We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

6

LDPC Decoders for the WiMAX (IEEE 802.16e) based on Multicore Architectures

Gabriel Falcão, Vitor Silva, José Marinho and Leonel Sousa
Instituto de Telecomunicações/ FCTUC/University of Coimbra
INESC-ID/IST/Technical University of Lisbon
Portugal

1. Introduction

The WiMAX is a wireless communications standard (IEEE 802.16e) used in small to medium distances in urban areas, that uses Low-Density Parity-Check (LDPC) codes, which are powerful error correcting codes very demanding from a computational perspective [Gallager, 1962; Mackay & Neal, 1996]. The throughput requirements imposed by the WiMAX standard define a maximum of 75 Mbps [IEEE P802.16e/D12, 2005], which typically demands a high number of arithmetic operations and memory accesses per second. In order to accommodate such requirements hardware-dedicated solutions were investigated and developed to deliver such computational power.

With the number of transistors on a die approximately doubling every 18 months, we have seen in last years processors scaling up to hundreds of millions of gates. To overcome power and memory wall constraints, the industry of processors has introduced a new trend: increasing the number of cores per processor rather than using higher clock speeds. Associated to this new paradigm, new kinds of different homogeneous and heterogeneous multicore architectures have been proposed. Initially developed essentially for rendering purposes in the industry of games, recently we have seen multicores start offering new possibilities to support general purpose computing. Nowadays largely disseminated worldwide and supported by appropriate tools, they can be exploited by convenient parallel programming models delivering unmatched performances. This reality introduced the massive generalization of general purpose processing in these parallel architectures.

So far, dedicated VLSI was the only solution capable of decoding LDPC codes at acceptable rates [Brack et al., 2006; Liu et al., 2008]. In this chapter we present a novel, programmable/flexible and scalable parallel LDPC decoding approach for the WiMAX standard based on multicore accelerators such as the Cell/B.E. architecture from Sony-Toshiba-IBM (STI) [Hofstee, 2005; International Business Machines Corporation, 2007; Falcão et al., 2008]. Moreover, we exploit parallel programming models and present parallel algorithms for this architecture [Falcão et al., 2009a]. We also report experimental results and compare them with state-of-the-art LDPC hardware decoding solutions, based on Application Specific Integrated Circuits (ASIC).

Besides the introductory and closure sections, we propose to organize this chapter around five other main sections. The second and third sections introduce, respectively, WiMAX LDPC codes and the algorithms used to perform LDPC decoding. The main characteristics of the Cell/B.E. multicore architecture are presented in section four. The fifth section describes the parallel programming models and the parallel algorithms developed to efficiently implement LDPC decoding. Finally, the experimental results section shows that the obtained throughputs compare fairly against state-of-the-art ASIC LDPC decoders.

2. WiMAX LDPC codes

The WiMAX standard (IEEE 802.16e) works in distances typically below the 10 Km range [Falcão et al., 2008], and uses LDPC codes, whose decoders can be very demanding from a computational perspective. For this reason, they are still implemented using dedicated hardware based on ASIC solutions. LDPCs are linear (n, k) block codes [Lin & Costello, 2004] defined by sparse binary parity-check **H** matrices of dimension (n-k) x n, and represented by bipartite graphs also called Tanner graphs [Tanner, 1981]. The Tanner graph is formed by Bit Nodes (BNs) and Check Nodes (CNs) linked by bidirectional edges. An example is shown in figure 2.

The Forward Error Correcting (FEC) system of the WiMAX standard is based on a special class of LDPC codes [IEEE P802.16e/D12, 2005] characterized by a sparse binary block parity-check matrix **H** of the form:

$$\mathbf{H}_{(n-k)\times n} = \left[\mathbf{H}1 \mid \mathbf{H}2\right] = \left[\begin{array}{cccc|c|ccccc} \mathbf{P}_{0,0} & \mathbf{P}_{0,1} & \cdots & \mathbf{P}_{0,k/z-1} & \mathbf{Pb}_0 & \mathbf{I} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{P}_{1,0} & \mathbf{P}_{1,1} & \cdots & \mathbf{P}_{1,k/z-1} & & \mathbf{I} & \mathbf{I} & & & \vdots \\ \vdots & \ddots & & \vdots & & \mathbf{0} & \mathbf{I} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots & \vdots & \ddots & \ddots & \ddots & \mathbf{I} & \mathbf{0} \\ \mathbf{P}_{n-k/z-2,0} & \mathbf{P}_{n-k/z-2,1} & \cdots & \mathbf{P}_{n-k/z-2,k/z-1} & \vdots & & \ddots & & \mathbf{I} & \mathbf{I} \\ \mathbf{P}_{n-k/z-1,0} & \mathbf{P}_{n-k/z-1,1} & \cdots & \mathbf{P}_{n-k/z-1,k/z-1} & \mathbf{Pb}_{n-k/z-1} & \mathbf{0} & & \cdots & \mathbf{0} & \mathbf{I} \end{array}\right], \quad (1)$$

where **H1** is sparse and adopted special periodicity constraints introduced in the pseudo random design of the matrix [IEEE P802.16e/D12, 2005; Brack et al., 2006] and **H2** is a sparse lower triangular block matrix with a staircase profile. The periodic nature of these codes defines **H1** based on permutation sub-matrices $\mathbf{P}_{i,j}$, which are: (1) quasi-random circularly shifted right identity sub-matrices (**I**) (as depicted in figure 1) , with dimensions $z \times z$ ranging from 24×24 to 96×96 and incremental granularity of 4 (table 1); or (2) $z \times z$ null sub-matrices. The periodic nature of such codes allowed simplifying the architecture of the system and storage requirements without code performance loss [IEEE P802.16e/D12, 2005]. Also, the right sub-matrix **H2** is formed by identity: (1) **I** sub-matrices of dimension $z \times z$; or (2) null sub-matrices of dimension $z \times z$.

Therefore, the LDPC codes adopted by the WiMAX standard (IEEE 802.16e) support 19 different codeword sizes with 4 distinct code rates (*rate*=k/n) and 6 different class codes (distinct distributions of the number of BNs per column or CNs per row). They are depicted

in table 1. Class $2/3A$ defines codes having {2, 3, 6} BNs per row and {10} CNs per column, while class $2/3B$ codes have {2, 3, 4} BNs and {10, 11} CNs. Also, class $3/4A$ has {2, 3, 4} BNs and {14, 15} CNs, and class $3/4B$ has {2, 3, 6} BNs per row and {14, 15} CNs per column.

Code	Codeword bits (n)	$z \times z$ factor	Information bits (k)			
			(rate) 1/2	*(rate)* 2/3	*(rate)* 3/4	*(rate)* 5/6
1	576	24×24	288	384	432	480
2	672	28×28	336	448	504	560
3	768	32×32	384	512	576	640
4	864	36×36	432	576	648	720
5	960	40×40	480	640	720	800
6	1056	44×44	528	704	792	880
7	1152	48×48	576	768	864	960
8	1248	52×52	624	832	936	1040
9	1344	56×56	672	896	1008	1120
10	1440	60×60	720	960	1080	1200
11	1536	64×64	768	1024	1152	1280
12	1632	68×68	816	1088	1224	1360
13	1728	72×72	864	1152	1296	1440
14	1824	76×76	912	1216	1368	1520
15	1920	80×80	960	1280	1440	1600
16	2016	84×84	1008	1344	1512	1680
17	2112	88×88	1056	1408	1584	1760
18	2208	92×92	1104	1472	1656	1840
19	2304	96×96	1152	1536	1728	1920

Table 1. Properties of LDPC codes used in the WiMAX IEEE 802.16e standard.

To illustrate the periodic nature introduced in the design of the **H** matrix, the generic structure of a sample parity-check matrix used in WiMAX, with n=2304 and *rate*=1/2, is depicted in figure 1.

3. Algorithms for LDPC decoding (SPA and MSA)

Computationally intensive message-passing algorithms can be used for LDPC decoding, varying from the well known belief propagation, or Sum-Product algorithm (SPA) [Falcão et al., 2009a] to the more efficient but yet still intensive Min-Sum algorithm (MSA) [Falcão et al., 2008]. The intensive message passing procedure is illustrated in figure 2. We will discuss the parallelization of the SPA and MSA, and show how data dependencies can be manipulated in order to allow the implementation of parallel decoders.

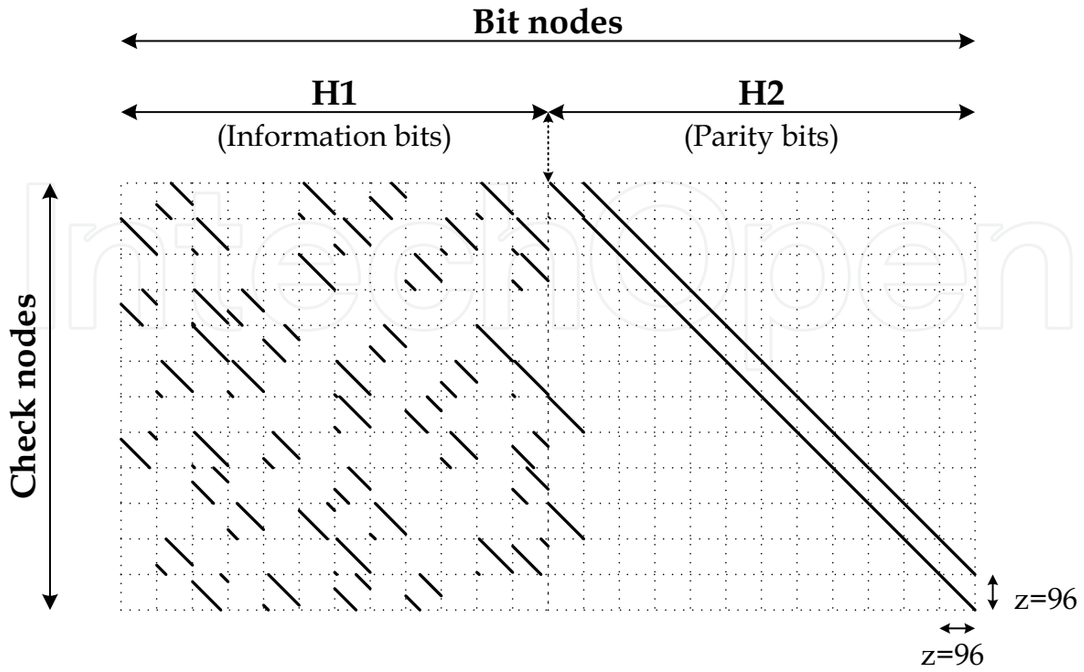
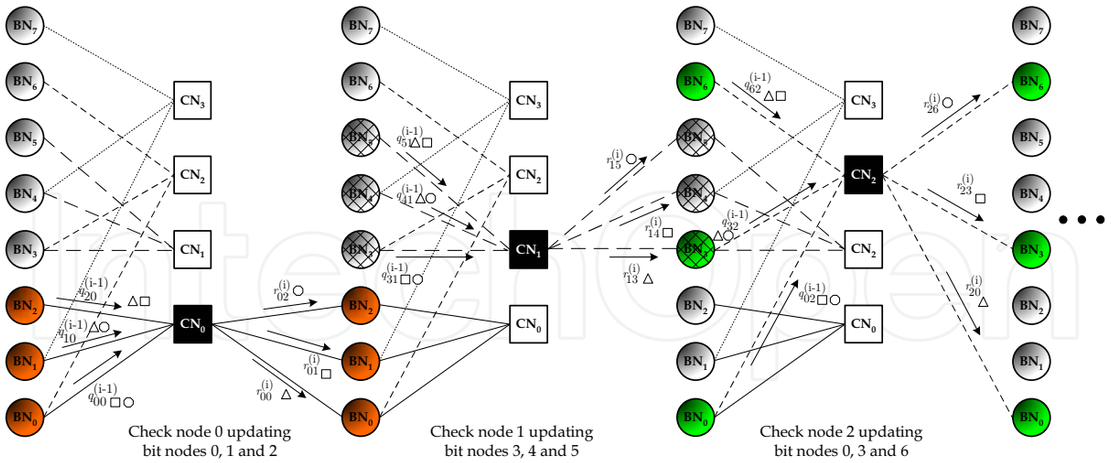


Fig. 1. Periodicity $z \times z = 96 \times 96$ for a matrix with n=2304, rate=1/2 and {3, 6} CNs per column.

3.1 The Sum-Product Algorithm (SPA)

The SPA is a very efficient algorithm [Lin & Costello, 2004] used for LDPC decoding and it is based on the belief propagation between adjacent nodes connected as indicated by the Tanner graph edges (figure 2). As proposed by Gallager, the SPA operates on probabilities [Gallager, 1962; Mackay & Neal, 1996; Lin & Costello, 2004]. Given a (n, k) LDPC code, we assume Binary-Phase Shift Keying (BPSK) modulation which maps a codeword c = (c_1, c_2, ... , c_n) into the sequence x = (x_1, x_2, ... , x_n), according to $x_i = (-1)^{c_i}$. Then, x is transmitted through an additive white Gaussian noise (AWGN) channel originating the received sequence y = (y_1, y_2 , ... , y_n) on the decoder side, with $y_i = x_i + n_i$, where n_i is a random variable with zero mean and variance σ^2.

The SPA is depicted from (2) to (8) [Lin & Costello, 2004]. It is mainly described by two horizontal and vertical intensive processing blocks, respectively defined by equations (3), (4) and (5), (6). The first two calculate messages moving from each CN_m to BN_n, considering accesses to **H** on a row basis. It indicates the probability of BN_n being 0 or 1. Figure 2 exemplifies, for a particular 4 x 8 **H** matrix, BN_0, BN_1 and BN_2 being updated by CN_0, then BN_3, BN_4 and BN_5 updated by CN_1 and finally BN_0, BN_3, BN_6 by CN_2. Similarly, the vertical processing block computes messages sent from BN_n to CN_m, assuming accesses on a column basis. The iterative procedure is stopped if the decoded word $\hat{\mathbf{c}}$ verifies all parity-check equations of the code $\hat{\mathbf{c}}.\mathbf{H}^T = \mathbf{0}$, or a maximum number of iterations I is reached, in which case no codeword is detected.



(8 bit nodes checked by 4 check node equations)

$$\mathbf{H} = \begin{array}{c} \begin{matrix} \mathbf{BN_0} & \mathbf{BN_1} & \mathbf{BN_2} & \mathbf{BN_3} & \mathbf{BN_4} & \mathbf{BN_5} & \mathbf{BN_6} & \mathbf{BN_7} \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} \mathbf{CN_0} \\ \mathbf{CN_1} \\ \mathbf{CN_2} \\ \mathbf{CN_3} \end{matrix}$$

Fig. 2. A 4x8 **H** matrix and corresponding Tanner graph representation. The example shows messages being exchanged from CN_m to BN_n. A similar representation applies for messages going in the opposite direction.

$$q_{nm}^{(0)}(0) = 1 - p_n;\ q_{nm}^{(0)}(1) = p_n;$$
$$p_n = \frac{1}{1 + e^{\frac{2y_n}{\sigma^2}}};\ \frac{E_b}{N_0} = \frac{N}{2K\sigma^2}\{Initialization\} \tag{2}$$

while ($\hat{\mathbf{c}}.\mathbf{H}^T \neq \mathbf{0} \cap \mathbf{i} < \mathbf{I}$) { $\hat{\mathbf{c}}$-decoded word; **I**-max. number of iterations.}
do
{For each node pair (BN_n, CN_m), corresponding to $\mathbf{H_{mn}} = \mathbf{1}$ in the parity-check matrix **H** of the code **do**:}
{1. **Horizontal Processing (kernel 1)** – Compute messages sent from CN_m to BN_n, that indicate the probability of BN_n being **0** or **1**:}

$$r_{mn}^{(i)}(0) = \frac{1}{2} + \frac{1}{2}\prod_{n' \in N(m)\backslash n}\left(1 - 2q_{n'm}^{(i-1)}(1)\right) \tag{3}$$

$$r_{mn}^{(i)}(1) = 1 - r_{mn}^{(i)}(0) \tag{4}$$

{where $N(m)\backslash n$ represents BNs connected to CN_m excluding BN_n.
{2. **Vertical Processing (kernel 2)** – Compute messages sent from BN_n to CN_m:}

$$q_{nm}^{(i)}(0) = K_{nm}(1-p_n)\prod_{m'\in M(n)\setminus m} r_{m'n}^{(i)}(0) \tag{5}$$

$$q_{nm}^{(i)}(1) = K_{nm}p_n\prod_{m'\in M(n)\setminus m} r_{m'n}^{(i)}(1) \tag{6}$$

{where k_{nm} are chosen to ensure $q_{nm}^{(i)}(0)+q_{nm}^{(i)}(1)=1$, and M(n)\m is the set of CNs connected to BN_n excluding CN_m.}
{3. Compute *a posteriori* pseudo-probabilities:}

$$\begin{aligned} Q_n^{(i)}(0) &= K_n(1-p_n)\prod_{m\in M(n)} r_{mn}^{(i)}(0) \\ Q_n^{(i)}(1) &= K_n p_n\prod_{m\in M(n)} r_{mn}^{(i)}(1) \end{aligned} \tag{7}$$

{where k_n are chosen to guarantee $Q_n^{(i)}(0)+Q_n^{(i)}(1)=1$.}
{Perform hard decoding} $\forall n$,

$$\hat{c}_n^{(i)} = \begin{cases} 1 \Leftarrow Q_n^{(i)}(1) > 0.5 \\ 0 \Leftarrow Q_n^{(i)}(1) < 0.5 \end{cases} \tag{8}$$

end while

3.2 The Min-Sum Algorithm (MSA)

The MSA [Guilloud et al., 2003] is a simplification of the well-known SPA in the logarithmic domain. It is also based on the intensive belief propagation between nodes connected as indicated by the Tanner graph edges, but that uses only comparison and addition operations. Being one of the most efficient algorithms used for LDPC decoding [Liu et al., 2008; Seo et al., 2007], even so, the MSA still requires intensive processing.
Let us denote the log-likelihood ratio LLR of a random variable as *L(x) = ln (p(x = 0)/p(x = 1))*. Also, considering the message propagation from nodes CN_m to BN_n and vice-versa, the set of bits that participate in check equation *m* with bit *n* excluded is represented by *N(m)\n* and, similarly, the set of check equations in which bit *n* participates with check *m* excluded is *M(n)\m*. LP_n designates the *a priori* LLR of BN_n, derived from the values received from the channel, and Lr_{mn} the message that is sent from CN_m to BN_n, computed based on all received messages from BNs *N(m)\n*. Also, Lq_{nm} is the LLR of BN_n, which is sent to CN_m and calculated based on all received messages from CNs *M(n)\m* and the channel information LP_n. For each node pair (BN_n, CN_m) we initialize Lq_{nm} with the *a priori* log-likelihood ratio (LLR) information received from the channel, LP_n. Then, we proceed to the iterative body of the algorithm by calculating (9) and (10), respectively, where Lr_{mn} denotes the message sent from CN_m to BN_n, and Lq_{nm} the message sent from BN_n to CN_m.

{1. **Horizontal Processing (kernel 1)** – Compute messages sent from CN_m to BN_n:}

$$Lr_{mn}^{(i)} = \prod_{n' \in N(m) \setminus n} sign\left(Lq_{n'm}^{(i-1)}\right) \min_{n' \in N(m) \setminus n} \left|Lq_{n'm}^{(i-1)}\right|. \tag{9}$$

{2. **Vertical Processing (kernel 2)** – Compute messages sent from BN_n to CN_m:}

$$Lq_{nm}^{(i)} = LP_n + \sum_{m' \in M(n) \setminus m} Lr_{m'n}^{(i)}. \tag{10}$$

{3. Finally, we calculate the *a posteriori* pseudo-probabilities and perform hard decoding: }

$$LQ_n^{(i)} = LP_n + \sum_{m' \in M(n)} Lr_{m'n}^{(i)}. \tag{11}$$

4. The Cell/B.E. multicore architecture

The required computational power and the irregular memory access patterns for LDPC decoding define hard challenges that we propose to tackle for the heterogeneous Cell/B.E. multicore [Hofstee, 2005] shown in figure 3. The Cell/B.E. processor provides a set of cores that include one main 64-bit PowerPC Processor Element (PPE) and eight Synergistic Processor Elements (SPEs) under a vectorized 128-bit wide Single Instruction Multiple Data (SIMD) oriented architecture [Sony Computer Entertainment Incorporated, 2005]. Data transfers between the main memory and each SPE's local memory (256KByte) are performed by using efficient Direct Memory Access (DMA) mechanisms that offload the processors from the expensive task of moving data. Each SPE, by its turn, exploits quite efficiently a dual pipeline mechanism executing independently: one supports arithmetic operations; while the other performs load and store memory operations.

The memory in the CELL/B.E. is organized as a distributed memory system. Data is loaded from the main memory into the SPE's local storage and vice-versa, allowing each processor to exploit data locality individually. As in opposition to architectures based on shared memory models, here the programmer is free from having to deal with strategies to avoid memory access conflicts.

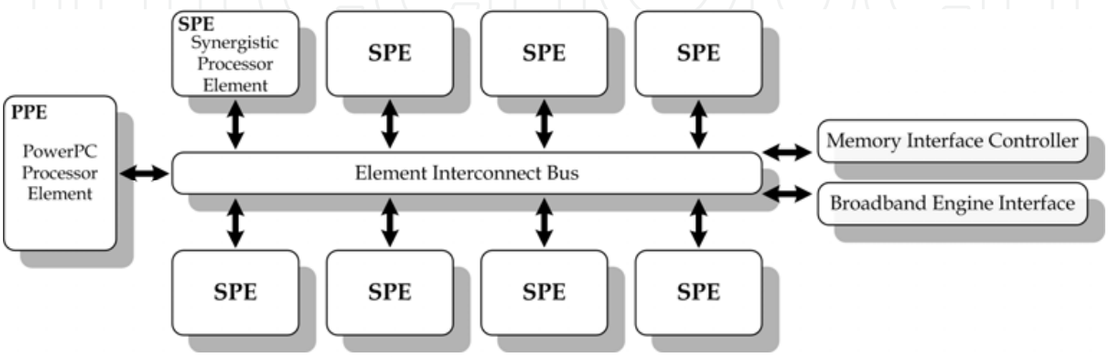


Fig. 3. Generic overview of the Cell/B.E. architecture.

5. The parallel programming model and parallel algorithms for LDPC decoding on multicores

The Single Program Multiple Data (SPMD) and the SIMD programming models are adopted to exploit data parallelism and to develop the parallel methods and algorithms for LDPC decoding on the Cell/B.E. A vectorized SIMD-based multicore approach that exploits data locality and fast data-block transfers associated to a powerful dual pipeline mechanism, allowed us to efficiently implement the concept of simultaneous multicodeword LDPC decoding on the Cell/B.E. Each SPE decodes several complete codewords and a total of 24 to 96 codewords, depending if we use 32- or 8-bit data precision, are decoded in parallel and at the same time in all the 6 SPEs available on the PlayStation 3 platform used in this work.

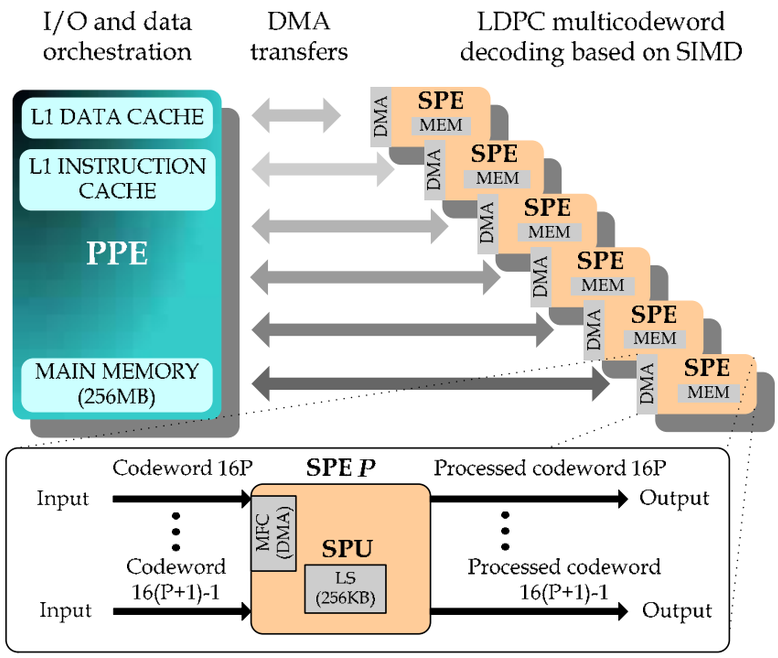


Fig. 4. Parallelization model for an LDPC decoder on the Cell/B.E. architecture.

The parallel LDPC decoder exploits SIMD data-parallelism by applying the same algorithm to different codewords on each SPE. As the Tanner graph is common to all codewords under decoding, these data structures can be shared allowing multicodeword decoding simultaneously in all SPEs. These features are illustrated in figure 4.

The irregular memory access patterns common in LDPC decoding represent a challenge to the efficiency of the algorithm as illustrated in figure 5 for the example shown in figure 2. The access to different nodes in the Tanner graph is defined by the **H** matrix and should favor randomization in order to allow good coding gains. For that reason, the data structures developed and represented in figure 6 try to minimize that effect, by grouping contiguously in memory associated data computed in the same kernel. A global irregular access pattern is translated into several partial regular access patterns. Moreover, only non-null elements of the **H** matrix are stored which represents savings in terms of memory usage.

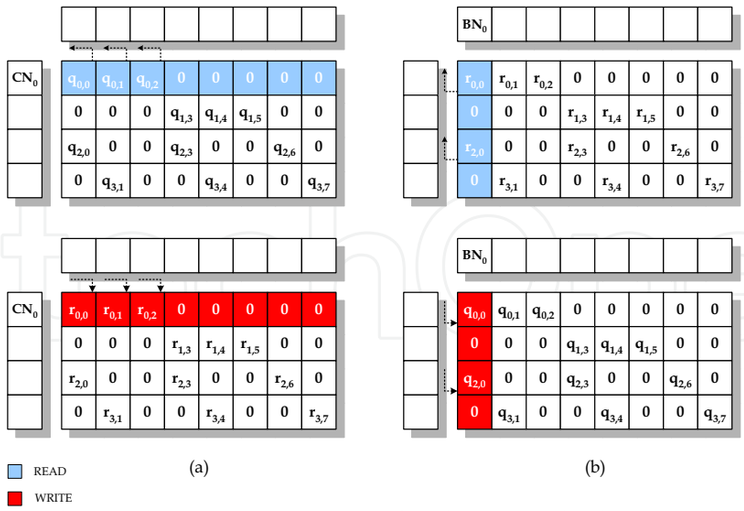


Fig. 5. Illustration of irregular memory accesses for the example shown in figure 2.

5.1 The parallel algorithm on the Cell/B.E.

The parallelization approach proposed for developing an LDPC decoder is explained in the context of the Cell/B.E. architecture. The data structures that define the Tanner graph and the program as well are loaded into the local storage on the SPEs where the processing is performed. The LDPC decoder processes on an iterative basis. The PPE reads information y_n from the input channel and produces the probabilities p_n as indicated in (2). The PPE controls the main tasks, offloading the intensive processing to the SPEs, where the processing is then distributed over several threads. Each SPE runs independently of the other SPEs. After receiving the p_n values associated to the corresponding codewords, each SPE performs two steps: *(i)* computes kernel 1 and kernel 2 alternately using SIMD instructions; and *(ii)* sends the final results back to the PPE, which concludes the computation of the current codewords and starts new ones by replacing data to be sent to the SPEs.

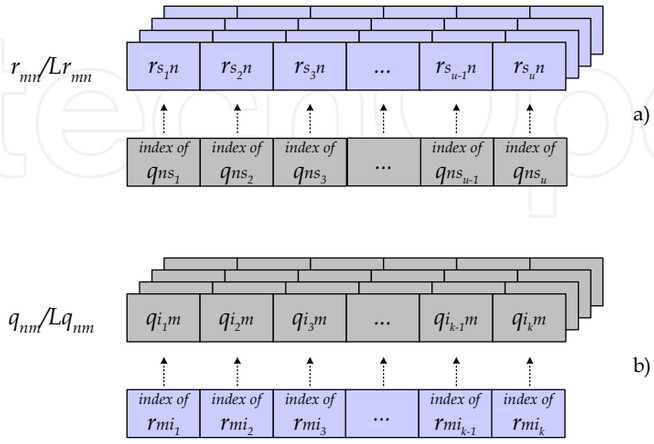


Fig. 6. Segment of the SIMD vectorized data structures in memory to represent: a) r_{mn}/Lr_{mn} messages associated to BNs; and b) q_{nm}/Lq_{nm} messages associated to CNs.

The parallel LDPC decoder explores data-parallelism by applying the same algorithm to several codewords simultaneously on each SPE (as shown in figure 4). Data is represented as 32-bit precision floating-point or 8-bit integer elements, depending on the algorithm used – SPA or MSA. The proposed LDPC decoder suits scalability and for that reason it can be easily adopted by future generations of the architecture with a higher number of SPEs. In that case, it will be able of decoding more codewords simultaneously, increasing the efficiency and aggregate throughput of the decoder.

Processing a complete iteration inside the SPE is performed in two phases: *(i)* kernel 1 computes data according to (3) and (4) for the SPA or (9) for the MSA, where the horizontal processing (row-major order) is performed. The data structure designed to represent r_{mn}/Lr_{mn} in order to perform this task is depicted in figure 6 a). It can be seen in this figure that data related to BNs common to a CN equation is stored in contiguous memory positions to optimize processing; and *(ii)* kernel 2 processes data according to (5) and (6) for the SPA or (10) for the MSA, performing the vertical processing (column-major order). The q_{nm}/Lq_{nm} data structure used in this case is depicted in figure 6 b). The SPE accesses data in a row- or column-major order, depending on the kernel that is being processed at the time. Also, the parallel algorithm implemented exploits the double buffering technique by overlapping processing and data accesses in memory.

Kernel 1 performs the horizontal processing according to the Tanner graph, and the $r_{mn}^{(i)} / Lr_{mn}^{(i)}$ data is updated for iteration *i*. The data is initially transferred to the local storage of the SPE by performing a DMA transaction, and its access organization maximizes data reuse, because a CN updating BNs reads common information from several BNs that share data among them. Figure 6 b) shows the data structures that hold the q_{nm}/Lq_{nm} values to be read and also the corresponding indexes of the r_{mn}/Lr_{mn} elements in figure 6 a) that they are going to update. As depicted in figure 6, BNs and CNs associated with the same r_{mn}/Lr_{mn} or q_{nm}/Lq_{nm} equation are represented in contiguous blocks of memory.

In kernel 2 data is processed in a column-major order. According to the Tanner graph, each BN updates all the CNs connected to it and holds the addresses necessary to complete the update of all $q_{nm}^{(i)} / Lq_{nm}^{(i)}$ data for iteration *i*. Once again, maximum data reuse is achieved, but this time among data belonging to the same column of the **H** matrix, as depicted in figures 2 and 6 b).

The computation is performed in the SPE for a predefined number of iterations. One of the purposes of this work is to assess the performance of the proposed solutions in terms of throughput. Pursuing this goal, we decided to develop a solution where the number of iterations is fixed to allow a fair comparison between different approaches, where the processing workload is known *a priori* and the same for all environments. This is why, at the end of an iteration, we don't check if the decoder produces a valid codeword, which could cause the decoding process to stop. Nevertheless, this operation represents a negligible overhead. The iterative updating mechanism applied to BNs and CNs is performed in a sequence of pairs (BN, CN) and tested for a number of iterations ranging from 10 to 100. When all the BNs and CNs are updated after the final iteration, the SPE activates a DMA transaction and sends data back to the main memory, signalizing the PPE to conclude the processing. As the DMA finishes transferring data, synchronization points are introduced to allow data buffers reuse.

Algorithm 1 PPE side of the algorithm

for $th\_ctr = 1$ *to* $NSPEs$: **do**
  Create th_ctrt *hread*
end for

repeat
  Receive y_n *from the channel and calculate* p_n *probabilities*
  Send msg NEW_WORD to MASTER SPE
Ensure : *Wait until mail is received* $(SPE[i].mailboxcount > 0)$ *from MASTER SPE*
  $msg = SPE[i].mailbox(received\ msg\ END\_DECODE\ from\ MASTER\ SPE)$
until *true*

Synchronization between the PPE and the SPEs is performed using mailboxes. This approach tries to exploit data-parallelism and data locality by performing the partitioning and mapping of the algorithm and data structures over the multiple cores, while at the same time minimizes delays caused by latency and synchronization.

5.2 Processing on the PPE

The part of the algorithm that executes on the PPE side is presented in Algorithm 1. We force the PPE to communicate with only one SPE, called MASTER SPE, which performs the control over the remaining SPEs. This is more efficient than putting the PPE controlling all the SPEs.

Algorithm 2 MASTER SPE side of the algorithm

repeat
Ensure : *Read mailbox* $(waiting$ a $NEW\_WORD\ mail\ from\ PPE)$
  Broadcast msg NEW_WORD to all other SPEs
  Get p_n *probabilities*
  for $i = 1$ *to* N_Iter : **do**
    Compute r_{mn} / Lr_{mn}
    Compute q_{nm} / Lq_{nm}
  end for
  Put final Q_n *values on the PPE*
Ensure : *Read mailbox* $(waiting$ an $END\_DECODE\ mail\ from\ all\ other\ SPEs)$
  Send msg END_DECODE to PPE
until *true*

The PPE receives the y_n information from the channel and calculates probabilities p_n, after which it sends a *NEW_WORD* message to the MASTER SPE. Then, it waits for the download of all p_n probabilities to the SPEs and for the processing to be completed in each one of them.

Algorithm 3 SLAVE SPE side of the algorithm

repeat
Ensure : *Read mailbox* $\left(waiting \text{ a } NEW\_WORD \text{ mail from MASTER SPE}\right)$
Get p_n *probabilities*
for $i = 1$ *to* N_Iter : **do**
Compute r_{mn} / Lr_{mn}
Compute q_{nm} / Lq_{nm}
end for
Put final Q_n *values on the PPE*
Send msg END_DECODE to MASTER SPE
until *true*

Finally, when all the iterations are completed, the MASTER SPE sends an *END_DECODE* message to the PPE to conclude the current decoding process and get ready to start processing a new set of codewords.

5.3 Processing on the SPE

The SPEs are used in the intensive task of updating all BNs and CNs by executing kernels 1 and 2 (either for the SPA or for the MSA), in each decoding iteration. Each thread running on the SPEs accesses data in the main memory by using DMA and computes data according to the Tanner graph, as defined in the **H** matrix (figure 2). The MASTER SPE side of the procedure is described in Algorithm 2. The *Get* operation is adopted to represent a communication PPE → SPE, while the *Put* operation is used for communications in the opposite direction.

We initialize the process and start an infinite loop, waiting for communications to arrive from the PPE (in the case of the MASTER SPE), or from the MASTER SPE (for all remaining SPEs). In the MASTER SPE, the only kind of message expected from the PPE is a *NEW_WORD* message. When a *NEW_WORD* message is received, the MASTER SPE broadcasts a *NEW_WORD* message to all other SPEs and loads corresponding p_n probabilities. After receiving these messages, each one of the other SPEs gets its own p_n values.

The processing starts and terminates when the number of iterations is reached and an *END_DECODE* mail is sent by all SPEs to the MASTER SPE, which immediately notifies the PPE with an *END_DECODE* message, as described in Algorithms 2 and 3.

The intensive part of the computation in LDPC decoding on the Cell/B.E. architecture takes advantage of the processing power and SIMD instruction set available on the SPEs, which means that several codewords are decoded in parallel.

6. Experimental evaluation

To evaluate the performance of the proposed LDPC decoder, the Cell/B.E. was programmed using: *(i)* the PPE alone which is denoted by serial mode in figure 7; and *(ii)* the complete set of PPE and 6 SPE processors denoted by parallel mode in the same figure.

The Cell/B.E. under test is included in a PlayStation 3 (PS3) platform, which restricts the number of available SPEs to 6, from a total of 8. The experimental setup of the PS3 platform is presented in table 2.

	Serial mode	Parallel mode	
Platform	PPE	STI Cell/B.E.	
Language	C	C	
OS	Linux (Fedora) kernel 2.6.16		
		PPE	SPE
Clock frequency	3.2GHz	3.2GHz	3.2GHz
Memory	256MB	256MB	256KB

Table 2. Experimental setup.

6.1 Serial versus parallel execution modes

The serial mode depicted in figure 7 uses a dual thread approach and exploits SIMD instructions. It should be noted that by performing the comparison based on the time per bit decoded, the serial solution that uses only the PPE is slower than the execution on a single SPE, because the PPE accesses slow main memory, while the SPE accesses faster local storage memory.

Code (n, k)	Edges	Occupancy of data structures on the local storage of a SPE (Bytes)
(504, 252)	1512	70560
(1024, 512)	3072	143360

Table 3. Size of data structures used in each SPE in parallel decoding mode.

On the parallel approach the experimental results were also obtained using SIMD instructions on the SPEs, which are responsible for executing the intensive decoding part of the algorithm. In this case the PPE orchestrates the execution on the SPEs as explained before, while inside each SPE several codewords are being simultaneously decoded in parallel. All the processing times were measured for a number of iterations ranging from 10 to 100. In literature, the average number of iterations considered for WiMAX LDPC decoders to work under realistic conditions is typically below 20 iterations [Brack et al., 2006; Seo et al., 2007; Liu et al., 2008].

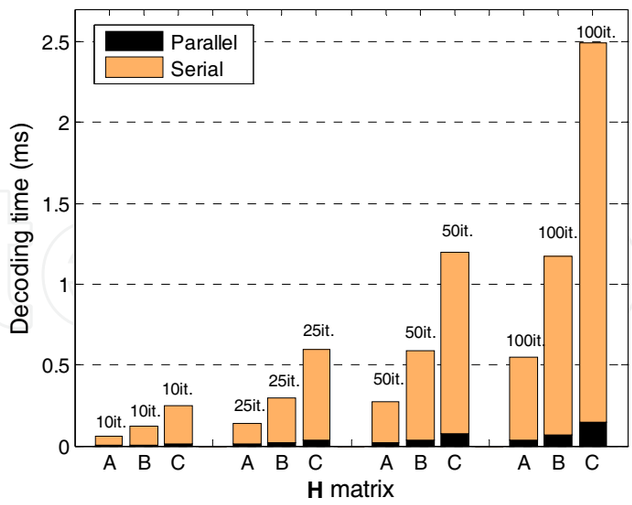


Fig. 7. Comparing LDPC decoding times for serial and parallel modes, where codes A, B and C represent, respectively, matrices (256, 128), (504, 252) and (1024, 512).

Table 3 shows the dimensions of two regular LDPC codes with *rate*=1/2, the corresponding number of edges and the size of data structures used to represent the Tanner graph. The local memory of the SPE is limited to 256 KByte. It should be taken in consideration that the SPE's local memory should hold both data structures and also the program.

6.2 LDPC decoding using the SPA

The first parallel approach mentioned in 6.1 uses the SPA. Data elements have 32-bit floating-point precision, and 4 floating-point elements are packed and operated on a single instruction, making it possible to decode 4 codewords in parallel on each SPE. Then, with 6 SPEs available, the global architecture can decode 24 codewords in simultaneous. We assessed the results for regular codes, which typically execute faster than irregular ones. The average throughput obtained is presented in table 4, and it ranges from 69.1 to 69.5 Mbps, when decoding regular codes (504, 252) and (1024, 512) in 10 iterations. It should be noticed that the decoding time per bit and per iteration remains approximately constant. Although real-life performances demand throughputs which can be typically in the order of 40 Mbps per channel, the theoretical maximum required by the WiMAX standard can go up to approximately 75 Mbps per channel. The throughputs reported in table 4 are inferior to 70 Mbps and do not guarantee such requirements. Also, adapting the algorithm to support the necessary irregular codes used in the WiMAX would produce even worst results, because in that case accesses to memory depend on a variable number of edges per row/column. Therefore, optimizing the LDPC decoding algorithm to make it execute in a shorter period of time became mandatory. One possible solution consisted of exploiting the computationally less demanding MSA described in section 3.2.

Code (n, k)	*rate*	10 iter.	25 iter.	50 iter.
(504, 252)	1/2	69.1	28.3	14.2
(1024, 512)	1/2	69.5	28.4	14.3

Table 4. Throughput (Mbps) obtained in the parallel mode for the SPA.

6.3 LDPC decoding using the MSA

To increase the efficiency of the LDPC decoder we implemented the MSA on the Cell/B.E. It requires less computation, based essentially in addition and comparison operations [Falcão et al., 2009b]. Additionally, we also adopted the Forward-and-Backward simplification of the algorithm [Mackay, 1999] that avoids redundant computation and eliminates repeated accesses to memory. In the MSA data elements have 8-bit integer precision, which allows packing 16 data elements per 128-bit memory access. This increases the arithmetic intensity of the algorithm, here defined as the number of arithmetic operations per memory access, which favors the global performance of the LDPC decoder. The instruction set of the Cell/B.E. architecture supports intrinsic instructions to deal efficiently with these parallel 128-bit data types. Moreover, because there are 6 SPEs available, the algorithm now supports the simultaneous decoding of 96 codewords in parallel. However, the set of 8-bit integer intrinsic parallel instructions of the Cell/B.E. is more limited than those of the 32-bit floating-point family of instructions. This explains that the speedup obtained when changing from the SPA to the MSA is lower than we would expect. Table 5 shows the throughputs obtained for some example codes used in the WiMAX IEEE 802.16e standard. For 10 iterations, in some cases they approach quite well while in others they even surpass the 75 Mbps required by the standard to work in (theoretical) worst case conditions.

Code (n, k)	*rate*	10 iter.	25 iter.	50 iter.
(576, 288)	1/2	79.8	32.7	16.5
(576, 432)	3/4	73.1	29.9	15.1
(576, 480)	5/6	79.3	32.5	16.4
(672, 448)	2/3	74.8	30.6	15.4
(672, 504)	3/4	72.6	29.7	15.0
(672, 560)	5/6	78.5	32.2	16.2
(960, 480)	1/2	79.6	32.6	16.4
(960, 640)	2/3	74.7	30.6	15.4
(960, 720)	3/4	72.6	29.7	15.0
(960, 800)	5/6	78.4	32.1	16.2
(1152, 576)	1/2	79.6	32.6	16.4
(1152, 768)	2/3	74.6	30.5	15.4
(1152, 864)	3/4	72.6	29.7	15.0
(1152, 960)	5/6	78.4	32.1	16.2
(1248, 624)	1/2	79.6	32.6	16.4
(1248, 832)	2/3	78.5	32.2	16.2
(1248, 936)	3/4	72.7	29.7	15.0
(1248, 1040)	5/6	78.4	32.1	16.2

Table 5. Throughput (Mbps) obtained in the parallel mode for the MSA.

For codes with n=576 running 10 iterations, it can be seen that throughputs range from 73.1 to 79.8 Mbps. For codes with n=1248 and for the same number of iterations, they vary from 72.7 to 79.6 Mbps. All codes in table 5 were tested for the MSA and approach quite well the maximum theoretical limit of 75 Mbps. They all show better performances than those obtained with the SPA. Furthermore, if we consider a lower number of iterations, the

decoder's throughput may rise significantly. For example, if we consider an LDPC decoder running 5 iterations, the throughput will approximately double to values above 145Mbps.

6.4 Discussion

In multicore architectures, efficient parallel programming, both in terms of computation and memory accesses, represent a significant challenge. The Cell/B.E. is based on a distributed memory model where the problem of data collisions can decrease when properly handled by the programmer. The reported experimental results allow assessing the performance of LDPC decoders based on multicores. We have shown that for LDPC decoders running the SPA on the Cell/B.E., throughputs can range from 68 to nearly 70 Mbps [Falcão et al., 2009a]. Concerning the MSA, a more efficient solution is achieved producing throughputs that range from 72 to 80 Mbps [Falcão et al., 2008]. Regarding to non-scalable hardware dedicated ASIC solutions, which typically adopt 5 to 6-bit precision arithmetic [Liu, 2008], the parallel programmable architecture here proposed allows using 8-bit data precision or even more, which produces lower Bit Error Rates (BER) and superior coding gains as depicted in figure 8. The adoption of specific parallelization techniques on a low-cost multicore platform allowed us to achieve throughputs that approach well those obtained with ASIC-based solutions for WiMAX [Brack, 2006; Seo, 2007; Liu, 2008;]. They also guarantee enough bandwidth for LDPC codes used in the WiMAX standard to work in worst case conditions.

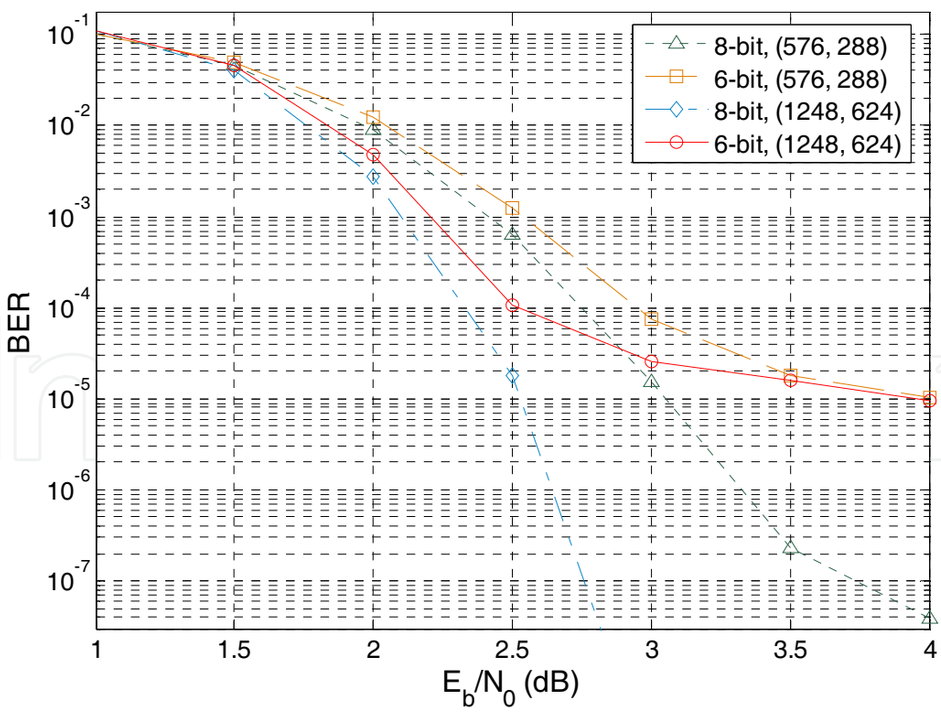


Fig. 8. BER curves for WiMAX codes (576, 288) and (1248, 624), considering both 6- and 8-bit data precision representations.

7. Conclusions

The advent of inexpensive multicore architectures has allowed to develop a novel programmable LDPC decoding solution for the WiMAX standard, with excellent throughputs, on the Cell/B.E. architecture. The LDPC decoder here presented exploits parallelism and data locality and is scalable to future generations of the Cell/B.E. architecture that are expected to have more SPEs, and should therefore improve the performance even further, processing more channels/subcarriers per second. The proposed decoder compares well with non-scalable and hardware-dedicated typical ASIC LDPC decoding solutions, reporting superior BER performances and throughputs above 72 Mbps. On going additional work related with LDPC decoders running on alternative parallel architectures can be found in [Falcão et al., 2009b; Seo et al., 2007].

8. References

Brack, T.; Alles, M.; Kienle, F. & Wehn, N. (2006). A Synthesizable IP Core for WIMAX 802.16E LDPC Code Decoding, *Proceedings of the IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications,* pp. 1—5, September 2006.

Falcão, G.; Silva, V.; Sousa, L. & Marinho, J. (2008). High coded data rate and multicodeword WiMAX LDPC decoding on the Cell/BE. *IET Electronics Letters*, Vol. 44, No. 24, November 2008, pp. 1415–1417.

Falcão, G.; Sousa, L.; Silva, V. & Marinho, J. (2009a). Parallel LDPC Decoding on the Cell/B.E. Processor, *Proceedings of the 4th International Conference on High Performance and Embedded Architectures and Compilers (HiPEAC 2009)*, pp. 389–403, Paphos, Cyprus, January 2009, Lecture Notes in Computer Science, Springer, Vol. 5409.

Falcão, G.; Silva, V. & Sousa, L. (2009b). How GPUs Can Outperform ASICs for Fast LDPC Decoding, *Proceedings of the 23rd International Conference on Supercomputing (ICS)*, New York, USA, June 2009, pp. 390–399

Gallager, R. (1962). Low-Density Parity-Check Codes. *IRE Transactions on Information Theory*, Vol. 8, No. 1, January 1962, pp. 21–28.

Guilloud, F.; Boutillon, E. & Danger, J.-L. (2003). λ-min decoding algorithm of regular and irregular LDPC codes. *Proceeding of the 3rd Int. Symp. Turbo Codes Relat. Topics*, pp. 1–4, September 2003.

Hofstee, H. (2005). Power Efficient Processor Architecture and the Cell Processor, *Proceedings of the 11th International Symposium on High-Performance Computer Architectures (HPCA)*, pp. 258–262, San Francisco, CA, USA, February 2005.

IEEE P802.16e/D12 (2005). Draft IEEE Standard for Local and Metropolitan Area Networks. Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems, October 2005.

International Business Machines Corporation (2007). *Synergistic Processor Unit Instruction Set Architecture*, IBM Systems and Technology Group.

Lin, S. & Costello, D. (2004). *Error Control Coding*, Prentice Hall, second edition, ISBN-13: 978-0130426727.

Liu, C.-H.; Yen, S.-W.; Chen, C.-L.; Chang, H.-C.; Lee, C.-Y.; Hsu, Y.-S. & Jou, S.-J. (2008). An LDPC Decoder Chip Based on Self-Routing Network for IEEE 802.16e Applications. *IEEE Journal of Solid-State Circuits*, Vol. 43, No. 3, March 2008, pp. 684–694.

Mackay, D. & Neal, R. (1996). Near Shannon Limit Performance of Low Density Parity Check Codes. *IEE Electronics Letters*, Vol. 32, No. 18, August 1996, pp. 1645–1646.

Mackay, D. (1999). Good Error-Correcting Codes Based on Very Sparse Matrices. *IEEE Transactions on Information Theory*, Vol. 45, No. 2, March 1999, pp. 399–431.

Seo, S.; Mudge, T.; Zhu, Y. & Chakrabarti, C. (2007). Design and Analysis of LDPC Decoders for Software Defined Radio, *Proceedings of the IEEE Workshop on Signal Processing Systems,* pp. 210–215, October 2007.

Sony Computer Entertainment Incorporated (2005). *SPU C/C++ Language Extensions*, Sony Corporation.

Tanner, R. (1981). A Recursive Approach to Low Complexity Codes. *IEEE Transactions on Information Theory*, Vol. 27, No. 5, September 1981, pp. 533–547.

WIMAX New Developments

Edited by Upena D Dalal and Y P Kosta

ISBN 978-953-7619-53-4

Hard cover, 442 pages

Publisher InTech

Published online 01, December, 2009

Published in print edition December, 2009

WiMAX (Worldwide Interoperability for Microwave Access) is a wireless broadband access network named by industry group called the WiMAX forum formed in June 2001. It is Wireless MAN with IEEE 802.16 family standards. Loosely, WiMAX is a standardized wireless version of Ethernet that enables the last mile, intended primarily as an alternative to wire technologies (such as Cable Modems, DSL and T1/E1 links) to provide broadband access to customer premises. Mission of the WiMAX forum is to promote and certify compatibility and interoperability of broadband wireless products. This book touches most of the above issues in form of 22 individuals' papers containing research work in WiMAX domain in particular. WiMAX has two important standards/usage models: a fixed usage model IEEE 802.16-2004 for Fixed Wireless Broadband Access (FWBA) and a portable usage model IEEE 802.16e-2005, which is mainly concentrated on Mobile Wireless Broadband Access (MWBA). Both are released as standards and amendments are available in form of drafts. Higher data rate transmissions (@ 100 Mbps) are achieved in IEEE 802.16-2004 WiMAX through LOS communications which incorporate a stationary transmitter and receiver but IEEE 802.16e supporting NLOS communication is much complicated and little less bit rate is achieved. 2-11 GHz licensed band is the range of frequencies with TDD and FDD supports. The book will provide a wide horizon to visualize the WiMAX technology and its developments leading towards 4G systems. It will provide a good platform to the researchers with clues to the innovative ideas in WiMAX domain. I wish all the best to the authors and readers of this book in their successful research of WiMAX technology.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

www.intechopen.com

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen

IntechOpen