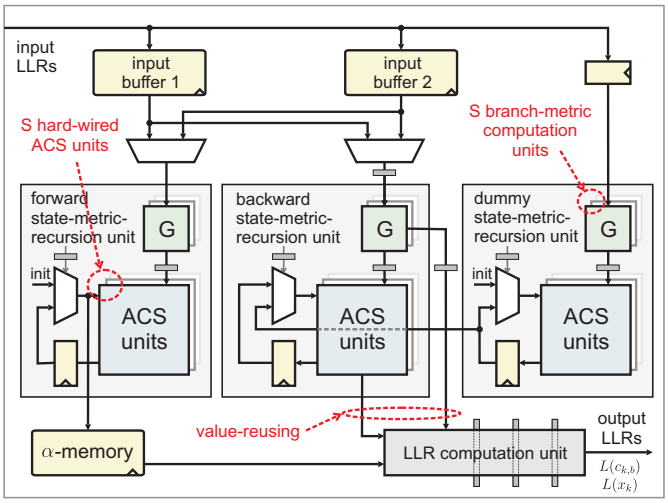


Fig. 2. High-level VLSI architecture of the implemented max-log M-BCJR decoders (thin grey boxes indicate pipeline registers).

B. State-metric recursion and ACS units

In order to maximize the throughput, the critical path of the decoder must be in the state-metric recursion units because their recursive nature of the algorithm inhibits pipelining. To this end, we employ *modulo normalization* [33], which results in significant shortening of the critical path compared to solutions employing costly renormalization circuitry [12],

[2]Considering windows of size $M = 5K$ was shown to achieve close-to-optimal performance for the Viterbi algorithm [27] and also provides near-optimal LLRs for rate-1/2 codes using the max-log M-BCJR algorithm.

[3]In order to further reduce the storage requirements for the state metrics, compression methods, such as the ones proposed in [32], can be used.

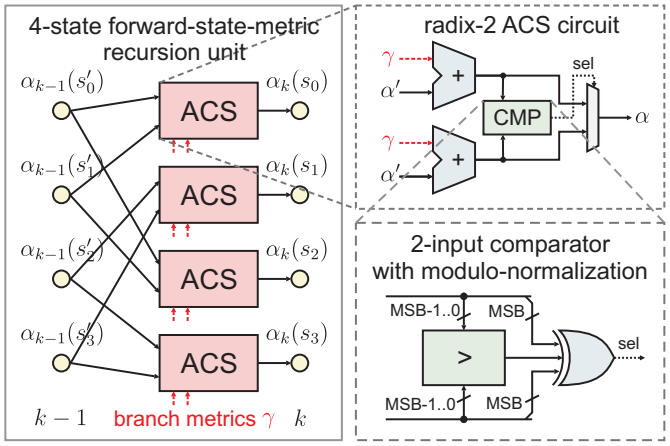


Fig. 3. Left: 4-state forward state-metric recursion unit. Right: Radix-2 ACS unit with a 2-input modulo normalization comparator. The comparison circuit shown at the bottom right consists of a comparator and a 3-input XOR gate.

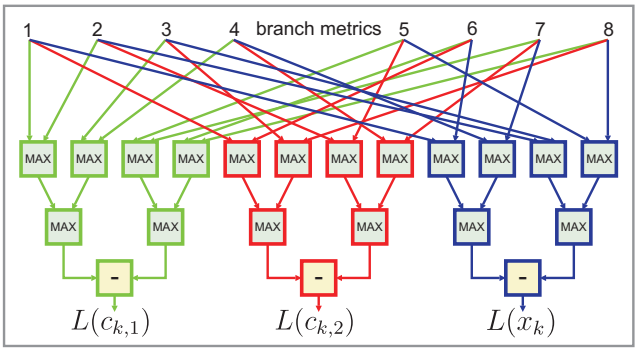
[16]. More specifically, modulo normalization relies on the fact that the difference between any pair of branch metrics (at a given state k) is bounded, which can be exploited to implement the state-metric renormalization with a controlled overflow. This approach does *not* cause any performance loss (given the number of bits to represent the state metrics is large enough; see [33] for details) and only requires an additional 3-input XOR gate in each ACS unit to take into account the overflows. Fig. 3 depicts a corresponding radix-2 ACS circuit.

We furthermore perform pipelining in all *non-recursive* units, i.e., just after the branch-metric computation units and within the LCU (see Fig. 2). If desired, the throughput can further be increased by performing radix-4 state-metric recursions [34]. This approach processes two trellis-steps per clock cycle, skipping the odd-numbered trellis steps. Specifically, the forward state metrics $\alpha_k(s)$ are computed from its *four* admissible predecessor states s_0'', s_1'', s_2'', and s_3'' (at step $k-2$) using a radix-4 ACS operation similar to (1). The radix-4 ACS units can be implemented efficiently by six parallel comparator units followed by a look-up table to control the output multiplexer (see [16] for circuit details).

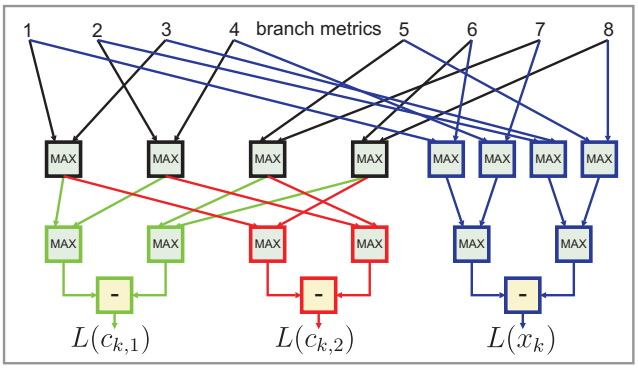
C. LLR computation unit with partial maximum sharing

The LCU computes the LLRs associated with the coded bits $L(c_{k,b})$ and the information bits $L(x_k)$ (see Section II-B1 and Fig. 2). LLR computation is realized by six maximization trees (two trees are required per LLR output) consisting only of 2-input maximization (MAX) units. Selection of the maximum input is carried out with a modulo-normalization comparator as depicted in Fig. 3. To keep the critical path within the state-metric recursion units (see Section III-B), the maximization trees in the LCU have been pipelined; one pipeline stage per tree layer is necessary.

A straightforward tree implementation for LLR computation leads to a large number of MAX units and pipeline registers, especially for 32-state and 64-state CCs. The complexity of the maximum trees contribute substantially to the overall silicon area and causes routing-congestion problems during the back-end design. In order to avoid both issues, we employ a novel



(a) Straightforward LLR computation using 2-input MAX units.



(b) LLR computation unit using PMS.

Fig. 4. Example of partial maximum sharing (PMS) applied to an unpipelined 4-state LLR computation unit.

technique referred to as partial maximum sharing (PMS). This approach re-uses partial maximization results required for computation of the LLRs $L(c_{k,b})$, $b = 1, 2$, to reduce the number of MAX units in the LCU. More specifically, careful inspection of the six maximization trees shows that certain intermediate results can be found in different trees. Since only one of these intermediate results needs to be computed, one can obviously avoid redundant computations. To implement PMS, we wrote a Matlab script, which identifies intermediate results occurring in different maximization trees and automatically generates a new LCU delivering the *same* LLRs while requiring less MAX units than a straightforward LCU implementation.

Fig. 4 illustrates an application of PMS to a 4-state LCU and Table I shows the savings (in terms of MAX units) achieved when using PMS for all codes considered in the paper. We emphasize that the number of MAX units required by a 64-state BCJR decoder can be reduced by almost 2/3, whereas the critical path remains virtually unaffected.[4] Corresponding synthesis results have shown that PMS reduces the silicon area of the 64-state LCU by 0.58× compared to a straightforward implementation, which results in 0.87× smaller area of the whole M-BCJR decoder. Furthermore, PMS effectively mitigates routing-congestion problems, which is crucial for high-throughput implementations of BCJR decoders supporting 32-state CCs and beyond.

[4]Even though the fan-out after each 2-input MAX unit slightly increases, the use of pipelining avoids any detrimental impact on the critical path.

D. Necessary modifications for decoding of turbo codes

The base architecture described above is able to decode *arbitrary* rate-1/2 CCs of a given constraint length and can easily be optimized for the use in turbo decoders. Specifically, the support of *recursive* CCs only requires different interconnections in the state-metric recursion units, which neither affects the critical path nor the silicon area. Furthermore, MAP decoders for turbo decoding require only the computation of the LLRs associated with the information bits $L(x_k)$ (see [12], [16]) and, therefore, the four maximization trees required for computation of $L(x_{k,b})$, $b = 1, 2$, can be omitted (cf. Fig. 4). Consequently, max-log M-BCJR implementations for turbo decoding require, in general, smaller silicon area while achieving the same throughput.

IV. Key Characteristics of the M-BCJR Designs

In order to assess the implementation trade-offs associated with max-log M-BCJR decoders, five radix-2 architectures with 4, 8, 16, 32, and 64 states (for the codes in Table I) have been implemented. To explore the differences between radix-2 and radix-4 designs, an additional radix-4 variant for the 8-state code, which is used in turbo codes of many wireless communication standards, has been integrated. To validate technology-scaling rules, all six architectures have been implemented in 180 nm, 130 nm, and 90 nm CMOS technology. Thus, we consider a total number of 18 max-log M-BCJR decoders in this work. As a proof-of-concept, we fabricated six max-log M-BCJR decoders in 180 nm CMOS technology and performed functional verification on a HP 83 000 F660 VLSI test system; the corresponding chip micrographs are depicted in Fig. 5.

We next detail the fixed-point parameters and summarize the key characteristics of the max-log M-BCJR decoder architectures. Finally, we compare our 8-state radix-2 and radix-4 decoder with existing 8-state SISO-MAP decoder ASICs to ensure that the trade-off analysis in Section V is performed on the basis of state-of-the-art SISO-MAP decoder designs.

A. Implementation parameters and error-rate performance

To achieve near-optimal SISO decoding performance, the input LLRs and output LLRs are quantized to 5 bit. The branch metrics are represented by 5 bit and the word-lengths W_{ACS} used within the ACS units (for proper functioning of modulo normalization) are provided in Table II. Note that radix-4 processing requires $W_{\text{ACS}}+1$ bit (compared with radix-2 designs), as the used branch metrics have twice the dynamic range [16]. The window length for all max-log M-BCJR decoder units was chosen $M = 32$, which provides close-to-optimal decoding performance for all implementations. Table II also shows the configuration details of the α-memories. Note that the α-memory of the radix-4 implementation is approximately half the size of the radix-2 design as only the branch metrics for the even-numbered trellis steps need to be stored. From Table III, we can observe that the α-memories for the radix-2 architectures require 14% to 30% of the total core area; for the radix-4 implementation, however, the α-memory occupies only 8% of the total core area.

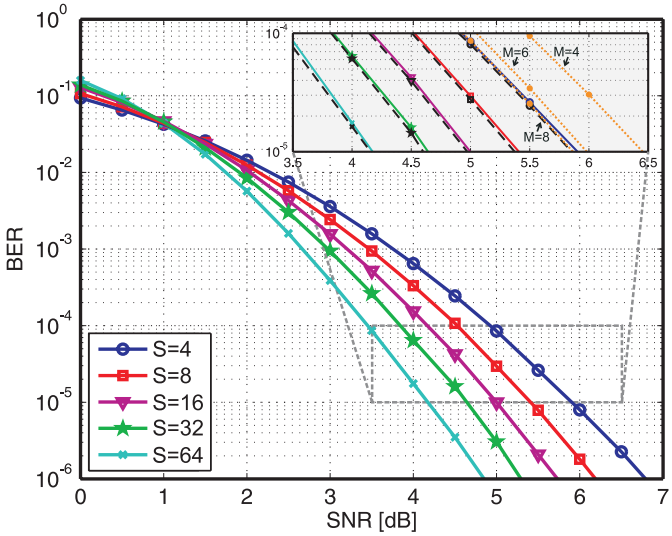
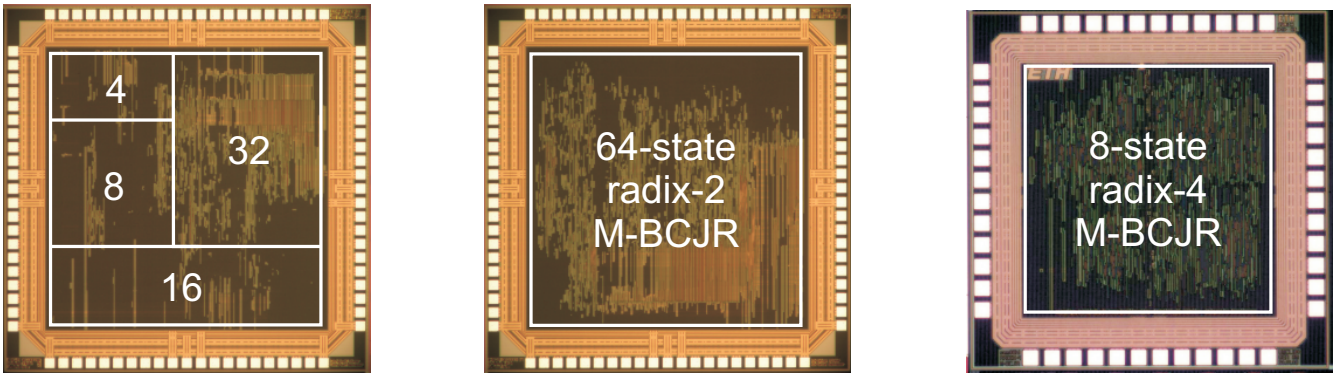


Fig. 6. Bit error-rate (BER) of the implemented max-log M-BCJR decoders. The dashed black curves in the zoom correspond to the performance of floating-point SISO-MAP decoding; the orange dotted curves show the performance of the 4-state floating-point max-log M-BCJR for $M \in \{4, 6, 8\}$.

Fig. 6 shows the bit error-rate (BER) performance of the implemented max-log M-BCJR decoders for various signal-to-noise ratios (SNRs). We simulated an additive white Gaussian noise (AWGN) channel using binary phase-shift keying (BPSK) modulation for a block-length of 1024 information bits; the BER is averaged over 100 000 Monte-Carlo trials. One can immediately observe that increasing the number of states S improves the BER performance (see Section V-A for more details). The zoom in Fig. 6 compares the BER performance to that of the reference SISO-MAP decoding algorithm. We see that the implementation loss is less than 0.13 dB compared to the optimal decoding algorithm, which can be addressed to finite-precision artifacts, the max-log approximation, and windowing. Note that for codes having less than 64-states, the window size M could be reduced (see, e.g., the BER of the 4-state floating-point max-log M-BCJR for $M \in \{4, 6, 8\}$ in Fig. 6). Nevertheless, we consider a constant window length for all designs to simplify the trade-off analysis in Section V.

B. Comparison with existing decoder implementations

Table III shows the (post-layout) results of all our max-log M-BCJR designs implemented in 180nm CMOS technology and provides a comparison with existing 8-state SISO-MAP decoder circuits for which corresponding implementation results are available [10], [11], [35]. Our radix-2 reference design is more than 6× better in terms of silicon complexity (in mm^{2}ns/bit) and is 2-to-3 times more energy efficient (in nJ/bit) compared to the 8-state log-MAP and max-log M-BCJR decoder in [10] and [11], respectively; this is a consequence of i) using the max-log approximation instead of log-MAP recursions (which only slightly affects the performance; see Fig. 6), ii) using modulo normalization instead of expensive renormalization circuitry (as used in [10]), iii) performing



(a) 4-state to 32-state radix-2 decoders (6.25 mm^2 die area, CLCC84 package).

(b) 64-state radix-2 decoder (6.25 mm^2 die area, CLCC84 package).

(c) 8-state radix-4 decoder (2.33 mm^2 die area, QFN48 package).

Fig. 5. ASIC micrographs of the fabricated max-log M-BCJR decoders in 180 nm (1P/6M) CMOS technology.

TABLE II
DESIGN PARAMETERS OF THE IMPLEMENTED MAX-LOG M-BCJR DECODER ARCHITECTURES

States/radix	4/2	8/2	16/2	32/2	64/2	8/4
Window length M	32	32	32	32	32	32
ACS word-length[a] [bit]	8	9	9	10	10	10
α–memory size [kbit]	1	2.25	4.5	10	20	1.25
S-RAM instances (words×bits)	1 (32×32)	1 (32×80)	2 (32×80)	3 (32×80) 3 (32×32)	8 (32×80)	1 (16×80)

[a]Corresponds to the number of bits required in the ACS units to enable modulo normalization [33].

extensive pipelining[5], and iv) the use of PMS in the LCU. Our radix-2 architecture achieves comparable silicon complexity and energy efficiency as the soft-output Viterbi algorithm (SOVA) implementation [35], which, however, delivers less accurate LLR-values than the BCJR algorithm (see the discussion in [35]). Hence, we can conclude that the trade-off analysis provided in the next section bases on state-of-the-art SISO-MAP decoder designs.

V. IMPLEMENTATION TRADE-OFFS

We now present the implementation trade-off analysis for max-log M-BCJR decoders and explore the validity of technology-scaling rules [36]. Our analysis is based on (post-layout) implementation results of the 18 max-log M-BCJR reference implementations detailed in Section IV.

A. Performance vs. silicon complexity

As mentioned in Section II-A1, increasing the number of states S leads to a larger minimum Hamming distance $d_{\min}$ (cf. Table I), which eventually improves the error-correction capabilities of CCs. This error-rate performance improvement comes, however, at the cost of an increase in terms of implementation complexity. This fundamental performance/complexity trade-off of SISO-MAP decoders is shown in Fig. 7. The performance is characterized by the *SNR-operating point*, which corresponds to the minimum signal-to-noise ratio (SNR) required to achieve a target bit error-rate of 10^{-5} in an AWGN channel using BPSK modulation and a block-length of 512 bit. The *silicon complexity* is characterized by the area-timing (AT-)product and is measured in mm^{2}ns/bit.

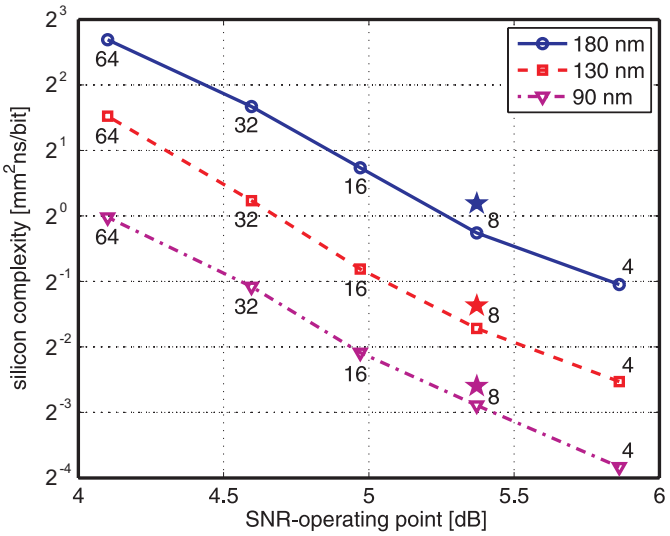


Fig. 7. Performance/complexity trade-off of the implemented SISO-MAP decoders in 180 nm, 130 nm, and 90 nm CMOS technology. The numbers correspond to states S and stars designate the results of the 8-state radix-4 decoder units.

[5]Note that pipelining can be beneficial from an energy-efficiency perspective as it i) reduces the critical path, while suffering only small power increase due to additional pipeline registers, and ii) effectively suppresses glitches.

TABLE III
180NM CMOS IMPLEMENTATION RESULTS AND COMPARISON TO OTHER 8-STATE SISO-MAP DECODERS

Publication	This work						[10]	[11]	[35]
Decoding algorithm	max-log M-BCJR						log-MAP M-BCJR	max-log M-BCJR	SOVA
States/radix	4/2	8/2	16/2	32/2	64/2	8/4	8/2	8/4×4	8/2
CMOS technology [nm]	180						180	130	180
Supply voltage [V]	1.8						1.8	1.32	1.8
Total cell area[a] [kGE]	22	38	67	130	241	67	150[b]	220	174
Total core area [mm^2]	0.26	0.45	0.79	1.54	3.0	0.78	1.5	1.96	0.5
α-memory area [mm^2]	0.037	0.077	0.155	0.464	0.618	0.063	–	–	–
Max. clock frequency [MHz]	539	542	476	484	465	344	285	238	–
Max. throughput [Mb/s]	539	542	476	484	465	687	285	952	500
Silicon complexity [mm^{2}ns/bit]	0.48	0.84	1.67	3.18	6.46	1.14	5.26	5.47[c]	1.00
Energy per bit [nJ/bit]	0.26	0.54	0.83	2.02	4.0	0.72	1.16	1.43[c]	0.80

[a]One gate equivalent corresponds to the size of a two-input drive-one NAND gate.
[b]Corresponding to the transistor count of the implementation.
[c]Technology scaling to 180 nm CMOS assuming: $A \sim 1/\ell^2$, $t_{\mathrm{pd}} \sim 1/\ell$, and $P_{\mathrm{dyn}} \sim 1/(V_\ell^2 \ell)$ [36].

From the double-logarithmic plot in Fig. 7 we observe that for a given technology the silicon complexity increases roughly linearly in S or, equivalently, *exponentially* in the constraint length (see Section V-B2 for details on the scaling behavior). Hence, improving the performance of CCs (i.e., reducing the SNR-operating point) quickly results in prohibitive silicon complexity, especially for BCJR decoders supporting more than $S = 64$ states. Therefore, alternative coding schemes, such as turbo codes (which typically rely on 8-state CCs), become necessary for communication systems targeting SNR-operating points close to the Shannon limit. Fig. 7 additionally shows that radix-4 implementations require 22% to 36% higher silicon complexity for a given performance and therefore, can be considered as less hardware efficient (in terms of area per throughput) than their radix-2 counterparts.

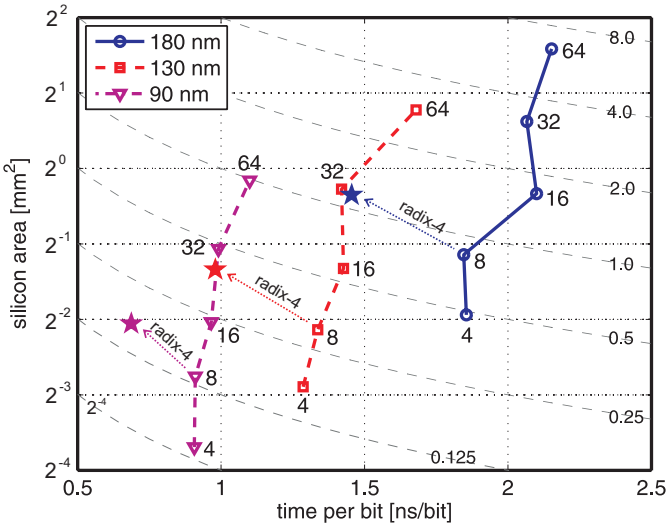


Fig. 8. Throughput/silicon-area trade-off. The number next to the curves correspond to the number of states, stars designate 8-state radix-4 architectures, and the thin dashed lines correspond to constant silicon complexity (in terms of mm^{2}ns/bit).

B. Throughput vs. silicon area

Fig. 8 shows the trade-off between throughput (in ns/bit) and silicon area (in logarithmic scale) depending on the number of states S supported by the decoder and the CMOS technology node.

1) Silicon area: We observe that the silicon area A scales linearly in S (for a given technology), which is an immediate consequence of the fact that decoder area is dominated by the state-metric recursion units and the α-memory, whose area scales linearly in S. In order to corroborate this observation, least-squares fitting to all radix-2 implementation results is performed. To this end, we consider common technology scaling rules [36] and propose the following first-order approximation for the silicon area:

$$A \approx (\ell/180)^2 \big(\alpha_A S + \beta_A\big) \quad [\mathrm{mm}^2] \tag{3}$$

where ℓ denotes the minimum feature size (in nm), and $\alpha_A = 0.0502$ and $\beta_A = 0.0712$ are the least-squares fitting parameters. The model (3) results in a relative error that is smaller than 25% for all considered implementations and hence, confirms the validity of the first order model and supports the fact that technology scaling behaves—among the considered technology nodes—as the predictions in [36]. Note that the linear scaling of the silicon area in S shown in (3) can also be observed in ASIC micrographs shown in Figs. 5(a) and 5(b), i.e., the 64-state decoder occupies the same die area as the 4, 8, 16, and 32 state decoders together.

2) Throughput: For the throughput, Fig. 8 shows only a weak dependence on S. The underlying reason is the fact that the critical path is essentially[6] determined by the number of bits W_{ACS} required in the ACS units, which scales only *logarithmically* in S for modulo normalization (see [33] for

[6]Note that for the 180nm designs, the critical path of the 16-state decoder is slightly longer than that of the 32-state design, which is an artifact of our 16-state decoder layout.

details). As a consequence, we propose the following first-order model for the critical path T of our radix-2 designs:

$$T \approx (\ell/180)\big(\alpha_T \log_2(S) + \beta_T\big) \quad \text{[ns]} \qquad (4)$$

where $\alpha_T = 0.0985$ and $\beta_T = 1.5839$ are obtained through least-squares fitting. For these parameters, the model (4) exhibits a maximum relative error of 6% which confirms i) the validity of the model (4) and ii) that common technology-scaling rules [36] apply for the critical path of BCJR decoders in 180 nm, 130 nm, and 90 nm CMOS technology. Furthermore, combining the models in (3) with (4) shows that the silicon complexity defined as AT scales with $S\log_2(S)$, which was observed before in Fig. 7.

Both models (3) and (4) confirm technology-scaling rules for area and throughput of radix-2 max-log M-BCJR implementations according to [36]. Hence, the silicon area and throughput of BCJR designs in most advanced CMOS technologies (with feature size of 65 nm and below) is expected to fully benefit from technology scaling, which implies that data-rates in the Gb/s regime (as targeted by next-generation wireless standards, such as LTE-Advanced [7]) can be achieved by migrating our reference architecture to more recent technology nodes (see Section VI-B).

3) Radix-4 implementations: The implementation results for the 8-state radix-4 decoders in Fig. 8 support the observation that radix-4 implementations achieve roughly 27% higher throughput than the corresponding radix-2 designs[7], but are less efficient (by 22% to 36%) from a silicon-complexity perspective. We emphasize, however, that radix-4-based designs have the potential of lowering the clock frequency for a given target throughput. Such a behavior can be beneficial—at least from a practical (implementation) viewpoint. Specifically, radix-4 can be used to avoid excessively high clock frequencies for more advanced technology nodes (which starts to happen when integrating the presented architectures in 65 nm CMOS) and hence, helps to alleviate the need for sophisticated clock generation and distribution schemes.

C. Energy efficiency vs. silicon area

Fig. 9 shows the trade-off between energy efficiency (in terms of nJ/bit) and silicon area in a double-logarithmic plot; the energy-efficiency figures include dynamic and static power consumption.[8] It is interesting to see that there is a (near-perfect) linear dependence between silicon area and energy efficiency, i.e., increasing the number of states S (e.g., to lower the SNR-operating point) leads to a linearly proportional degradation in terms of energy efficiency. This behavior renders implementations supporting a large number of states (e.g., $S > 64$) to be energy inefficient (analogously to the silicon-complexity behavior in Section V-A).

[7]The throughput gain of radix-4 architectures (i.e., decoding 2 bits/cycle instead of 1 bit/cycle), is reduced by the substantially longer critical path compared to radix-2 ACS circuits.

[8]Note that dynamic power consumption is dominating for all considered technology nodes.

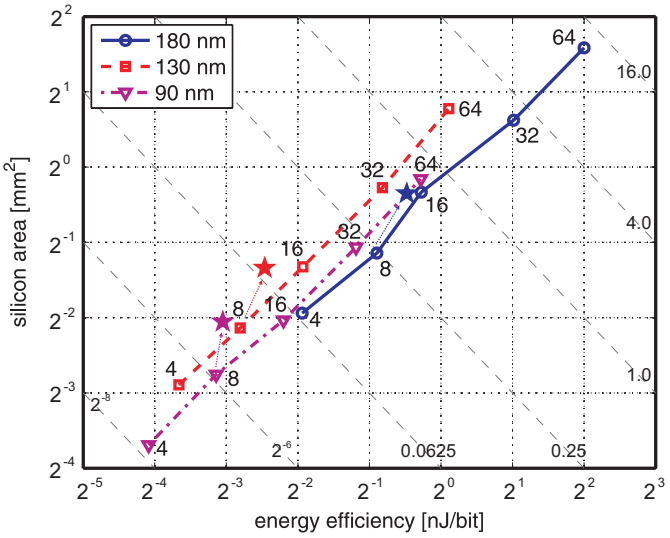


Fig. 9. Energy efficiency/silicon-area trade-off. The numbers next to the curves correspond to the number of states, stars designate 8-state radix-4 architectures, and the thin dashed lines correspond to a constant AE-product.

1) First-order model: We next validate technology scaling rules for the energy efficiency E based on the radix-2 sample points shown in Fig. 9. To this end, we assume the energy being proportional to the square of the supply voltage V_ℓ of the process technology ℓ, and proportional to the gate capacitances according to [36]. These assumptions and the observation that E scales linearly proportional with the silicon area A lead to the following first-order model:

$$E \approx (V_\ell/1.8)^2(\ell/180)\big(\alpha_E S + \beta_E\big) \quad \text{[nJ/bit]} \qquad (5)$$

with the least-squares fitting parameters $\alpha_E = 0.066$ and $\beta_E = 0.03$, and the supply voltages $V_\ell \in \{1.0, 1.2, 1.8\}$ associated with the considered technology nodes $\ell \in \{90, 130, 180\}$ (in nm). The relative error associated with the model (5) is within 32% for the considered technology nodes and is, therefore, slightly less accurate than the model (3) for the silicon area A.

2) Supply-voltage scaling: We additionally see from Fig. 9 that the improvement in terms of energy efficiency from 180 nm to 130 nm is more pronounced than it is the case for 130 nm to 90 nm. The underlying reason is the fact that the supply-voltage reduction from 130 nm to 90 nm is only 0.2 V (compared with a 0.6 V reduction from 180 nm to 130 nm), which—in accordance to the model (5)—yields a diminishing reduction in terms of nJ/bit.

The migration to more recent technology nodes (e.g., 65 nm or 45 nm) is expected to provide even less improvement in terms of energy efficiency, as scaling of the supply voltage below 1 V becomes challenging (see, e.g., [37], [38]). In order to arrive at energy-efficient designs in such technologies, voltage-frequency scaling [2], [39] provides a potential solution. This technique amounts to designing the decoder circuit for a higher throughput than necessary and using part of the margin to reduce the clock frequency together with the supply voltage for further reduction of the power consumption (see, e.g., [12] for an application of voltage-frequency scaling in a low-power

turbo-decoder implementation).

3) Radix-4 implementations: For the radix-4 implementation we can observe a degradation of the energy efficiency (compared with radix-2 designs) by 34%, 27% and 7% for 180 nm, 130 nm and 90 nm CMOS technology, respectively. Therefore, we conclude that radix-4 max-log M-BCJR decoders are, in general, less energy efficient than corresponding radix-2 designs. This behavior is mainly a result of the fact that radix-4 requires more logic per throughput (compared with radix-2 designs; cf. Section V-B3), which causes higher switching activity, eventually resulting in higher (static and dynamic) power consumption for a given throughput.

VI. Case Study: Parallel Turbo Decoder for LTE

Modern wireless communication standards, such as 3GPP-LTE [6], specify the use of turbo codes, in addition to CCs. In order to achieve the LTE peak throughput of 326.4 Mb/s [40], multiple SISO-MAP decoder units which process the trellis in a parallel fashion, are necessary. This approach is known as *parallel turbo decoding* and was shown to increase the throughput roughly linearly in the number of SISO-MAP decoder instances (see, e.g., [16], [41], [42]). In order to demonstrate the usefulness of the trade-off analysis shown in Section V, we next derive the key design parameters (i.e., the number of parallel BCJR instances and the ACS radix) for a parallel turbo decoder in 90 nm technology that achieves the LTE peak throughput with margin. We furthermore provide a brief outlook on turbo-decoder circuits for next-generation wireless standards in more advanced CMOS technology nodes.

A. Design-space exploration

We start with the 3.57 mm^2 8× parallel turbo-decoder ASIC implementation in [16], which employs eight radix-4 M-BCJR instances to achieve 390.6 Mb/s in 130 nm CMOS. In a first step, we migrate from 130 nm to 90 nm and assume technology scaling for silicon area and throughput as detailed in Section V-B; this results in an estimated 564.2 Mb/s throughput at 1.7 mm^2. Since 564.2 Mb/s exceeds the LTE peak throughput of 326.4 Mb/s [40], we are able to employ radix-2 processing instead of radix-4, which lowers the throughput by 24% but improves the overall hardware and energy efficiency. This architectural transformation is expected to result in approximately 430 Mb/s. The resulting (estimated) silicon area of the 8× radix-2-based parallel turbo decoder is only 1.3 mm^2, since 90 nm radix-2 designs exhibit roughly 38% smaller area than radix-4 designs and 2/3 of the turbo-decoder area is occupied by the eight radix-4 max-log M-BCJR instances [16].[9] This case study shows that migrating from 130 nm to 90 nm and performing radix-2 instead of radix-4 is expected to reduce the silicon complexity by 3×. We also expect the corresponding energy efficiency to improve by no less than a factor of two as our radix-2 designs have shown to be significantly more energy efficient than radix-4 max-log M-BCJR decoders (see Section V-C3 for the details).

[9]For the area estimates, we ignore the impact of radix-2 processing to the complexity of the interleaver circuit. This simplification is expected to result in *pessimistic* estimates, since interleavers for radix-2 designs require substantially lower complexity than interleavers for radix-4-based turbo decoders.

B. Predictions for next-generation wireless standards

Our trade-off analysis also allows us to study turbo-decoder implementations for next-generation cellular communication systems targeting 1 Gb/s, such as LTE-Advanced [7]. As detailed before, the M-BCJR throughput and, hence, the throughput of corresponding turbo-decoder implementations scale linearly with the feature size. Therefore, the re-integration of turbo-decoder designs in more recent technology nodes (e.g., 65 nm, 45 nm, or 32 nm) will benefit from this technology-driven performance gain. Specifically, migrating the 90 nm turbo-decoder design evaluated in Section VI-A to 32 nm is expected to exceed the 1 Gb/s peak throughput targeted by LTE-Advanced. However, the power consumption of such high-throughput turbo-decoder implementations will render the use in battery-powered mobile devices extremely challenging. This fact is further aggravated by the observation that the technology-driven energy-efficiency improvements will slow down significantly with more advanced technology nodes [38]. Assuming that beyond 90 nm the energy efficiency scales only *linearly* in the feature size (for reasons discussed in Section V-C2), a 1 Gb/s turbo-decoder architecture in 32 nm would consume roughly 300 mW to 400 mW, which is substantially higher than the typical 100 mW power budget of decoding circuits for mobile devices [16]. We, therefore, conclude that even more advanced technologies in combination with voltage-frequency scaling are required to arrive at 1 Gb/s turbo-decoder solutions that meet the desired power budget of battery-powered mobile devices.

VII. Conclusions

In this paper, we analyzed the design and implementation trade-offs for SISO-MAP decoding based on the max-log M-BCJR algorithm. To this end, we presented reference decoder designs for 4-state to 64-state convolutional codes (CCs) in 90 nm, 130 nm, and 180 nm CMOS technology, and extracted the underlying area, throughput, and energy efficiency trade-offs. Our decoder implementations show superior hardware and energy efficiency compared with existing BCJR designs and the provided implementation trade-off analysis turns out to be a useful tool for identifying the key design parameters for parallel turbo decoders used in today's and next-generation wireless communication systems.

References

[1] C. Studer, "Iterative MIMO decoding: Algorithms and VLSI implementation aspects," Ph.D. dissertation, ETH Zürich, Switzerland, Series in Microelectronics, vol. 202, Hartung-Gorre Verlag Konstanz, 2009.

[2] C. Benkeser, "Power efficiency and the mapping of communication algorithms into VLSI," Ph.D. dissertation, ETH Zürich, Switzerland, Series in Microelectronics, vol. 209, Hartung-Gorre Verlag Konstanz, 2010.

[3] P. Elias, "Coding for noisy channels," in *IRE Convention Record*. Part IV, 1955, pp. 37–46.

[4] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261–1271, Oct. 1996.

[5] *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Channel Coding and Multiplexing Examples*, 3GPP Organizational Partners TS 25.944, Rev. 4.1.0, Jun. 2001.

[6] *3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and channel coding (Release 9)*, 3GPP Organizational Partners TS 36.212, Rev. 8.3.0, May 2008.

[7] *3rd Generation Partnership Project; LTE; Feasibility study for Further Advancements for E-UTRA (LTE-Advanced)*, 3GPP Organizational Partners TR 36.912, Rev. 10.0.0, Apr. 2011.

[8] *IEEE Draft Standard; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications; Amendment 4: Enhancements for Higher Throughput*, P802.11n/D3.0, Sep. 2007.

[9] M. Mansour and N. Shanbhag, "VLSI architectures for SISO-APP decoders," *IEEE Trans. VLSI*, vol. 11, no. 4, pp. 627–650, Aug. 2003.

[10] S.-J. Lee, N. R. Shanbhag, and A. C. Singer, "A 285-MHz pipelined MAP decoder in 0.18μm CMOS," *IEEE J. Solid-State Circuits*, vol. 40, no. 8, pp. 1718–1725, Aug. 2005.

[11] C.-H. Tang, C.-C. Wong, C.-L. Chen, C.-C. Lin, and H.-C. Chang, "A 952Ms/s max-log MAP decoder chip using radix-4×4 ACS architecture," in *IEEE A-SSCC*, Hangzhou, China, Nov. 2006, pp. 79–82.

[12] C. Benkeser, A. Burg, T. Cupaiuolo, and Q. Huang, "Design and optimization of an HSDPA turbo decoder ASIC," *IEEE J. Solid-State Circuits*, vol. 44, no. 1, pp. 98–106, Jan. 2008.

[13] C.-H. Lin, C.-Y. Chen, T.-H. Tsai, and A.-Y. Wu, "Low-power memory-reduced traceback MAP decoding for double-binary convolutional turbo decoder," *IEEE Trans. Circ. Systems I*, vol. 56, no. 5, pp. 1005–1016, May 2009.

[14] F. Naessens, V. Derudder, H. Cappelle, L. Hollevoet, P. Raghavan, M. Desmet, A. AbdelHamid, I. Vos, L. Folens, S. O'Loughlin, S. Singirikonda, S. Dupont, J.-W. Weijers, A. Dejonghe, and L. Van der Perre, "A 10.37 mm^2 675 mW reconfigurable LDPC and Turbo encoder and decoder for 802.11n, 802.16e and 3GPP-LTE," in *Dig. Techn. Papers, Symp. on VLSI Circuits*, Honolulu, HI, USA, June 2010, pp. 213 –214.

[15] C.-C. Wong, M.-W. Lai, C.-C. Lin, H.-C. Chang, and C.-Y. Lee, "Turbo decoder using contention-free interleaver and parallel architecture," *IEEE J. Solid-State Circuits*, vol. 45, no. 2, pp. 422 –432, Feb. 2010.

[16] C. Studer, C. Benkeser, S. Belfanti, and Q. Huang, "Design and implementation of a parallel turbo-decoder ASIC for 3GPP-LTE," *IEEE J. of Solid-State Circuits*, vol. 46, no. 1, pp. 8–7, Jan. 2011.

[17] B. M. Hochwald and S. ten Brink, "Achieving near-capacity on a multiple-antenna channel," *IEEE Trans. Commun.*, vol. 51, no. 3, pp. 389–399, Mar. 2003.

[18] M. Tüchler, A. C. Singer, and R. Koetter, "Minimum mean squared error equalization using a priori information," *IEEE Trans. Sig. Proc.*, vol. 50, no. 3, pp. 673–983, Mar. 2002.

[19] C. Studer, S. Fateh, and D. Seethaler, "ASIC implementation of soft-input soft-output MIMO detection using parallel interference cancellation," *IEEE J. Solid-State Circuits*, vol. 56, no. 7, pp. 1754–1765, July 2011.

[20] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260–269, Apr. 1967.

[21] P. J. Black and T. H. Meng, "A 140-Mb/s, 32-state, radix-4 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 27, no. 12, pp. 1877–1885, Dec. 1992.

[22] Y.-N. Chang, H. Suzuki, and K. K. Parhi, "A 2-Mb/s 256-state 10-mW rate-1/3 Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 35, no. 6, pp. 826–834, June 2000.

[23] T. Gemmeke, M. Gansen, and T. G. Noll, "Implementation of scalable power and area efficient high-throughput Viterbi decoders," *IEEE J. Solid-State Circuits*, vol. 37, no. 7, pp. 941–948, July 2002.

[24] A. Burg, S. Haene, M. Borgmann, D. Baum, T. Thaler, F. Carbognani, S. Zwicky, L. Barbero, C. Senning, P. Greisen, T. Peter, C. Foelmli, U. Schuster, and P. Tejera, "A 4-stream 802.11n baseband transceiver in 0.13 µm CMOS," in *Dig. Techn. Papers, Symp. on VLSI Circuits*, Kyoto, Japan, Jun. 2009, pp. 282–283.

[25] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284–287, Mar. 1974.

[26] B. Bougard, A. Giulietti, L. Van der Perre, and F. Catthoor, "A class of power efficient VLSI architectures for high speed turbo-decoding," in *Proc. of GLOBECOM*, vol. 1, Taipei, Taiwan, Nov. 2002, pp. 549–553.

[27] J. G. Proakis, *Digital Communications*, 4th ed. New York, USA: McGraw–Hill, 2001.

[28] J. Hagenauer, E. Offer, and L. Papke, "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inf. Theory*, vol. 42, no. 2, pp. 429–445, Mar. 1996.

[29] J. P. Woodard and L. Hanzo, "Comparative study of turbo decoding techniques: an overview," *IEEE Trans. Vehicular Tech.*, vol. 49, no. 6, pp. 2208–2233, Nov. 2000.

[30] V. Franz and J. B. Anderson, "Concatenated decoding with a reduced-search BCJR algorithm," *IEEE J. on Sel. Areas in Comm.*, vol. 16, no. 2, pp. 186–195, Feb. 1998.

[31] P. Meinerzhagen, C. Roth, and A. Burg, "Towards generic low-power area-efficient standard cell based memory architectures," in *Proc. of IEEE MWSCAS*, Seattle, WA, USA, Aug. 2010, pp. 129–132.

[32] M. Martina and G. Masera, "State metric compression techniques for turbo decoder architectures," *IEEE Trans. Circ. Systems I*, vol. 58, no. 5, pp. 1119–1128, May 2011.

[33] C. B. Shung, P. H. Siegel, G. Ungerboeck, and H. K. Thapar, "VLSI architectures for metric normalization in the Viterbi algorithm," in *Proc. of IEEE ICC*, vol. 4, Atlanta, GA, USA, Apr. 1990, pp. 1723–1728.

[34] G. Fettweiss and H. Meyr, "Parallel Viterbi algorithm implementation: breaking the ACS-bottleneck," *IEEE Trans. Commun.*, vol. 37, no. 8, pp. 785–790, Aug. 1989.

[35] E. Yeo, S. Augsburger, W. Davis, and B. Nikolić, "A 500-Mb/s soft-output Viterbi decoder," *IEEE J. Solid-State Circuits*, vol. 38, no. 7, pp. 1234 – 1241, July 2003.

[36] B. Razavi, *Design of analog CMOS integrated circuits*. New York, NY: McGraw-Hill, 2002.

[37] M. Horowitz, E. Alon, D. Patil, S. Naffziger, R. Kumar, and K. Bernstein, "Scaling, power, and the future of CMOS," in *IEDM Techn. Digest. IEEE Int. Electron. Devices Meeting*, Dec. 2005, pp. 7–15.

[38] J. Rabaey, "Ultra low power/voltage design," in *Low Power Design Essentials*, ser. Integrated Circuits and Systems. Springer US, 2009, pp. 289–316.

[39] A. Chandrakasan and R. W. Brodersen, *Low Power Digital CMOS Design*. New York, USA: IEEE, 1998.

[40] 3GPP TR25.912 V8.0.0, "Feasibility study for evolved universal terrestrial radio access (UTRA) and universal terrestrial radio access network (UTRAN)," Feb. 2009.

[41] C.-C. Wong and H.-C. Chang, "Reconfigurable turbo decoder with parallel architecture for 3GPP LTE system," *IEEE Trans. Circ. Systems II*, vol. 57, no. 7, pp. 566–570, July 2010.

[42] ——, "High-efficiency processing schedule for parallel turbo decoders using QPP interleaver," *IEEE Trans. Circ. Systems I*, vol. 58, no. 6, pp. 1412–1420, June 2011.