Hindawi Publishing Corporation
VLSI Design
Volume 2012, Article ID 942893, 10 pages
doi:10.1155/2012/942893

Review Article

Design Space of Flexible Multigigabit LDPC Decoders

Philipp Schläfer,[1] Christian Weis,[1] Norbert Wehn,[1] and Matthias Alles[2]

[1] *Microelectronic Systems Design Research Group, University of Kaiserslautern, 67663 Kaiserslautern, Germany*
[2] *R&D Department, Creonic GmbH, 67655 Kaiserslautern, Germany*

Correspondence should be addressed to Philipp Schläfer, schlaefer@eit.uni-kl.de

Received 2 December 2011; Accepted 7 February 2012

Academic Editor: Guido Masera

Multigigabit LDPC decoders are demanded by standards like IEEE 802.15.3c and IEEE 802.11ad. To achieve the high throughput while supporting the needed flexibility, sophisticated architectures are mandatory. This paper comprehensively presents the design space for flexible multigigabit LDPC applications for the first time. The influence of various design parameters on the hardware is investigated in depth. Two new decoder architectures in a 65 nm CMOS technology are presented to further explore the design space. In the past, the memory domination was the bottleneck for throughputs of up to 1 Gbit/s. Our systematic investigation of column- versus row-based partially parallel decoders shows that this is no more a bottleneck for multigigabit architectures. The evolutionary progress in flexible multigigabit LDPC decoder design is highlighted in an extensive comparison of state-of-the-art decoders.

1. Introduction

Over the past years, a trend to ever increasing data rates could be observed which is enforced mostly by wireless applications. New standards like WiMedia 1.5 [1] for ultra wideband (UWB), IEEE 802.15.3c [2] for wireless personal area networks (WPANs), and IEEE 802.11ad [3] for Gigabit-WLAN (WiGig) define data rates of several Gbit/s. These standards allow for wireless broadband internet, wireless HDMI, and other advanced technologies. To approach the channel capacity while serving these high data rates, Low-density parity-check (LDPC) codes are excellent candidates, see Section 2. Even though LDPC codes belong to the best channel coding schemes known today, their implementation poses big challenges. This is due to the iterative, computational expensive decoding procedure, requiring massive parallel architectures. Therefore, design decisions concerning the decoder architecture have to be carefully evaluated.

Many papers have been published in the past using different approaches of (partially) parallel LDPC decoder architectures. The decoder design space including hardware mapping, node parallelism, as well as other parameters is discussed in detail in Section 3. Each of these parameters has a strong impact on the resulting hardware efficiency, the achievable throughput, and the communications performance.

This is the first paper comprehensively discussing the design space in particular for flexible multigigabit LDPC decoders. Due to the high degree of parallelism, there are new associations which have not been observed earlier. This paper discusses the multigigabit LDPC decoder design space and its influence on the hardware implementation. The benefits and drawbacks of single design decisions are pointed out. The impact a decision has on the hardware is directly shown by the comparison of decoders. This knowledge can be used to determine design decisions at the outset, based on the aimed target properties.

Section 5 shows the progress in multigigabit decoder architectures. Two new decoders are published for the first time in this paper. By these designs, the flexible multigigabit design space is further explored and important information about decoder architectures is gained.

2. LDPC Codes

LDPC codes [4] are linear block codes defined by a sparse parity check matrix H of dimension $M \times N$, see Figure 1(a).

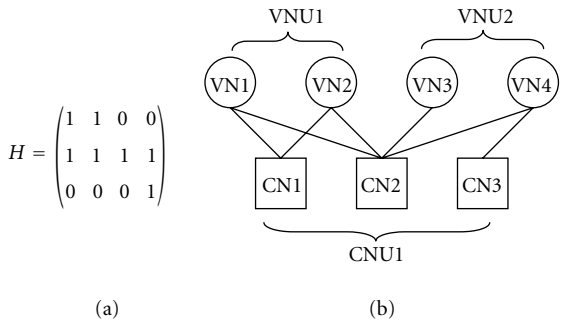

Figure 1: H matrix Tanner graph hardware mapping.

A valid code word $\vec{x}$ has to satisfy $H\vec{x}^T = \vec{0}$ in modulo-2 arithmetic. A descriptive graphical representation of the whole code is given by a Tanner graph. Each row of the parity check matrix is represented by a check node (CN) and corresponds to one of the M parity checks. Respectively each column corresponds to a variable node (VN) corresponding to one of the N code bits. The Tanner graph shown in Figure 1(b) is the alternative representation for the parity check matrix of Figure 1(a). Edges in the Tanner graph reflect the "1"s in the H matrix. There is an edge between VN n and CN m if and only if $H_{mn} = 1$. Figure 1(b) shows the mapping of the Tanner graph to hardware processing units. In the given example, each two VNs are mapped to one variable node unit (VNU). For the CNs, only one hardware instance, a check node unit (CNU), is instantiated to process all the CNs of the parity check matrix. This of course implies a time multiplexed operation of the nodes and spans a wide range of possible implementations. The different approaches are highlighted in Section 3 as they have great influence on the resulting architecture. However, random-structured Tanner graphs pose big problems for hardware implementations. The interconnection network between the different kind of nodes has to process the random message exchange, which results in inefficient architectures. For the construction of structured LDPC codes, shifted identity matrices I^x of size $P \times P$ and the zero matrix are used as submatrices in H, (1).

$$I^1 = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{1}$$

Equation (2) shows the construction principle of the H matrix. This matrix structure in the first place allows for the exploitation of the algorithm's inherent parallelism by parallel instantiation of the functional units.

$$H = \begin{pmatrix} H_{1,1} & \dots & H_{1,N/P} \\ H_{2,1} & \dots & H_{2,N/P} \\ \vdots & \ddots & \vdots \\ H_{M/P,1} & \dots & H_{M/P,N/P} \end{pmatrix} H_{mp,np} = \begin{cases} 0, \\ I^x. \end{cases} \tag{2}$$

LDPC codes can be decoded by the Sum-Product Algorithm (SPA). Probabilistic messages are iteratively exchanged between variable and check nodes.

First, the variable nodes are initialized with the channel values λ_n^{ch}. According to the connections in the parity check matrix, these values are passed from each variable node n with $n \in \{0, \dots, N-1\}$ to the connected check nodes $\mathcal{M}(n)$ with

$$\mathcal{M}(n) = \{m \mid m \in \{0, \dots, M-1\} \wedge H_{mn} \neq 0\}. \tag{3}$$

The set of all variable nodes connected to check node m with $m \in \{0, \dots, M-1\}$ is defined as

$$\mathcal{N}(m) = \{n \mid n \in \{0, \dots, N-1\} \wedge H_{mn} \neq 0\}. \tag{4}$$

The check node computation can be split in two parts, the actual parity check, represented by the sign $\operatorname{sgn}(\epsilon)$, and the result's probability represented by the magnitude $|\epsilon|$. These extrinsic messages are computed as follows:

$$\operatorname{sgn}\left(\epsilon_{m\to n}^{(i)}\right) = \prod_{n' \in \mathcal{N}(m)\setminus n} \operatorname{sgn}\left(z_{n'\to m}^{(i)}\right),$$

$$\left|\epsilon_{m\to n}^{(i)}\right| = \Phi^{-1}\left(\sum_{n' \in \mathcal{N}(m)\setminus n} \Phi\left(\left|z_{n'\to m}^{(i)}\right|\right)\right), \text{ with} \tag{5}$$

$$\Phi(x) = \Phi^{-1}(x) = -\ln\left(\tanh\left(\frac{x}{2}\right)\right).$$

Finally, the messages are returned to the variable nodes which generate new intrinsic messages $z^{(i)}$ for iteration i:

$$z_{n\to m}^{(i)} = \lambda_n^{ch} + \sum_{m' \in \mathcal{N}(n)\setminus m} \epsilon_{m'\to n}^{(i-1)} = \Lambda_n^{(i-1)} - \epsilon_{m\to n}^{(i-1)}, \tag{6}$$

with

$$\Lambda_n^{(i)} = \lambda_n^{ch} + \sum_{m' \in \mathcal{M}(n)} \epsilon_{m'\to n}^{(i)}. \tag{7}$$

The sign of the a posteriori probability (APP) Λ can be interpreted as the hard decision bit. Message exchange between variable and check node continues until either a valid code word is found or a maximum number of iterations is exceeded.

3. Decoder Design Space

Figure 2 gives an overview of the multigigabit LDPC decoder design space. Appropriate decisions have to be met in order to derive an area and energy efficient decoder while serving the demanded flexibility. The final decisions heavily depend on constraints like communications performance, throughput, area, or energy consumption, which are dictated by the applications and supported standards. The following sections present the complex interdependencies which have to be considered during the design of hardware decoders. The focus in this overview is on design decisions of multigigabit decoder architectures.

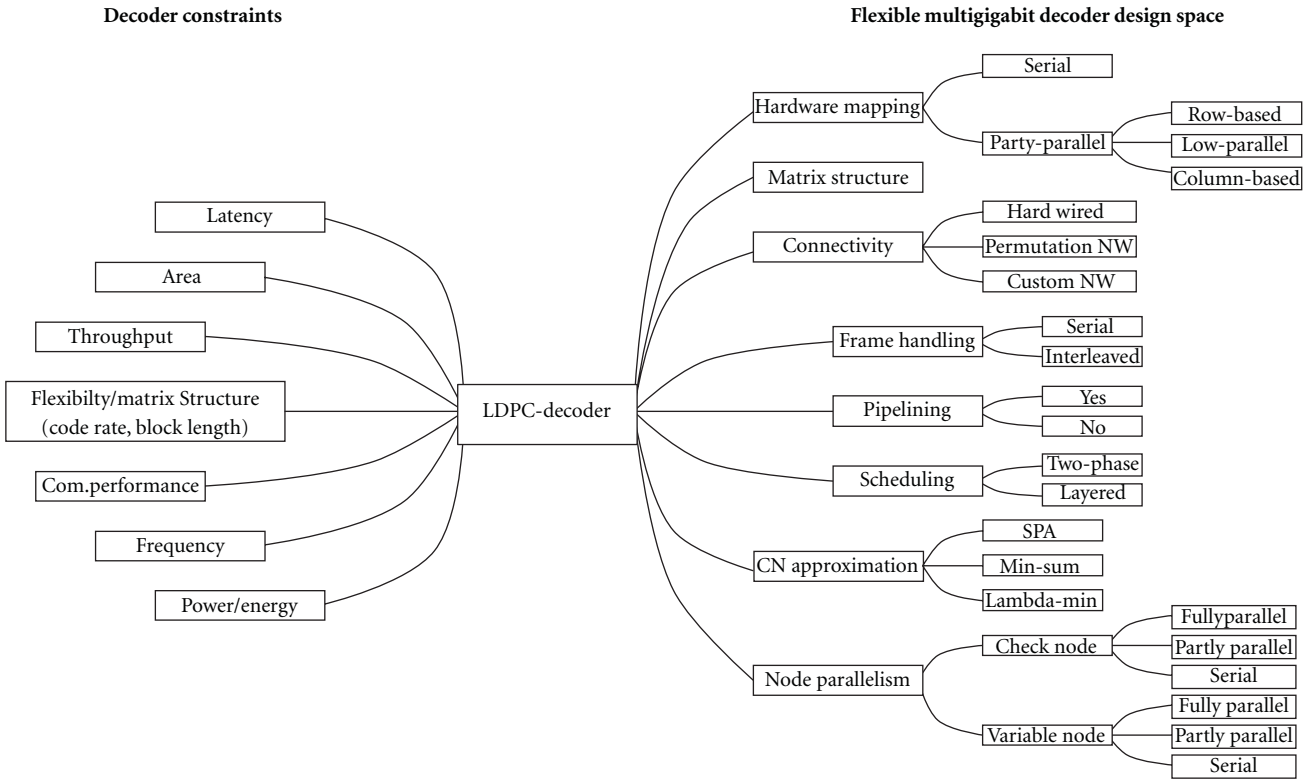

Figure 2: Flexible multigigabit LDPC decoder design space and decoder constraints.

3.1. Scheduling. A partially parallel decoder can use different node update schedules, two-phase, or layered scheduling. The two-phase schedule processes all check nodes before the variable nodes produce new messages. In the layered schedule, the check nodes are processed in a way that the variable nodes are intermediately updated [5–7]. Thus, the remaining CNs can take results into account which have been generated in the same iteration. The layered schedule can achieve the same communications performance as the two-phase schedule while reducing the needed iterations by about 50%. This allows for much more energy efficient decoders, reducing the frequency by 50% with the same throughput. Another possibility is to reduce the degree of parallelism and thus the decoder's area demand. Therefore, layered scheduling should be applied whenever possible.

3.2. Node Parallelism. The node degree defines how many edges a single node can read or write per clock cycle. This parameter can be configured independently for the check and variable nodes. A straight forward serial implementation reads one edge and produces one output value per cycle. This node architecture results in large flexibility as each node degree can easily be supported. However, serial node implementations are prohibitive for multigigabit decoders because of the moderate achievable throughput. For higher target throughputs, either the VNU, the CNU, or both node parallelisms must be increased. To get an efficient hardware utilization, the node degree must match the number of computational units generating data for the respective node. In this case, all produced data can be processed without introducing wait states. The required node parallelism thus needs to be derived from the chosen hardware mapping. Finally, the node degree is also a factor with strong impact on the functional unit size. The more edges a node can process in parallel the bigger it gets.

3.3. Hardware Mapping. The hardware mapping is one of the most important parameters for the decoder design because it has great impact on the resulting hardware. Three kinds of implementation styles for the Tanner graph exist: Fully parallel, partly parallel, and serial processing. For serial processing, only one instance of each functional unit, one VNU, and one CNU are implemented, gaining high flexibility but only moderate throughput. A fully parallel hardware mapping implementing all nodes of the Tanner graph is also possible, gaining high throughput. However, this hardware mapping lacks flexibility and is limited to small block sizes. Routing congestion is another obstacle when applying this approach. In [8] it is reported, that only 50% of the chip is used by logic for a fully parallel mapping. A possibility to relax the routing congestion problem is the use of bit-serial networks which reduce the node interconnections. As shown in [9] very efficient fully parallel decoder architectures are achievable by this approach. However fully parallel decoders lack the flexibility which is mandatory for the most current standards. Therefore, we will focus only on partly parallel hardware mappings in this paper. This still includes a wide span of decoder architectures which can be distinguished

Table 1: Multigigabit decoder hardware mapping.

Hardware mapping	CNU instances	VNU instances
Low-parallel	$i \times P$	$j \times P$
Row-based	$i \times P$	N
Column-based	M	$i \times P$

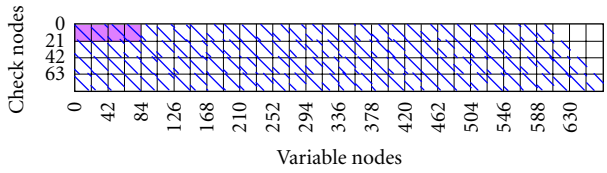

Figure 3: State-of-the-art decoder processing P CNs and $4 \times P$ VNs per cycle.

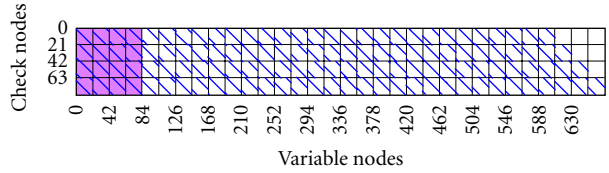

Figure 4: Column-based decoder processing M CNs and $4 \times P$ VNs per cycle.

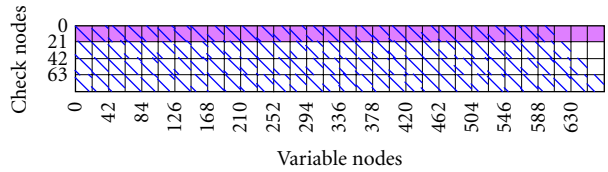

Figure 5: Row-based decoder processing P CNs and N VNs per cycle.

by the ratio of VNUs to CNUs. Basically, there are three approaches, a low-parallel, a column-based, and a row-based decoder. Table 1 points out the respective VNU to CNU ratios.

The low-parallel approach is useful for moderate throughput demands of up to 1 Gbit/s. It instantiates a small number of VNUs and CNUs and thus needs a huge number of cycles per iteration. Figure 3 shows the processing of four submatrices per cycle. The highlighted area represents the part of the parity check matrix which is processed in the first cycle. Overall this architecture needs 32 cycles per iteration. This hardware mapping requires partly parallel CNUs, processing j edges per cycle. The VNUs work serially, consuming and generating one edge per cycle. Overall only few functional units need to be instantiated. However, for low-parallel decoder architectures, we observe memory dominations of up to 70%. Our investigations have shown that the dominant memory is the extrinsic message RAM storing the messages which have to be passed from CNs to VNs. The poor ratio of logic to memory area is a result of the low parallelism and cannot be improved for this kind of hardware mapping.

The column-wise decoding making a one to one mapping of check nodes to CNUs can overcome this drawback. An example for a column-based decoder is given in Figure 4. All M CNUs and $4 \times P$ VNUs are instantiated. This architecture requires partly parallel CNUs, processing $4 \times P$ edges per cycle. Moreover, the VNUs need to be implemented fully parallel, consuming and generating M edges per cycle. Due to the higher degree of parallelism, this hardware mapping needs only eight cycles per iterations. All CNUs work in parallel and thus produce the CN to VN messages just in time. This makes memories between CNs and VNs redundant and can save a significant amount of area. However, for decoders instantiating all M CNUs for column-wise decoding, the hardware mapping restricts the possible scheduling to the two-phase approach. This is due to the parallel working CNUs making intermediate VNU updates impossible. Moreover, our investigations have shown that, due to their complexity, CNUs need significantly more area than VNUs. Especially for highly parallel decoders, instantiating a huge number of functional units, this fact is most important. In Section 5, it is shown that, for the same area, a higher degree of parallelism can be achieved by parallel instantiation of VNUs instead of CNUs. We will also show that these two drawbacks of the column-based decoder overcome the benefit of saved area in comparison with the row-based design.

For a row-based decoder, all N VNs are instantiated as VNUs and a multiple of P number of CNUs. Figure 5 highlights the part of the parity check matrix H which is processed in the first cycle. Serially working VNUs and fully parallel CNUs processing N edges per cycle are required. The given example instantiates P CNUs and needs only four cycles per iteration.

Furthermore, all three approaches can be extended by processing not only one row or column but instead several of them in parallel. Due to the parity check matrix structure, usually a multiple of the submatrix size is chosen, which yields a good unit utilization, see Table 1.

3.4. Pipelining. Adding register stages in the decoder core efficiently shortens the critical paths. This can be exploited to reduce the area by smaller but slower cells, or higher frequencies can be achieved within the same area. Regardless of these benefits, the pipeline has to be filled each iteration, which introduces a clock cycle penalty per stage and iteration. In most serially working decoders, this does no harm to the overall throughput because even without the pipeline delays, a high number of cycles is needed per iteration. In contrast to this, highly parallel architectures are processing one complete iteration in only a few clock cycles, often less than ten. For these decoders, each pipeline stage adding one cycle to the iteration period has an enormous impact on the throughput. Therefore, for massive parallel decoder architectures, the number of pipeline stages has to be kept as low as possible.

3.5. Frame Handling. The latency introduced by pipelining has strong impact on massive parallel iterative decoders

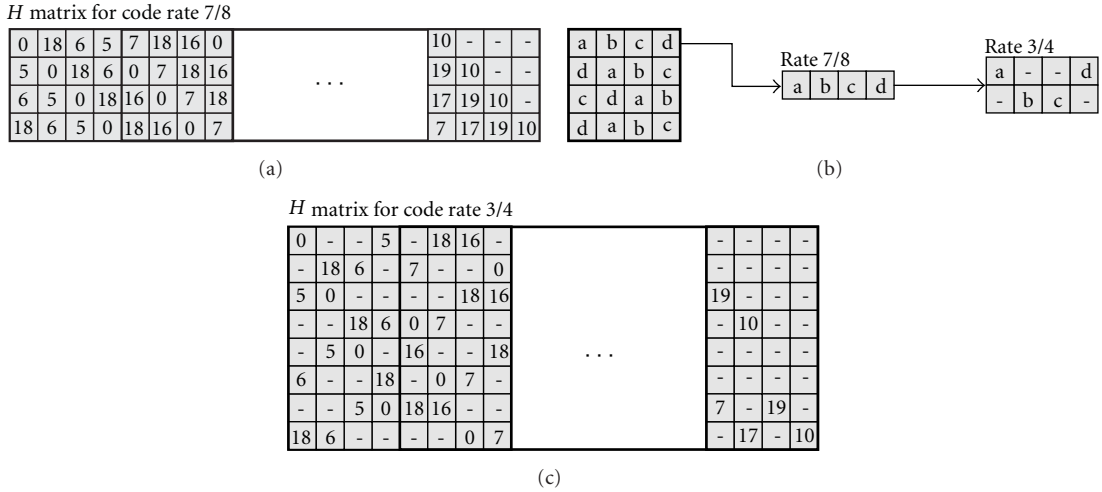

Figure 6: Row split pattern for IEEE 802.15.3c parity check matrix.

due to the data dependencies. To overcome this problem, several independent frames can be processed by the decoder in parallel. This frame-interleaved pipelining is comparable to pipelining as it is done in a modern general purpose CPU. A very high hardware utilization can be achieved by this technique because pipeline stalls hardly appear in this application. Due to the additional complexity and implementation effort introduced by this approach, there are only few decoders making use of this feature. However, for future decoders especially in the high throughput domain, where only few cycles per iteration are needed, handling of multiple frames should be concerned.

3.6. Matrix Structure Design. The parity check matrix H has significant impact on the decoder design. For high throughput applications-typically very sparse matrices are used to reduce the computational complexity. An innovation introduced in upcoming standards like [2, 3] is a special method to generate matrices for different code rates. By the so-called row splitting, the matrices for lower code rates are derived from one base matrix. Figure 6(a) shows the base matrix of the IEEE 802.15.3c code. The bold divided parts of the matrix are the so-called macroblocks. The principle of row splitting is presented in Figure 6(b). From each row of the base matrix, multiple rows for the lower code rate matrices are generated, see Figure 6(c). It can be observed that the number of nonempty entries never increases. This matrix design allows for efficient decoder hardware implementations. CNUs processing all nonzero entries of one macroblock need only little modification to split the inputs to several independently processed groups of inputs. Thus, the arising bigger macroblocks can still be processed by the same hardware, gaining flexibility while minimizing the hardware overhead.

3.7. Check Node Approximation. The optimal belief propagation algorithm SPA as it is presented in Section 2 is usually replaced by suboptimal solutions. Dependant on the requirements, an appropriate approximation needs to be chosen. If the requirements are tight, a more accurate algorithm has to be chosen, for example, the λ-Min [10] algorithm allows for a good communications performance for a wide range of code rates. If a performance degradation for low code rates is acceptable, a Min-Sum-based [11] algorithm can reduce the implementation complexity even further. This is also the algorithm of choice in multigigabit applications where a high degree of parallelism is applied, and thus the use of a suboptimal algorithm saves a significant amount of area. Additionally, applying extrinsic memory compression [10] can reduce the need for extrinsic message RAMs. Therefore, the extrinsic messages are not stored for each outgoing edge, but reconstructed, in the case of Min-Sum from only two messages. All decoders currently published for multigigabit standards make use of the Min-Sum Algorithm and apply extrinsic memory compression.

3.8. Connectivity. Different types of connectivity can be considered. For instance, fully parallel decoder architectures use a hard-wired connectivity. For partly parallel decoders however, flexible permutation networks are required [12, 13]. The most common interconnection networks are barrel shifters. They support the cyclically shifted identity matrices of structured codes. As mentioned before upcoming standards like IEEE 802.15.3c use a special row split pattern to generate different code rates. For efficient support of these kind of codes, custom networks are required in addition to the shifters. Furthermore, the parallelism of the interconnection network has to be considered. Most high throughput decoders found in literature pass messages bit-parallel. However, this results in routing dominated architectures. Bit-serial message passing drastically reduces the need for interconnection wires and relaxes the routing congestions. This leads to very power and area efficient fully parallel decoder designs as shown in [9]. Currently, there are to the best of our knowledge no publications applying

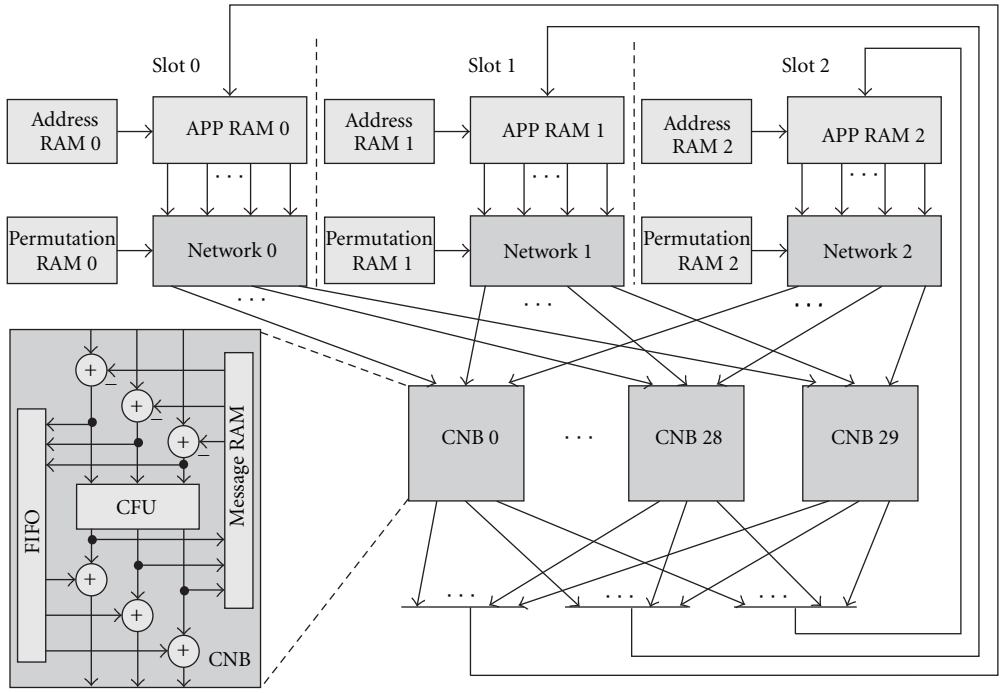

Figure 7: Slot-Layered low-parallel LDPC decoder, WiMedia 1.5 based.

bit-serial message passing to flexible partly parallel decoder architectures.

All these issues must not be considered as isolated decisions, they rather interact. In [14, 15], the interrelations are discussed in more detail. For example, the scheduling has impact on the required decoder parallelism, or a hard-wired interconnection is prohibitive for partly parallel architectures.

4. Decoder Architectures

The discussed design space allows for a variety of architectures. We present three decoder architectures in more detail to give a better understanding of the interoperation of the numerous design decisions. Because the hardware mapping has a major impact on the resulting architecture, each decoder uses a different mapping.

The first decoder we present makes use of a low-parallel hardware mapping and is designed for a WiMedia 1.5 based code. The architecture uses the concepts presented in [16], it is shown in Figure 7. Each set of 30 variable nodes is attributed to one of the three slots. One slot processes one submatrix per clock cycle. In contrast to a serial architecture, each of the check node blocks (CNBs) reads and writes three edges in parallel, with each edge coming from a different slot. The minimum search within the CNU is performed according to the Min-Sum algorithm. The variable node operations according to (6) and (7) are processed within the CNBs and do not require an explicit functional unit. This architecture gains an excellent flexibility while serving throughputs of more than 1 Gbit/s. Moreover, it supports layered decoding, resulting in much faster convergence than a comparable two-phase decoder.

The next approach we present implements the column-based hardware mapping. Figure 8 shows the decoder's basic architecture. The main building blocks are the VNUs, CNUs, and the permutation networks. As mentioned before, the different code rate matrices of the IEEE 802.15.3c standard are generated by the presented row-split pattern. A specialty for this decoder is the custom networks (CNWs) which are introduced before and after the CNUs. They are used to support the different code rates and introduce only little hardware overhead. This decoder shows how the extrinsic message RAM can be removed by the fully parallel instantiation of CNs. As explained before, the four groups of CNUs process all CNs at the same time. Thus extrinsic messages can immediately be consumed by the variable nodes. However, this architecture is only capable of two-phase decoding. This leads to a significantly increased number of iterations and thus lowers the efficiency of the architecture as will be discussed in Section 5.

Finally, a row-based hardware mapping for the IEEE 802.15.3c standard is investigated. The high parallelism of the decoder arises from the fully parallel instantiated VNUs. 672 VNUs and 21 CNUs in number are used to process one quarter of the parity check matrix in parallel. The basic structure is similar to the one of the low parallel decoder presented before. However, instead of only three, now 32 slots are instantiated, see Figure 9. This architecture contains an extrinsic RAM which is needed because the matrix is processed row-wise, the intermediate data has to be stored for the next iteration. Because the VNUs work serially, each

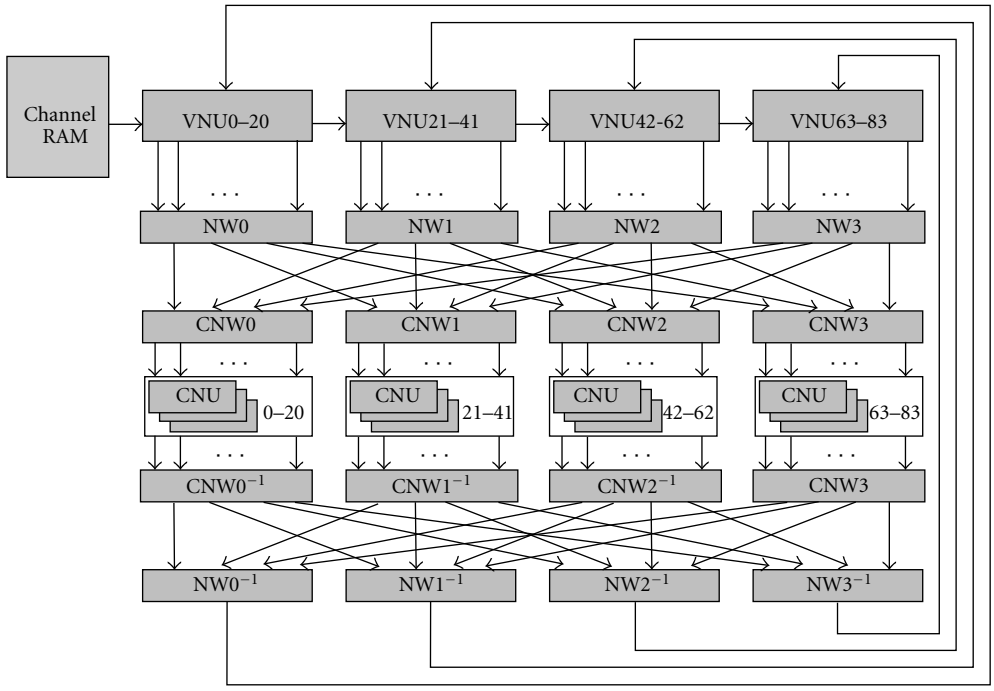

Figure 8: Column-based LDPC decoder architecture for IEEE 802.15.3c standard.

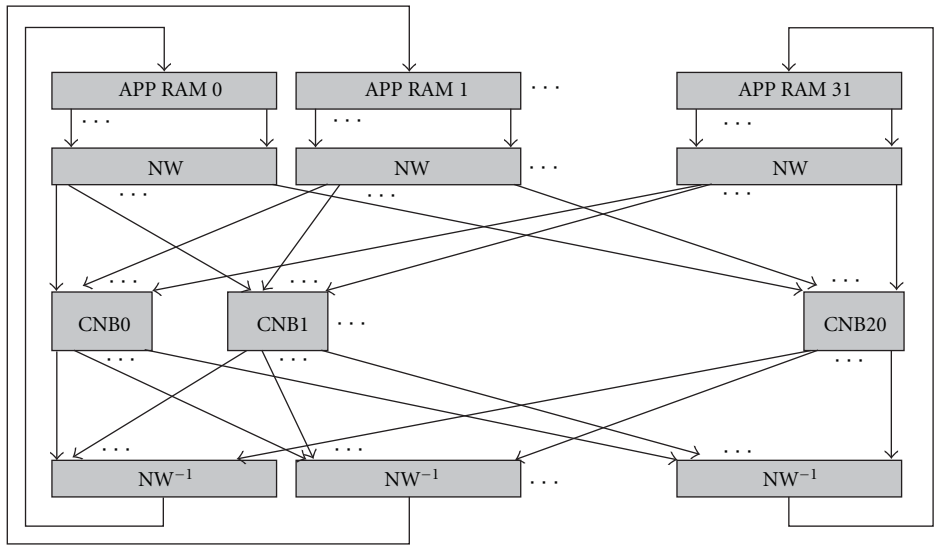

Figure 9: Row-based LDPC decoder architecture for IEEE 802.15.3c standard.

producing and consuming one edge per cycle, they are small in comparison with the partly parallel ones of the column-based decoder. The CNUs each are capable to process 32 edges which makes them complex to implement. However, the small number of CNU instances keeps the overall area demand in an acceptable range. Finally, the architecture has a very good VNU to CNU ratio, which results in a high efficiency as we discuss in the following section.

5. Decoder Evolution

Several approaches to design flexible high throughput LDPC decoders have been published in the past. However it is not yet investigated which of these designs is preferable in means of area and energy efficiency.

Estimating the decoders parameters before the implementation is a very complex task. Many factors like the

Table 2: Multigigabit decoder evolution.

	This paper	WPAN [17]	This paper	WPAN [18]	WiGig [19]	WPAN [20]
Standard	WiMedia 1.5-based	IEEE 802.15.3c draft	IEEE 802.15.3c	IEEE 802.15.3c	IEEE 802.11ad	IEEE 802.15.3c
Technology	65 nm	65 nm	65 nm	65 nm	65 nm	65 nm
Postplace and route	No	No	Yes	Yes	No	Yes
Codeword length	600	576	672	672	672	672
Code rates	1/2, 5/8, 3/4, 4/5	1/2, 3/4, 7/8	1/2, 5/8, 3/4, 7/8	1/2, 5/8, 3/4, 7/8	1/2, 5/8, 3/4, 13/16	1/2, 5/8, 3/4, 7/8
Submatrix size	15	18	21	21	42	21
Edges/cycle	45	288	336	672	672	672
$F_{\max}$ **[MHz]**	400	500	500	197	150	400
Scheduling	Layered	Two-phase	Two-phase	Layered	Two-phase	Layered
Algorithmic iterations	5	7	9	4	N/A	9
Cycles/iteration	40–45	10	11	4	4	4
Pipeline stages	3	2	3	0	4	4
Frame interleaved	No	No	No	No	Yes	Yes
Input quantization	6	6	5	6	N/A	6
Hardware mapping	Other	Column-based	Column-based	Row-based	Row-based	Row-based
VNU degree	1	4	4	1	1	1
CNU degree	3	1–4	1–4	8–32	8–16	8–32
VNU instances	45	72	84	672	672	672
CNU instances	15	72–288	84–336	21–84	42–84	21–84
Logic Area	30%	N/A	62%	N/A	N/A	68%
Memory area	70%	N/A	38%	N/A	N/A	32%
Area [mm^2]	0.37	0.50	1.15	1.56	1.30	1.30
Supply [V]	1.2	N/A	1.2	1.0	0.8	1.2
Power [mW]	N/A	N/A	630	361	84	538
Energy eff. [pJ/bit/Iter]	N/A	N/A	21	13	7	8
Thr. [Gbit/s/mm^2]	2.62	7.20	2.66	3.24	2.37	5.17
Thr. [bit/Cycle]	2.43	7.20	6.11	33.60	20.50	16.80
Thr. Air [Gbit/s]	0.97	3.60	3.06	6.62	3.08	6.72

matrix structure, underlying memory generators as well as the supported flexibility, have huge impact on the results and cannot be expressed analytically. For fully parallel decoders without flexibility, estimations are possible and recently published in [21]. Table 2 gives a detailed comparison of flexible multigigabit decoders. All the design space parameters presented in the previous section are pointed out. Thus, the table can be used to get an impression of the influence of the design space parameters.

A large overhead is introduced by routing which impacts area and energy. It makes synthesis results only not conclusive. Only for three of the decoders postplace and route (PAR), results are available which have to be considered for comparison.

State-of-the-art LDPC decoders process only few submatrices per cycle, and their area is dominated by memory. They make use of a low-parallel hardware mappings which results in poor logic to memory ratios as pointed out earlier. In the WiMedia 1.5 based decoder, we present in Table 2, 70% of the overall area is spent for memory.

As described in Section 3, instantiating all CNUs can solve this issue. A decoder using this approach for a draft of the IEEE 802.15.3c standard was first published in [17]. However, the mentioned paper presents only synthesis data which are not comparable to post PAR results. Therefore, we present a column-based decoder built in a 65 nm process using a low-power, low V_T library. Synthesis as well as PAR is executed under worst case conditions, ending up with 74% standard cell utilization. The results in Table 2 show the achieved logic domination for this architecture. Figure 10 underlines the area demand of the CNU groups. Only 38% of the overall area is used by memory, which is mainly made up by the channel RAM and pipeline registers. However, it has been shown that, due to their complexity, CNUs need more area than VNUs. For a given area, a higher parallelism can be achieved by increasing the number of VNUs instead of the CNUs as discussed in the previous section. The column-based design, using massive parallel instantiated CNUs, thus needs more area than a rowbased design with the same degree of parallelism. This can be

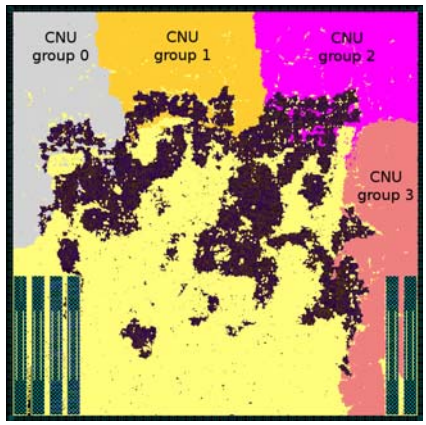

Figure 10: Physical design of the column-based LDPC decoder. The four check node groups are located in the top and right. The channel RAM is split in two parts and located in the bottom corners.

seen in the direct comparison of our column-based and the row-based decoder presented in [18]. Even though all 672 VNUs are instantiated and thus the parallelism is doubled in comparison to the column-based decoder, there is only a small growth in area. This yields an increased throughput per area result of 3.24 Gbit/mm^2.

For a further comparison of the logic to memory area ratio, the decoder presented in [20] is considered. Even though the row-based design needs extrinsic message RAM for the check to variable node messages, overall only 32% of the area is memory area. The memory domination seen in low parallelism decoders like the WiMedia-based decoder presented in this paper is no more an issue for massive parallel decoders. Due to the high number of instantiated VNUs, the RAMs play only a minor role in the overall area utilization. The other important difference between the row and the column-based decoder is the scheduling. While the column-based decoder is restricted to two-phase scheduling, row-based decoders can use layered scheduling. By this difference, the communications performance can be improved or, like in [18], the throughput can be enhanced by reduction of the iterations.

An architecture proposed in [19] for the IEEE 802.11ad standard uses a similar approach as the one presented before. The decoder is also row-based and thus instantiates all variable nodes. Even though area and power consumption are only estimated, the results show that this approach can also be adapted to other multigigabit applications.

The next evolutionary step is taken by the decoder presented in [20]. By the interleaved frame handling, regardless of the pipelining, no wait state is introduced. Finally, this represents the most advanced decoder architecture for IEEE 802.15.3c and is also a candidate for other multigigabit standards featuring LDPC codes.

6. Conclusion

In this paper, we have in detail presented the design space for flexible multigigabit LDPC decoders. The memory domination as known from state-of-the-art decoders is investigated in the context of highly parallel decoder architectures. It is pointed out that the need for memories can be reduced by column-based decoders because intermediate message storage becomes obsolete. Our systematic investigations have shown that, in row-based designs, the memory domination disappears because of the large amount of logic introduced by the high degree of parallel instantiated functional units. Further exploring the design space, two new decoder architectures serving throughputs of one and three Gbit/s are presented including post PAR results. Especially row- and column-based hardware mappings are investigated. It is shown that the row-based hardware mapping has significant benefits compared to column-based designs because of the VNU to CNU ratio and the possibility to apply layered scheduling.

Acknowledgment

This work has been supported by the Deutsche Forschungsgemeinschaft (DFG) within the project "Entwicklung effizienter und flexibler VLSI-Architekturen für die Kanaldecodierung in drahtlosen Multi-Gigabit-Kommunikationssystemen auf der Basis von LDPC-Codes."

References

[1] WiMedia Alliance, "Multiband OFDM physical layer specification, release candidate 1.5," March 2009.

[2] IEEE 802.15.3c, "Part 15.3: wireless medium access control (MAC) and physical layer (PHY) specifications for high rate wireless personal area networks (WPANs)," IEEE 802.15.3c-2009, 2009.

[3] IEEE 802.11ad, ""Part 11: wireless LAN medium access control (MAC) and physical layer (PHY) specifications—amendment: enhancements for very high throughput in the 60 GHz band," IEEE 802.11ad-draft, 2010.

[4] R. G. Gallager, "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[5] J. Zhang and M. Fossorier, "Shuffled belief propagation decoding," in *Proceedings of the 36th Asilomar Conference on Signals Systems and Computers*, pp. 8–15, November 2002.

[6] D. E. Hocevar, "A reduced complexity decoder architecture via layered decoding of LDPC codes," in *Proceedings of the IEEE Workshop on Signal Processing Systems Design and Implementation (SiPS'04)*, pp. 107–112, Austin, Tex, USA, October 2004.

[7] J. Dielissen, A. Hekstra, and V. Berg, "Low cost LDPC decoder for DVB-S2," in *Proceedings of the Design, Automation and Test in Europe (DATE'06)*, Munich, Germany, March 2006.

[8] A. J. Blanksby and C. J. Howland, "A 690-mW 1-Gb/s 1024-b, rate-1/2 low-density parity-check code decoder," *IEEE Journal of Solid-State Circuits*, vol. 37, no. 3, pp. 404–412, 2002.

[9] M. Korb and T. G. Noll, "Area and latency optimized high-throughput Min-Sum based LDPC decoder architectures," in *Proceedings of the 35th European Solid-State Circuits Conference (ESSCIRC'09)*, pp. 408–411, September 2009.

[10] F. Guilloud, E. Boutillon, and J. Danger, "λ-Min decoding algorithm of regular and irregular LDPC codes," in *Proceedings of the 3rd International Symposium on Turbo Codes & Related Topics*, pp. 451–454, Brest, France, September 2003.

[11] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X.-Y. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Transactions on Communications*, vol. 53, no. 8, pp. 1288–1299, 2005.

[12] G. Masera, F. Quaglio, A. Tarable, and F. Vacca, "Interconnection structure for a flexible LDPC decoder," in *Proceedings of the Wireless Reconfigurable Terminals and Platforms (WiRTeP'06)*, pp. 58–62, April 2006.

[13] M. Rovini, G. Gentile, F. Rossi, and L. Fanucci, "A scalable decoder architecture for IEEE 802.11n LDPC codes," in *Proceedings of the Global Telecommunications Conference (GLOBECOM'07)*, Washington, DC, USA, November 2007.

[14] T. Brack, F. Kienle, and N. Wehn, "Disclosing the LDPC code decoder design space," in *Proceedings of the Design, Automation and Test in Europe (DATE'06)*, pp. 200–205, Munich, Germany, mARCH 2006.

[15] F. Kienle, *Implementation issues of low-density parity-check decoders*, Ph.D. dissertation, University of Kaiserslautern, 2006.

[16] M. Alles, F. Berens, and N. Wehn, "A synthesizable IP core for WiMedia 1.5 UWB LDPC code decoding," in *Proceedings of the IEEE International Conference on Ultra-Wideband (ICUWB'09)*, pp. 591–601, Vancouver, Canada, September 2009.

[17] J. Sha, J. Lin, Z. Wang, L. Li, and M. Gao, ""Ldpc decoder design for high rate wireless personal area networks," *IEEE Transactions on Consumer Electronics*, vol. 55, no. 2, pp. 455–460, 2009.

[18] S.-Y. Hung, S.-W. Yen, C.-L. Chen, H.-C. Chang, S.-J. Jou, and C.-Y. Lee, "A 5.7 Gbps row-based layered scheduling LDPC decoder for IEEE 802.15.3c applications," in *Proceedings of the IEEE Asian Solid-State Circuits Conference (A-SSCC'10)*, pp. 309–312, Beijing, China, 2010.

[19] M. Weiner, B. Nikolic, and Z. Zhang, "LDPC decoder architecture for high-data rate personal-area networks," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'11)*, pp. 1784–1787, 2011.

[20] Z. Chen, X. Peng, X. Zhao et al., "A macro-layer level fully parallel layered LDPC decoder SOC for IEEE 802.15.3c application," in *Proceedings of the International Symposium on VLSI Design, Automation and Test (VLSI-DAT'11)*, pp. 298–301, 2011.

[21] M. Korb and T. G. Noll, "Ldpc decoder area, timing, and energy models for early quantitative hardware cost estimates," in *Proceedings of the International Symposium on System-on-Chip (SoC'10)*, pp. 169–172, 2010.