A Novel Parallel Viterbi Decoding Scheme for NoC-Based Software-Defined Radio System

Jian Wang, Yubai Li, and Huan Li

In this paper, a novel parallel Viterbi decoding scheme is proposed to decrease the decoding latency and power consumption for the software-defined radio (SDR) system. It implements a divide-and-conquer approach by first dividing a block into a series of subblocks, then performing independent Viterbi decoding for each subsequence, and finally merging the surviving subpaths into the final path. Moreover, a network-on-chip-based SDR platform is used to evaluate the performance of the proposed parallel Viterbi decoding scheme. The experiment results show that our scheme can speed up the Viterbi decoding process without increasing the BER, and it performs better than the current state-of-the-art methods.

Keywords: Network on chip, Viterbi decoding, parallel processing, software-defined radio, divide-and-conquer.

I. Introduction

To support a high data rate and a long transmission distance, many wireless communication standards, such as Wideband Code Division Multiple Access, Wi-Fi, and long-term evolution (LTE), have been proposed and implemented in application-specific integrated circuits (ASICs). However, this kind of platform cannot adapt to multiple standards with reconfigurability for different kinds of demands in the future. Therefore, the term software-defined radio (SDR) [1], which can meet the current and future standards by reconfigurating, is introduced into the wireless communication system.

On the other hand, the Viterbi decoder, which plays an important role in many wireless communication standards and whose computational complexity is quite significant, becomes one of the major factors determining the overall throughput of the system (for example, based on our complexity analysis, the Viterbi decoder block consumes the largest computation resource on the software modem out of all the blocks of mobile WiMAX [2]). Therefore, the parallel Viterbi decoding methods, which decrease the decoding latency, have been widely studied by many researchers [3]-[5]. In this paper, we propose a novel parallel Viterbi decoding scheme for the SDR system, which speeds up the Viterbi decoding process without increasing the BER and performs better than the state-of-the-art methods.

The rest of this paper is organized as follows. In section II, we briefly comment on the related works. The network on chip (NoC)-based SDR platform is presented in section III, and our proposed parallel Viterbi decoding scheme is introduced in section IV. In section V, experiment results are given to show the effectiveness and benefit of our method. Finally, we summarize our contribution in this paper in section VI.

Manuscript received Jan 9, 2013; revised Apr. 2, 2013; accepted Apr. 13, 2013.

This work was supported by the National Natural Science Foundation of China [61201005], the Fundamental Research Funds for the Central Universities [ZYGX2012J001], and the National Major Projects [2011ZX03003-003-04].

Jian Wang (phone: +86 28 61831246, wangjian3630@uestc.edu.cn), Yubai Li (ybli@uestc.edu.cn), and Huan Li (photon_lee@126.com) are with DSP Lab, School of Communication and Information Engineering, University of Electronic Science and Technology of China, Chengdu, Sichuan, China.

http://dx.doi.org/10.4218/etrij.13.0113.0032

 © 2013 ETRI

II. Related Work

To meet the throughput requirement in high data rate systems, a parallel decoding algorithm that can reduce decoding latency is desirable. There are several parallel approaches for the traditional Viterbi Algorithm (VA), which can be categorized into three levels [6]: the bit level [7], the word level [8], and the algorithm level [9]. In the bit-level and the word-level parallel Viterbi decoders, researchers concentrate on the specific component, which makes the two kinds of parallel decoder more suitable for ASICs (for example, bit-level techniques optimize the add-compare-select [ACS] operations in path metrics computation and decision process, while word-level optimizations work on unfolding the ACS recursion loop, such that two or more trellis stages are processed in a single recursion). In the algorithm-level parallel Viterbi decoder, the Viterbi decoding is converted to a purely feedforward computation by algorithmic transformations, and this allows independent processing of input blocks or states (for example, one method is to divide a coded word into a series of blocks and these blocks are simultaneously decoded. Another method is to schedule the trellis states onto different processors, such that in a trellis stage, a processor processes only one or several states but not all states). Obviously, the algorithm-level parallel Viterbi decoder is more suitable for software implementations on multiprocessors and thus more suitable for SDR systems.

Lai and others realized a Viterbi decoder on the CELL SDR platform [10]. Since the CELL processor is single-instruction multiple-data (SIMD) architecture with two pipelines, the authors packed the scalar data into a vector, decreased the dependency among the nearby instructions, and put instructions to two pipelines together to speed up the Viterbi decoding process. The authors also pointed out the possibility that distributing the decoding task of one data frame into different processors could further decrease the decoding latency. However, the actual implementation of this thought was not given in their paper.

Kim and others realized a parallel Viterbi decoder on a graphics processing unit (GPU)-based SDR platform [11]. In [11], many threads are generated by the GPU and the path candidates for each stage are calculated at the same time in parallel. More precisely, the states in a stage are divided into 2^m groups, and each group is processed by a thread in the GPU. Note that m denotes the number of registers in the convolution coder. The theoretical speed up of Kim's method is 2^m times compared to the classic method; however, the Viterbi decoder on GPU-based SDR cannot achieve its theoretical performance. The reason is that the parallel decoding latency in a multiprocessor platform is affected by two parts: the processing time consumed by processors and the communication time used to exchange the data among processors. When the method in [11] is applied, a lot of time is spent on the data communication, such that the Viterbi decoder on GPU-based SDR cannot speed up 2^m times.

Lin and others proposed a tiling-scheme Viterbi decoder for the GPU-based SDR platform [12]. In [12], the best path was selected by calculating Hamming distances trellis by trellis in parallel. More precisely, the decoding data are divided into a series of chunks, and each chunk is separately decoded. Then, the survival paths in each chunk are merged into the best path. Since the required communication time in this paper is much less than the communication time in [11], this method outperforms Kim's method up to 2.5 times. However, since there are no protecting stages for each chunk, the first and last few stages in a chunk cannot be protected and thus may cause a decoding error. Although it may not be a serious problem for the long convolution code under the high signal-to-noise ratio (SNR) channel environment, it decreases the decoding performance when the short convolution code travels on the low SNR channel.

III. NoC-Based SDR Platform

Besides the GPU-based SDR platform [13], [14], the NoC-based SDR platform is widely used for data-intensive computing [15], [16]. In this paper, the NoC-based SDR platform is selected for our parallel decoding. In the platform, OR1200 cores are used as the process element (PE) and a mesh NoC is used to connect them together [17], as shown in Fig. 1.

In Fig. 1, DC is the data cache and IC is the instruction cache. RAM is used to store the processing data, such that the system is applicable to data-intensive computing. NI is the internal network interface, which is used to connect a PE to an NoC router (in Fig. 1, the capitalized letter R is used to represent an NoC router). When there is data exchanged among PEs, dimension-order routing without deadlock (for example, the XY routing algorithm [18]) is used to deliver the data across the NoC. Moreover, when different data flows contending for the same output channel in the input ports of a router, a lottery schedule mechanism, which authorizes the permission for each to flow with the same probability, is used in the arbiter module. Additionally, a wormhole mechanism, in which data is divided into a sequence of flits and each input port of all routers has a buffer with just one flit size, and a hop-to-hop credit mechanism are used to ensure that a flit is transmitted only when the receiving buffer has free space.

As shown in Fig. 1, a CPU is used to run the main function, and the function data is transferred to the NoC-based SDR

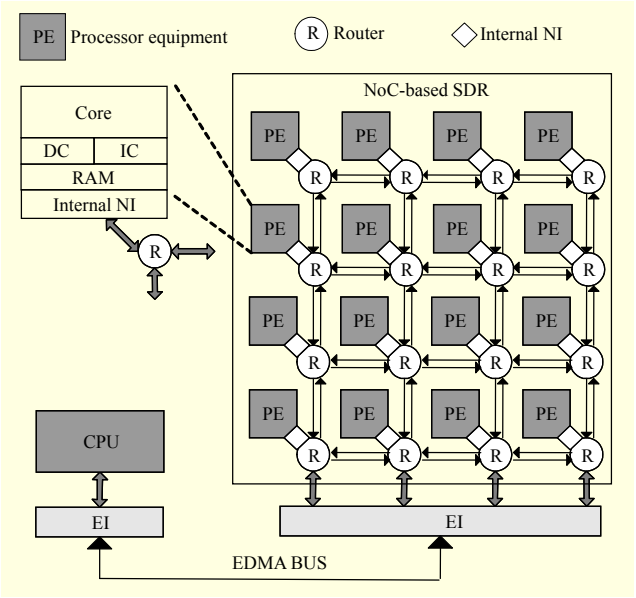

Fig. 1. NoC-based SDR platform.

platform through the external interface (EI) and bus when the program comes across massive computational functions (for example, Viterbi decoding in this paper). Then, the NoC-based SDR platform processes the function data by using the predetermined parallel algorithm. Once the NoC-based SDR platform runs to the end of the parallel algorithm, the function result is returned to the CPU.

IV. Parallel Viterbi Decoding Scheme

In this section, we propose a novel parallel Viterbi decoding scheme. Two aspects are taken into consideration in developing this decoding method. First, the BER of our parallel decoding method should be on par with the BERs of some of the well-known decoding algorithms (such as WAVA [19]), and it should not decrease with the increase of the number of processors (otherwise, the scalability of our method is not suitable). Second, regarding the decoding latency and power consumption, our method should perform better than the state-of-the-art parallel decoding method.

For convenience, some notations used throughout this paper are defined in Table 1. With these notations, the problem of the parallel Viterbi decoding scheme for the SDR platform can be formulated as follows: given a convolution code C and the SDR architecture A, find a parallel decoding scheme S, such that the decoding latency in the SDR platform is minimal without increasing the BER from that of the traditional decoding method.

The problem of minimizing the decoding latency can be formulated as follows:

Table 1. Parameter notation.

Notations	Description
A	architecture of SDR platform
C	convolution codes
S	parallel decoding scheme used on SDR platform
l	memory depth of convolution encoder
r	number of output bits in an encode stage; note that if one bit can generate r bits output, $1/r$ is the code rate
D_S	Viterbi decoding latency of scheme S on SDR platform
I_i	input data of i-th processor
L_C	length of C
L_i	length of I_i.
N_s	number of stages in a trellis
N_b	number of processors used in parallel decoding
E_R	system requirements for BER
P_R	system requirements for power consumption
$T_{p,q}$	$T_{p,q}=1$ represents existence of state transition from State p in previous stages to State q in current stages, and $T_{p,q}=0$, otherwise
$d_{p,q,I}$	Hamming distance between State p in previous stages to State q in current stages with a corresponding input I
$H^j_{q,h}$	Processor j's accumulated Hamming distance of State q in stages h
$H^j_{p,q,h}$	Processor j's accumulated Hamming distance of State q in stages h when the previous State in stage $(h-1)$ is p
$Q^j_{q,i}$	first state of the survival path corresponding to $H^j_{q,h}$

Algorithm 1. Divide data block.

1. Fetch received convolution code C
2. $L_i = \text{floor}(L_C/N_b)$
3. **while** $(L_i < 5rl)$ **do**
4. $\quad N_b = N_b - 1$;
5. $\quad L_i = \text{floor}(L_C/N_b)$;
6. **end while**
7. **for** $i = 1$ to $N_b - 1$ **do**
8. $\quad I_i = [C(L_i \times (i-1)+1)$ to $C(L_i \times i + r)]$
9. **end for**
10. $I_{N_b} = [C(L_i \times (N_b - 1)+1)$ to $C(L_B)$, $C(1)$ to $C(r)]$

$$\min(D_S), \quad \text{s.t. } E_R \text{ and } P_R. \tag{1}$$

To decode the data in parallel, a data block is divided into a series of subsets, as shown in Fig. 2. Then, these subsets are scheduled to different processors.

To protect the first and last stages of every subset, each subset has more than $5rl$ stages (line 2 through line 6 in

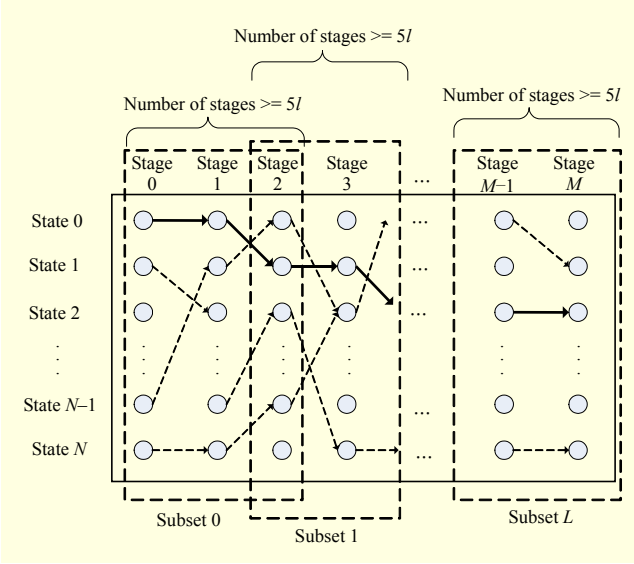

Fig. 2. Subsets with at least 5*l* stages.

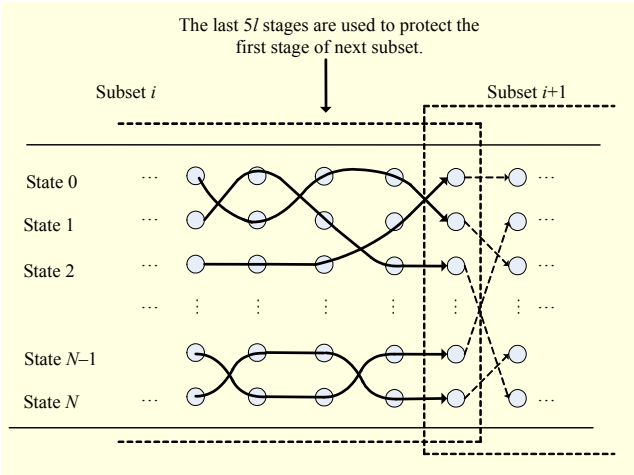

Fig. 3. Protecting first stage for each subset.

Algorithm 2. Parallel decoding on SDR platform on processor *j*.

1. Initial all $H_{q,h}^{j}$ to be zero, where *h* is the 5*l* stage from the end.
2. **for** $i = (h+1)$ to N_s **do**
3. Fetch input *I*
4. **for** $q = 1$ to 2^l **do**
5. **for** $p = 1$ to 2^l **do**
6. $H_{p,q,i}^{j} = H_{p,(i-1)}^{j} + T_{p,q}\, d_{p,q,I}$;
7. **end for**
8. $H_{q,i}^{j} = \min(H_{p,q,i}^{j})$;
9. **end for**
10. **end for**
11. Sent H_{q,N_S}^{j} to processor $(j+1)$
12. receive $H_{q,N_S}^{(j-1)}$ from the processor $(j-1)$
13. Initial $H_{q,1}^{j}$ by using the value of $H_{q,N_S}^{(j-1)}$
14. **for** $i = 2$ to N_s **do**
15. Fetch input *I*
16. **for** $q = 1$ to 2^l **do**
17. **for** $p = 1$ to 2^l **do**
18. $H_{p,q,i}^{j} = H_{p,(i-1)}^{j} + T_{p,q}\, d_{p,q,I}$;
19. **end for**
20. $H_{q,i}^{j} = \min(H_{p,q,i}^{j})$;
21. $Q_{q,i}^{j} = \text{State}(H_{q,i}^{j})$;
22. **end for**
23. **if** $i = 5l$ **then**
24. Send $Q_{q,5l}^{j}$ to processor $(j-1)$ and receive $Q_{q,5l}^{(j+1)}$
25. **end if**
26. **end for**

Algorithm 1) and there are overlapped *r* bits between two tandem subsets (line 7 through line 10 in Algorithm 1). Note that in most of the convolution encoders, *r* is much smaller than 5*l*.

Algorithm 2 describes the decoding progress in a processor; it contains two steps. In the first step (line 1 through line 11), the last 5*l* stages in subset *i* are preprocessed to get the minimal accumulated Hamming distance for each state of the last stage. Then, these minimal accumulated Hamming distances are transmitted to subset *i*+1 and used as the initial value of the corresponding state in the first stage of subset *i*+1. Therefore, the first stage of subset *i*+1 is protected, as shown in Fig. 3 (line 1 through line 13 in Algorithm 2).

In the second step of Algorithm 2 (line 13 through line 25), we decode the subset *i* by using the Viterbi algorithm. At the same time, when the first 5*l* stages are processed, a state in the first stage of subset *i*, which corresponds to the minimal accumulated Hamming distance, is found out and transmitted to subset *i*−1 to protect the last stage of subset *i*−1, as shown in Fig. 4.

Algorithm 3 is used to obtain the decoding results. It finds the surviving subpath for each processor and merges them into a surviving path for the whole code word. Finally, it outputs the decoding results based on the surviving path.

V. Experiment Results

To evaluate the performance of our parallel Viterbi decoding scheme, the convolution codes in the LTE specification, whose

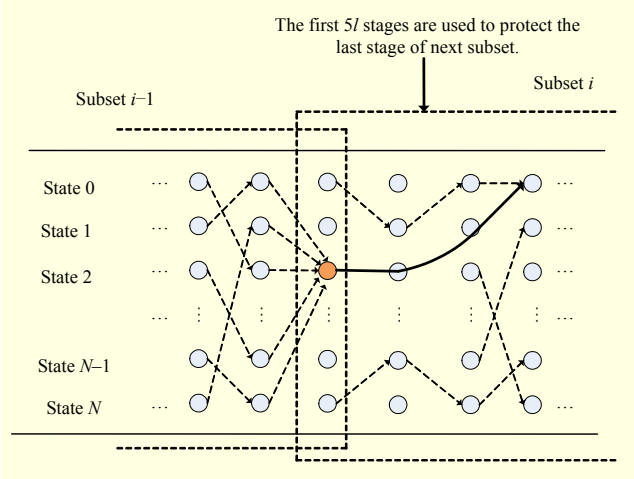

Fig. 4. Protected state in last stage of subset i–1.

Algorithm 3: Getting decoding results.

1. **for** $i = 1$ to N_b **do**
2. Find the minimal $H_{q,i}^{j}$, where $q = Q_{q,5l}^{(j+1)}$
3. Trace back from state $Q_{q,5l}^{(j+1)}$
4. Output the surviving subpath path$_i$ from stage 1 to stage N_s
5. **end for**
6. Merge the surviving subpath and output the decoding results

original code words have lengths ranging from 40 bits to 6,144 bits [20], are parallel decoded in the NoC-based SDR platform mentioned in section III. The coder has a memory depth of $l = 6$, the code rate is 1/3, and the generator polynomial is G = [133,171,165].

1. Bit Error Rate

In this subsection, we show the performance of the proposed decoding scheme for LTE convolution codes with binary phase-shift keying modulation and additive white Gaussian noise channels. A soft decision is used in the decoding process. At each SNR point, 200,000 code words are simulated to calculate the average BER. Figures 5 and 6 show a BER comparison for the short code (which contains 64 bits) and the long code (which contains 6,144 bits), respectively. Note that the short code being 64 bits and the long code being 6,144 bits means that the original uncoded data is 64 bits and 6,144 bits. After this data comes through the convolution coder with generator polynomial G = [133,171,165], the coder outputs 192 bits and 18,432 bits for the code rate of 1/3.

In the two figures, the red solid line, the green solid line, and the blue solid line respectively represent the decoding BER when using Lin's method on a 1-core, 16-core, and 64-core

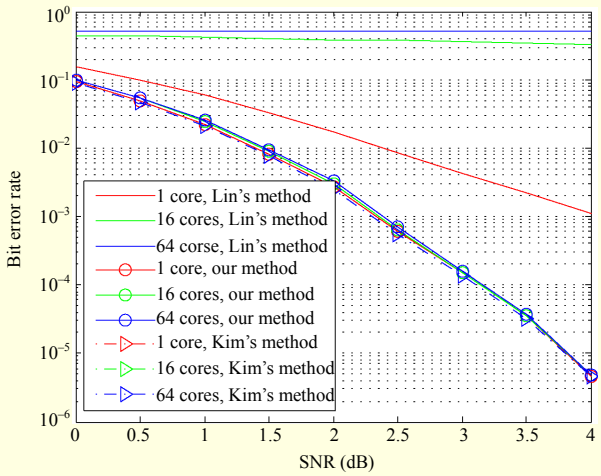

Fig. 5. Comparison of BER for short code.

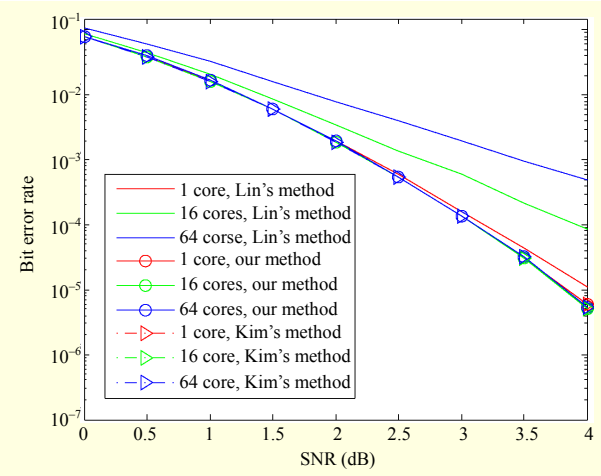

Fig. 6. Comparison of BER for long code.

SDR platform. The red solid line with a circle, the green solid line with a circle, and the blue solid line with a circle represent the BER when using our method on a 1-core, 16-core, and 64-core SDR platform, respectively. The red dashed line with a right-pointing triangle, the green dashed line with a right-pointing triangle, and the blue dashed line with a right-pointing triangle represent the BER when using Kim's method on a 1-core, 16-core, and 64-core SDR platform, respectively. Notice that in Kim's method, the WAVA algorithm is used and the states in a stage are processed in parallel on multicore platforms. When there is only one core in the platform, this method equals the standard WAVA algorithm.

Figures 5 and 6 show that the decoding BER of our method is almost the same as that of the WAVA method for both the short code and the long code, and it does not decrease as the number of processors increases. However, the decoding BER of Lin's method is larger than that of the WAVA method.

Table 2. Decoding latency comparison for short code (64 bits).

Method	Decoding latency (clks)			
	2 cores	4 cores	16 cores	64 cores
Kim's	1.68748×10^{7}	1.04974×10^{7}	8.76141×10^{6}	6.71025×10^{6}
Lin's	3.97432×10^{6}	3.14784×10^{6}	2.80282×10^{6}	2.67921×10^{6}
Ours	8.08580×10^{6}	8.08580×10^{6}	8.08580×10^{6}	8.08580×10^{6}

Table 3. Decoding latency comparison for long code (6,144 bits).

Method	Decoding latency (clks)			
	2 cores	4 cores	16 cores	64 cores
Kim's	3.49294×10^{8}	2.21458×10^{8}	1.78460×10^{8}	1.38720×10^{8}
Lin's	2.07192×10^{8}	1.08751×10^{8}	3.25150×10^{7}	1.43242×10^{7}
Ours	3.18625×10^{8}	1.64774×10^{8}	4.94149×10^{7}	2.13476×10^{7}

Moreover, the more processors we use to accelerate the decoding, the higher the BER of Lin's method is.

2. Decoding Latency Comparison

In this subsection, we compare the decoding latency of Kim's method and Lin's method to that of our method since the decoding latency is an important guideline for parallel decoding algorithms. Tables 2 and 3 show the number of clocks used in decoding for the long code word and the short code word, respectively.

According to Tables 2 and 3, Lin's method is the best one if the decoding latency is the only metric. The reason for this phenomenon is that the decoding latency is composed of the processing time and the communication time, and both elements of Lin's method are smaller or equal to that of Kim's method and our method. More precisely, in Kim's method, a processor must exchange data with all other processors at every stage of the trellis, but, in Lin's method, a processor only needs to exchange data with another processor at the last stage of a subset. Therefore, the communication time of Kim's method is much longer than that of Lin's method, such that the total decoding latency of Kim's method is longer than the decoding latency of Lin's method. In our method, to decrease the BER in the decoding results, the last $5l$ stages of subset i are preprocessed to protect the first stage of subset i+1, and the accumulated Hamming distance of the first $5l$ stages in subset i is transmitted to subset $i-1$ such that the last stage of subset $i-1$ is protected. Therefore, the time required for both processing and communication in our method is more than that needed in Lin's method. However, although Lin's method is superior regarding decoding latency, Kim's method and our method have better decoding performances because Lin's method has the highest BER, as shown in Fig. 5.

Another interesting result shown in Table 2 is that the decoding latency of our method does not change as the number of processors increases. The reason for this phenomenon is that for the 64-bit short code, the maximal number of processors used for parallel decoding is two since the length of L_i should be equal to or larger than $5rl$, as mentioned in section III. However, although the decoding latency of our method does not decrease as the number of processors increases, it is still less than that of Kim's method under the configuration of a 2-, 4-, or 16-core platform. Moreover, from Table 3, we see that our method always performs better than Kim's method for the long code.

Table 4. Power consumption comparison for short code (64 bits).

Number of cores	Total power consumption (J)		SAV(%)
	Kim's	Ours	
2 cores	0.410858878	0.196867454	52.08%
4 cores	0.509487127	0.196867454	61.36%
16 cores	1.696699864	0.196867454	88.40%
64 cores	5.197915733	0.196867454	96.21%

Table 5. Power consumption comparison for long code (6,144 bits).

Number of cores	Total power consumption (J)		SAV (%)
	Kim's	Ours	
2 cores	8.504429178	7.757660882	8.78%
4 cores	10.748374329	7.9971606950	25.60%
16 cores	34.559857968	9.5694433970	72.31%
64 cores	107.455739469	16.5362849760	84.61%

3. Power Consumption Comparison

Since power consumption plays an important role in parallel decoding algorithms, in this subsection, we compare the power consumption of Kim's decoding method with that of our decoding method in the case that each method is separately realized on the same NoC-based SDR platform. (As mentioned in subsection V.1, Lin's method is not at the same BER level as our method and Kim's method, and we thus do not take the power consumption of Lin's method into consideration in this comparison.) Notice that the power consumption is composed of two parts: the core's power consumption measured by the McPAT tool [21] and the NoC power consumption measured by ORION 2.0 [22]. Both cores and the NoC are working on 100 MHz with 90-nm technology. SAV in the last column of

Table 4 and that of Table 5 is the percentage of power saved in relation to Kim's method (SAV=|$Power_{our}$–$Power_{kim}$|/$Power_{kim}$).

From Tables 4 and 5, we see that our method always performs better than Kim's method in terms of power consumption. For example, for the 64-bit short code word, our method can save about 52% to 96% power. For the 6,144-bit long code word, our method can save about 8% to 84% power. The reason for this phenomenon is that the power consumed in the NoC is proportional to the mount of data transmitted over the NoC, and Kim's method exchanges more data among the processors, as presented in subsection V.2. Therefore, our method consumes less power than Kim's method.

VI. Conclusion

In this paper, a novel parallel Viterbi decoding scheme was proposed to decrease decoding latency and power consumption, and an NoC-based SDR platform was used to evaluate the performance of our method. From the experiment results, we can conclude that our method and Kim's method are at the same BER level, superior to the BER level of Lin's method. Moreover, compared to Kim's method, with the exception of the short code decoding on the 64-core platform, our method performs with less latency and consumes less power. Therefore, our method is a useful parallel Viterbi decoding method for the SDR platform.

References

[1] A.L.G. Reis et al., "Introduction to the Software-Defined Radio Approach," *IEEE Latin America Trans.*, vol. 10, no. 1, Jan. 2012, pp. 1156-1161.

[2] P. Suárez-Casal et al., "A Multicore SDR Architecture for Reconfigurable WiMAX Downlink," *13th Euromicro Conf. Dig. Syst. Design: Architectures*, *Methods*, *Tools*, 2010, pp. 801-804.

[3] G. Fettweis and H. Meyr, "High Speed Parallel Viterbi Decoding: Algorithm and VLSI Architecture," *IEEE Commun. Mag.*, vol. 29, no. 5, May 1991, pp. 46-55.

[4] G. Fettweis, H. Dawid, and H. Meyr, "Minimized Method Viterbi Decoding: 600 Mbit/s per Chip," *IEEE GLOBECOM*, 1990, pp. 1712-1716.

[5] P.J. Black and T.H.Y. Meng, "A 1-Gb/s, Four-State, Sliding Block Viterbi Decoder," *IEEE J. Solid-State Circuits*. vol. 32, no. 6, June 1997, pp. 797-805.

[6] I. Habib, O. Paker, and S. Sawitzki, "Design Space Exploration of Hard-Decision Viterbi Decoding: Algorithm and VLSI Implementation," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 18, no. 5, May 2010, pp. 794-807.

[7] E. Yeo et al., "A 500-Mb/s Soft-Output Viterbi Decoder," *IEEE J. Solid-State Circuits*, vol. 38, no. 7, July 2003, pp. 1234-1241.

[8] T. Jarvinen et al., "Systematic Approach for Path Metric Access in Viterbi Decoders," *IEEE Trans. Commun.*, vol. 53, no. 5, May 2005, pp. 755-759.

[9] I. Lee and J.L. Sonntag, "A New Architecture for the Fast Viterbi Algorithm," *IEEE Trans. Commun.*, vol. 51, no. 10, Oct. 2003, pp. 1624-1628.

[10] J.J. Lai et al., "High Performance Viterbi Decoder on Cell/B.E.," *China Commun.*, vol. 6, no. 2, Apr. 2009, pp. 150-156.

[11] J. Kim, S. Hyeon, and S. Choi, "Implementation of an SDR System Using Graphics Processing Unit," *IEEE Commun. Mag.*, vol. 48, no. 3, Mar. 2010, pp. 156-162.

[12] C.S. Lin et al., "A Tiling-Scheme Viterbi Decoder in Software Defined Radio for GPUs," *7th Int. Conf. Wireless Commun., Netw. Mobile Comput.*, 2011, pp. 1-4.

[13] W. Plishker et al., "Applying Graphics Processor Acceleration in a Software Defined Radio Prototyping Environment," *22nd IEEE Int. Symp. Rapid Syst. Prototyping*, May 2011, pp. 67-73.

[14] T. Nylanden et al., "A GPU Implementation for Two MIMO-OFDM Detectors," *Int. Conf. Embedded Comput. Syst.*, July 2010, pp. 293-300.

[15] F. Clermidy et al., "A 477mW NoC-Based Digital Baseband for MIMO 4G SDR," *IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2010, pp. 278-279.

[16] G. Schelle, J. Fifield, and D. Grunwald "A Software Defined Radio Application Utilizing Modern FPGAs and NoC Interconnects," *Int. Conf. Field Programmable Logic Appl.*, 2007, pp. 177-182.

[17] J.H. Bahn, J. Yang, and N. Bagherzadeh, "Parallel FFT Algorithms on Network-on-Chips," *5th Int. Conf. Inf. Technol.: New Generations*, 2008, pp. 1087-1093.

[18] L.M. Ni and P.K. McKinley, "A Survey of Wormhole Routing Techniques in Direct Networks," *Comput.*, vol. 26, no. 2, Feb. 1993, pp. 62-76.

[19] R.Y. Shao, S. Lin, and M.P.C. Fossorier, "Two Decoding Algorithms for Tailbiting Codes," *IEEE Trans. Commun.*, vol. 51, no. 10. Oct. 2003, pp. 1658-1665.

[20] ETSI, "Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access, Multiplexing and Channel Coding (release 10)," 3GPP TS 36.212 V10.0.0, 2010.

[21] S. Li et al., "McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Manycore Architectures," *Proc. 42nd Annual IEEE/ACM Int. Symp. Microarchitecture*, 2009, pp. 469-480.

[22] A.B. Kahng et al., "ORION 2.0: A Power-Area Simulator for Interconnection Networks," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 20, no. 1, Jan. 2012, pp. 191-196.

Jian Wang received his BS, MS, and PhD in communication engineering from the University of Electronic Science and Technology of China, in 2005, 2008, and 2011, respectively. He is currently a lecturer in the School of Communication and Information Engineering, University of Electronic Science and Technology of China. His current research focuses on network-on-chip communication architectures, with a special interest in network-on-chip modeling and optimization.

Yubai Li received his MS in communication engineering from the University of Electronic Science and Technology of China in 1988. He is currently a professor with the School of Communication and Information Engineering, University of Electronic Science and Technology of China. His current research focuses on design methodologies for system on chip and network on chip.

Huan Li received his BS and MS in communication engineering from the University of Electronic Science and Technology of China, in 2001 and 2008, respectively. He is currently working toward his PhD in the School of Communication and Information Engineering, University of Electronic Science and Technology of China. His current research focuses on network-on-chip communication architectures, with a special interest in network-on-chip power consumption.