Jestr Journal of Engineering Science and Technology Review 7 (1) (2014) 52 – 59

JOURNAL OF
Engineering Science
and Technology Review

www.jestr.org

Research Article

A Parallelized Implementation of Turbo Decoding Based on Network on Chip Multi-core Processor

Chaolong ZHANG[1,*], Zhekun HU[2] and Jie Chen[1]

[1] *Institute of Microelectronics , Chinese Academy of Sciences, Beijing 10029, China*
[2] *The 709th Research Institute of China Shipbuilding Industry Corporation, Wuhan 430009, China*

Received 1 May 2013; Accepted 13 March 2014

Abstract

With the evolution of wireless communication systems, it is increasingly difficult for Application Specific Integrated Circuit (ASIC) solutions to meet the daily changing requirements. A network on chip (NOC) multi-core processor based on message-passing programming model is designed to implement the LTE-A turbo decoder in a parallel mode using pure Software Defined Radio (SDR) approach. The NOC is well balanced between the hardware and software design with a high degree of programmability and re-configurability. According to the features of the NOC multi-core processor, the implementation of turbo decoder is optimized to reduce the computational complexity and to increase the parallelization. Several aspects of turbo decoder are investigated in software radio approach rather than hardware. Compared with the results of the software simulation and the Field Programmable Gate Array (FPGA) demonstration, the NOC multicore processor is flexible to realize the proposed turbo decoding algorithm. In addition, our solution has comparable performance with other published ones.

Keywords: Network on Chip, Turbo Decoding, Low Complexity, Parallel Computation

1. Introduction

Since Berrou published the seminal paper about turbo coder[1], the properties of turbo code has been studied thoroughly in academy and industry for over twenty years. Recently, the research has been focusing on how to reduce the complexity and latency resulted from MAP decoding (also known as the BCJR algorithm). In order to relax the high demanding on memory during decoding, sliding window algorithm is invented to separate the long data block into short sub-blocks[2]. By adopting appropriate early stopping criteria[3], the number of iteration is reduced, so is the computational complexity.

However, with the development of communications systems, the baseband solutions are changed rapidly. Besides the computational complexity, being programmable and configurable is also very important for wireless baseband processing. Because of lack of flexibility, long development period and high cost of ASIC solutions, Software Defined Radio (SDR) arises to be a much better choice. Network on Chip, a new architecture for multi-core processor, is well balanced between hardware and software degign. Due to SDR approach, it is easy to map various signal processing algorithms onto NOC.

In this paper, a NOC multi-core processor is designed. Being different with[4], by using a message-passing programming model, the proposed NOC processor has low complexity in hardware and high flexibility of software which gives the programmer more freedom to realize the parallelized turbo decoding algorithm. The software radio approach makes the system easy to transplant and update. In addition, a parallel turbo decoder solution based on SDR is proposed and mapped onto the NOC. Several aspects of the turbo decoder such as the max* operators used in the log-MAP algorithm and early stopping criteria are re-investigated to reduced the computational complexity.

The other sections of this paper are organized as below: section 2 describes the turbo decoding algorithm and several modifications compared with[5]; the hardware architecture and software communication protocol of the proposed NOC processor are explained in section 3 and how the proposed turbo algorithm is mapped on to NOC is also given in this section; section 4 provides the simulation results coming from both the software simulation and the NOC on FPGA board; at last, section 5 contains the conclusion.

2. Turbo Coder and Decoder

2.1 Turbo decoder

In this paper, as a concrete example, the turbo coder used in 3GPP LTE-A is adopted. As illustrated in Fig 1, the LTE-A turbo encoder is composed of two parallel recursive systematic convolutional (RSC) encoders with constraint length 4.

* E-mail address: zhangchaolong@ime.ac.cn

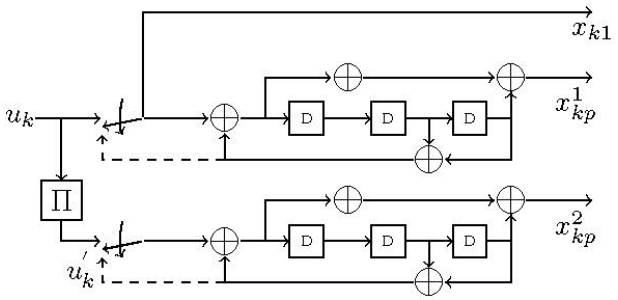


Fig. 1. Turbo encoder adopted by 3GPP LTE-A

In Fig. 1, u_k is the information bits, u'_k the interleaved version of u_k, x_{k1} the system information bit and u^1_{kp}, u^2_{kp} the parity check coded bits coming from the two RSC coders. The length of interleaver is K, where $40 \le K \le 6144$. A contention-free quadratic polynomial permutation (QPP) interleavers with block size K is defined as

$$\Pi(i) = (f_1 x + f_2 x^2) \mod K \tag{1}$$

The index can be calculated iteratively.

2.2 Turbo Decoder

The decoding algorithm is based on log-MAP algorithm and the algorithm is implemented using single-precision floating-point arithmetic on the NOC. The sliding-window algorithm described in[2] is improved in order to reduce the overhead communication between the micro-processors on the NOC. The architecture of the log-MAP decoder is shown as Fig 2:

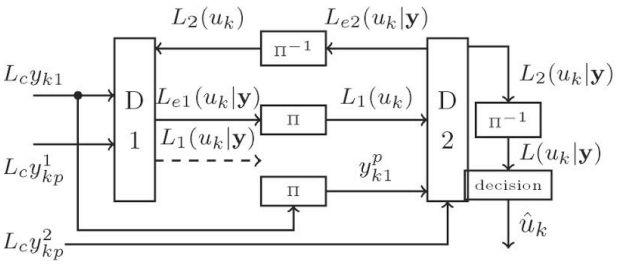


Fig. 2. Architecture of the Proposed turbo decoder

In Fig 2, D1 and D2 stand for the two soft-input soft-output decoders. The input to the decoder is supposed to be in log-likelihood ratio (LLR) format which takes the soft information about the channel gain and the noise variance into consideration. The LLR is defined as equation (2):

$$L(u_k \mid \mathbf{y}) = \ln \frac{P(u_k = +1 \mid \mathbf{y})}{P(u_k = -1 \mid \mathbf{y})} \tag{2}$$

With the assistance of trellis graph, equation (2) can be written as:

$$L(u_k \mid \mathbf{y}) = \ln \frac{\sum_{R_1} P(s', s, \mathbf{y})}{\sum_{R_0} P(s', s, \mathbf{y})} = \ln \frac{\sum_{R_1} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)}{\sum_{R_0} \alpha_{k-1}(s') \gamma_k(s', s) \beta_k(s)} \tag{3}$$

Where R_1, R_0 stand for the paths corresponding input 1 and 0 in the trellis graph from time $k-1$ to k, respectively. $P(s', s, \mathbf{y})$ represents the joint probability of receiving the sequence $\mathbf{y}$ and being in state s' at time $k-1$ and in state s at time k. α, β are called the forward and backward variables. Actually, for the memory-less channel α, β, γ can be calculated iteratively.

The main difference between the variant of MAP algorithm is the so-called "$\max^*$" operator[5] which use the Jacobi algorithm to calculate $\ln(e^x + e^y)$, namely:

$$\max^*(x, y) = \ln(e^x + e^y) \tag{4}$$

Currently，There are four "$\max^*$" operators used:

1) Log-MAP operator

$$\max^*(x, y) = \max(x, y) + \ln(1 + e^{-|x-y|}) \tag{5}$$

2) Max-MAP operator

$$\max^*(x, y) = \max(x, y) \tag{6}$$

3) Constant-log-MAP operator

$$\max^*(x, y) = \max(x, y) + \begin{cases} 0 & \text{if } |x-y| \text{ T} \\ C & \text{if } |x-y| \le \text{T} \end{cases} \tag{7}$$

4) Linear-log-MAP operator

$$\max^*(x, y) = \max(x, y) + \begin{cases} 0 & \text{if } |x-y| \text{ t} \\ k(|x-y| - T) & \text{if } |x-y| \le \text{T} \end{cases} \tag{8}$$

In the fourth max* operator, the linear-log-MAP operator, it's shown in Ref. [8] that the optimal values for k and T are $k = -0.24904$ and $T = 2.5068$, respectively. Different from the linear-log-MAP, first introduced in[8], in this paper, through quadratic curve fitting, we get a quadratic polynomial approximation to the max* operator. The quadratic max* is shown as:

$$\max^*(x, y) = \max(x, y) + k_2(|x-y|^2) + k_1(|x-y|) + k_0 \tag{9}$$

With 95% confidence bounds,

$$k_2 \in [0.05635, 0.05714],$$

$$k_1 \in [-0.3783, -0.3751],$$

$$k_0 \in [0.6503, 0.6531].$$

The $\max^*$ operators are shown in Fig 3.

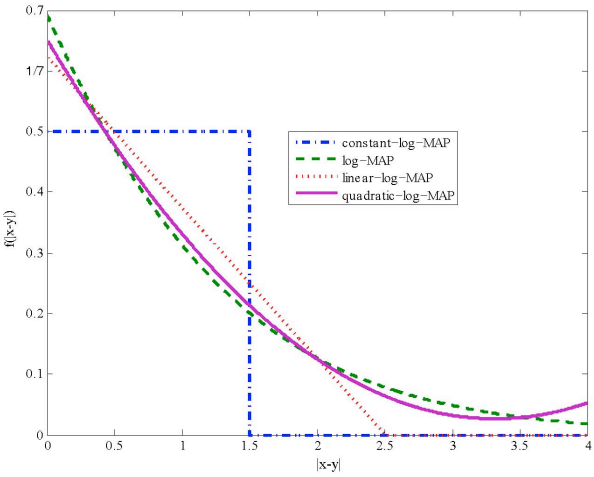


Fig. 3. different $\max^*$ operators used in log-MAP, linear-log-MAP, constant-log-MAP and the quadratic-log-MAP.

As shown in Fig 3, the quadratic-log-MAP fits the log-MAP most effectively. In section 4, the BER-SNR curves resulted from different max* operators are also shown that algorithm using the quadratic-log-MAP operator has the best performance.

Another technique used in this paper to reduce the computational complexity is the early stopping criteria. In general, as the number of iteration increases, the performance of the decoder will not be improved linearly which means too many iterations only bring high complexity and high latency. To combat this problem, early stopping criteria is adopted during the decoding process[3]. Basically, there are two kinds of early stopping criteria: 1) the hard-decision principle, which compares two output of successive iteration of SISO decoder. If the outputs are equal, the iteration is stopped. 2) the soft-decision principle, which compares the LLR of the SISO with a given threshold. In this paper, the later principle is adopted, and the impact of different threshold on the BER performance is simulated and analyzed in section 4. Of course, a maximum number of iteration is set in case that the iteration will not stop.

3. *Mapping Turbo decoder to NOC processor*

In this section, the architecture of the NOC multi-core processor is described briefly and the parallel turbo decoding is explained and mapped onto the NOC.

3.1 The Hardware of the NOC Multi-core Processor

Based on the message-passing programming model, within a 4×4 2D mesh network on chip, the multi-core processor is designed with 16 small RISC micro-processors[6]. The architecture of the NOC and each micro-processor is shown as Fig 4.

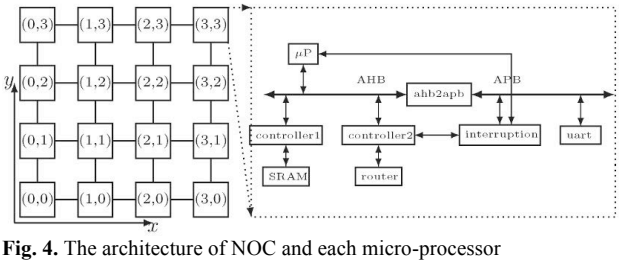


Fig. 4. The architecture of NOC and each micro-processor

In Fig 4, controller1 is the SRAM controller while controller2 is the router controller, namely the network interface. The processing unit (PU) on the NOC is a small-scale 32-bit RISC processor which has 3-level pipeline architecture. The data and command are accessed through AHB bus from the private 32 KB SRAM. Inter-processor communication is conducted via the worm-hole switch routers and network interfaces using two virtual channels. In the wormhole switching network, the data packet is passed from one PU to another in the format of "flit" containing the routing information and data whose format will be shown later. The circuit of the worm-hole router with two virtual channels is shown as Fig 5.

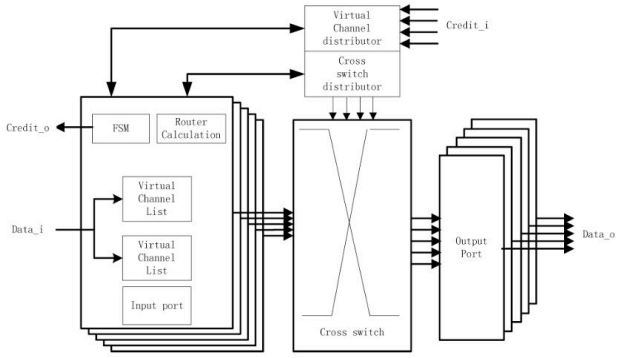
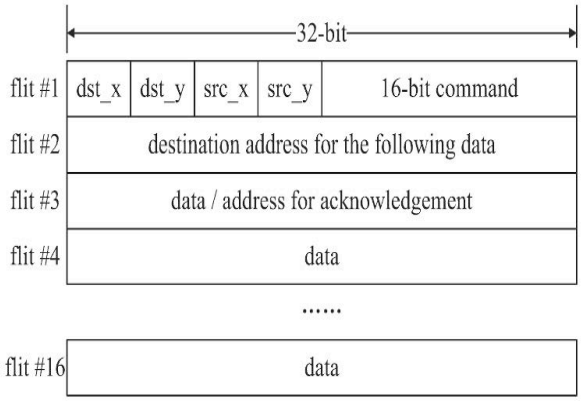


Fig. 5. circuit block graph of the wormhole switch router

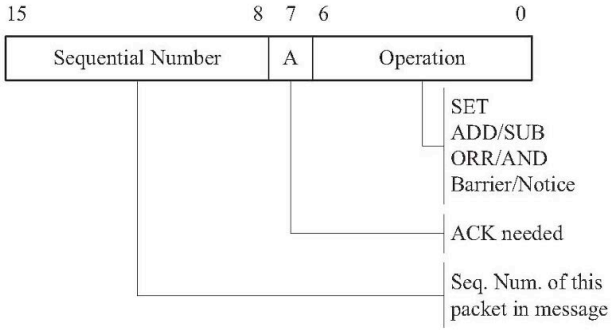
The capacity of the virtual channel list is set to 8 during the routing processing. The adjacent routers utilize the measure of credit to control the data flow as show in Fig 5. In order to avoid deadlock during routing process, XY-algorithm is adopted[6]. The XY-algorithm is not only easy to implement but also effective to guarantee the order and consistence of the data packets.

3.2 The Software Protocol of the NOC Multi-core Processor

The hardware provides the foundation in switching the data packet while the software tells the hardware what and where to send. To achieve this goal, inter-processor software communication protocol is to be made. As mentioned before, the data is transmitted between the PUs in the format of "flit" which contains the routing information and data. One data packet is composed of 16-32 flits. The format of a data packet is shown as Fig 6(a).



(a)



(b)

Fig. 6. the format of it and command field in the data packet

The first flit of the data packet is composed of five fields: the first and second fields are the coordinate of the destination PU to which the packet are sent while the third and the fourth are the coordinate of the source PU from which the packet are sent; the last fields is a 16-bit command whose format is shown as Fig 6(b). The second flit of the data packet is the starting address (in C programming language, the pointer of the data) of SRAM to store the data. The third flit may be either the data part or the address for the acknowledgement according to the parameter set in the programming API. The following flits contain the data.

Basically, the software protocol provides three important APIs for the programmer to conduct the data transmitting from one PU to another.

1) ***void get_curr_pos(int *x, int *y);*** used to get the coordinate of the current PU, and stored the result in ***x*** and ***y***.
2) ***int send_message(long msg[], int length, int dest_x, int dest_y, unsigned long dest_address,unsigned long cmd, int vcid, int blocking);*** used to send messages between the PU on the chip. ***msg*** and ***length*** is the starting address and length of the data to be sent, respectively. ***(dest_x,dest_y)*** is the coordinate of the destination PU. ***cmd*** is the command which will be executed on the data and can be defined by the programmer. ***vcid*** is used to indict which virtual channel is to be used to send the data and blocking tell the destination PU to send back an acknowledgement message if set to be 1.
3) ***void barrier(volatile struct sync s* sync_p, unsigned int proc_mask, int proc_num, int major_x, int major_y);*** used to synchronize the data stored in different PUs. ***sync_p*** is the synchronizing struct. ***proc_mask***: a two bytes value, masks the processors engaged in synchronisation. ***proc_num***: the number of processors engaged in synchronization, equal to the number of 1s in ***proc_mask***. ***(major_x, major_y)***: the coordinate of the processor responsible for the synchronization.

3.2 The Mapping Procedure

The key step of mapping the turbo decoder to the NOC is to separate the data block $\mathbf{y}$ into sub-blocks without interaction with each other so as to reduce the intercommunication between the PUs. The received symbols in $\mathbf{y}$ is separated without overlapping by improving algorithm used in[2][7]. A received $\mathbf{y}$ of length L is divided into K sub-blocks of length $M = L / K$ trellis stages. However, instead of using overlapping, the forward and backward variables which were computed in the previous iteration of the adjacent sub-blocks are utilized to provide appropriate boundary distributions for each sub-block MAP decoder. All sub-block MAP decoders perform the computation simultaneously and hence, the proposed algorithm (as shown in Fig. 7) reduces the decoding computation time exactly by a factor of K.

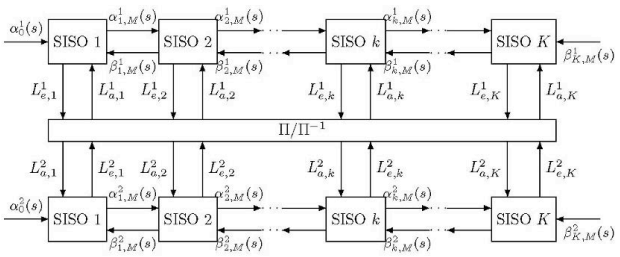


Fig. 7. Parallel turbo decoding

In Fig. 7, each SISO decoder is implemented on one PU, and all the SISO can be executed in a parallelized and pipelined way.

At first, the received data for each RSC codes are separated into several continuous non-overlapping sub-blocks (also known as windows). Each sub-block is decoded separately using the quadratic-log-MAP method. The initial values for α and β variables come from previous iteration of adjacent windows. All the sub-blocks are decoded in the parallel mode at the same time, during the next iteration the initial values for all the sub-blocks are ready to be read. As can be seen, no extra procedure needed for the initialization of state probabilities at each iteration. The length of the sub-blocks is an important factor that will be analyzed in section 4.

The decoding timing diagram of the parallel sub-block processors (also known as the SISOs) on the NOC is shown as figure 8(a), and its vector version is shown as 8(b). The symbols that processed at the same time are written as a vector. The length of the vector is shown is M. The elements of each vector coming from different sub-block processors.

SISO_1

Backward: $y_{N-1}y_{N-2}\cdots y_1y_0$

$b_Nb_{N-1}\cdots b_1b_0$

Forward: $y_0y_1\cdots y_{N-2}y_{N-1}$

$a_0a_1\cdots a_{N-1}a_N$

Output: $x_0x_1\cdots x_{N-2}x_{N-1}$

SISO_2

Backward: $y_{2N-1}y_{2N-2}\cdots y_{N+1}y_N$

$b_{2N}b_{2N-1}\cdots b_{N+1}b_N$

Forward: $y_Ny_{N+1}\cdots y_{2N-2}y_{2N-1}$

$a_Na_{N+1}\cdots a_{2N-1}a_{2N}$

Output: $x_Nx_{N+1}\cdots x_{2N-2}x_{2N-1}$

$\vdots$

SISO_M

Backward: $y_{MN-1}y_{MN-2}\cdots y_{(M-1)N+1}y_{(M-1)N}$

$b_{MN}b_{MN-1}\cdots b_{(M-1)N+1}b_{(M-1)N}$

Forward: $y_{(M-1)N}y_{(M-1)N+1}\cdots y_{MN-2}y_{MN-1}$

$a_{(M-1)N}a_{(M-1)N+1}\cdots a_{MN-1}a_{MN}$

Output: $x_{(M-1)N}x_{(M-1)N+1}\cdots x_{MN-2}x_{MN-1}$

(a)

Backword: $\mathbf{y}_{N-1}\mathbf{y}_{N-2}\cdots\mathbf{y}_1\mathbf{y}_0$

$\mathbf{b}_N \quad \mathbf{b}_{N-1}\cdots\mathbf{b}_2\,\mathbf{b}_1\,\mathbf{b}_0$

Forword: $\mathbf{y}_0\mathbf{y}_1\cdots\mathbf{y}_{N-2}\mathbf{y}_{N-1}$

$\mathbf{a}_0\mathbf{a}_1\cdots\mathbf{a}_{N-2}\mathbf{a}_{N-1}\mathbf{a}_N$

Output: $\mathbf{x}_0\mathbf{x}_1\cdots\mathbf{x}_{N-2}\mathbf{x}_{N-1}$

(b)

Fig. 8. The timing diagram of the decoder running on the NOC multicore processor.(b) is the vector version of (a).

The proposed structure is inspired by the message-passing algorithm which can be illustrated by graphs. In this paper, the graphs are divided into several sub-graphs and scheduled for parallel decoding. This partition scheme is useful to parallelize the decoding of on constituent code. The graph of the turbo decoder and its partitioned version are shown in Fig. 9.

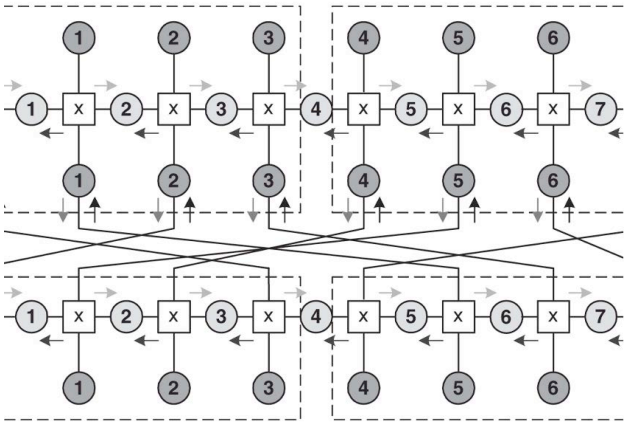


Fig. 9. The graph of turbo decoder and its partitioned version

The optimum algorithm is the log-MAP method. Here we use the quadratic-log-MAP version which is analyzed in section 2 and will be simulated in section 4. The processing of the sub-blocks in two constituent codes can be executed in parallel mode. However, thanks to the QPP interleaver[7] adopted in LTE-A standard, in parallel turbo code, when the two RSCs are the same encoder(which is satisfied in this paper), the SISO blocks for all constituent codes can be shared. Eventually, the number of processors used to implement turbo decoding can be only half of the method shown in Fig. 7. The new architecture of the parallel turbo decoder is shown as Fig. 10.

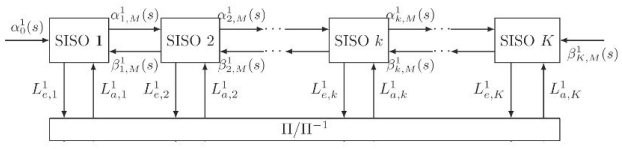


Fig. 10. New parallel turbo decoder architecture.

All the SISOs illustrated in Fig. 7 and Fig. 10 are mapped onto the NOC as depicted in Fig. 11.

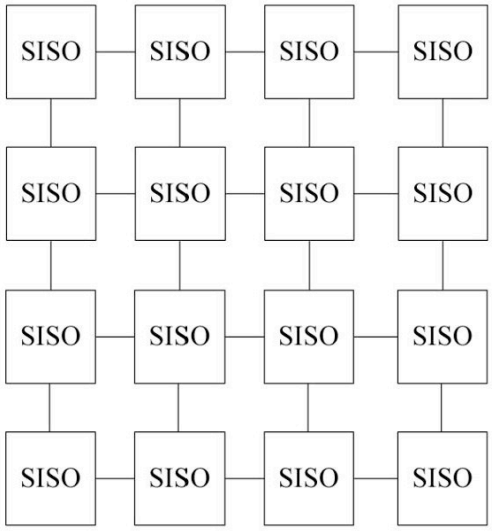


Fig. 11. Map the SISO onto the NOC multi-core processor. Each PU is executed as a SISO decoder.

4 Simulation Results

4.2 Frequency and area of the NOC

To evaluate the frequency and area of the NOC, The PUs and the router of the multi-core processor are synthesized using the low leakage low threshold voltage 65nm library of Semiconductor Manufacturing International Corporation (SMIC). The frequency and area of the single PU and the router are shown as table 1.

Table 1: The Synthesis Results of PU and Router

	PU	**Router**
clock	200MHz	400MHz
combinational circuit	49204.44μm^2	21042.00μm^2
sequential circuit	115839.36μm^2	50555.88μm^2
SRAM	196479.34μm^2	×
total area	361523.14μm^2	71597.88μm^2
gate equivalent	188293	37291

From table 1, we can see that the area of the router accounts for about $1/5$ of the single PU and about 75% of the router is sequential logic circuit. The router can work at frequency double times as that of the PU which means when the bandwidth and latency can not satisfy the applications, we could separate the clock of the PU and the router to improve the performance.

4.3 Simulation of Turbo Decoder on the NOC

The NOC processor is verified on an Altera EPS180 evaluation FPGA board and the BER-SNR(Bit Error Rate-Signal to Noise Ratio) curve(under AWGN channel, the length of interleaver is 1024) is drawn as Fig. 12. which is the same as the simulation results coming from Matlab simulation on PC and shows that the parallel turbo decoding algorithm can work correctly on the NOC multi-core processor.

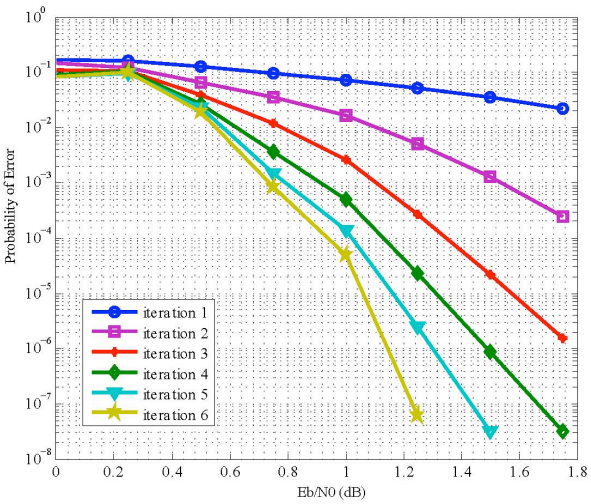


Fig. 12. BER-SNR curve(under AWGN channel, with interleaver length 1024)

4.4 Performance of Different Max* Operators

The BER-SNR curves resulted from different max* operators mentioned in section 2 is drawn in Fig 13 and Fig 14. The data block length is 512 in Fig 10 while in Fig. 14 the data block length is 4096. In Fig. 13 and Fig. 14, the blue lines are the BER-SNR curves under AWGN channel while the red lines are the BER-SNR curves under Rayleigh fading channel.

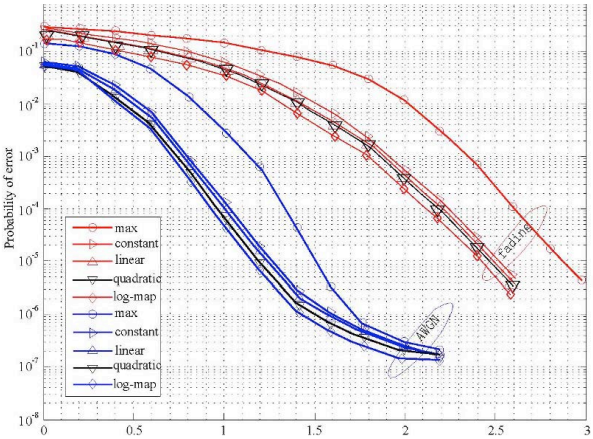


Fig. 13. the BER-SNR curves with different max* operators under AWGN channel(the blue curves) and Rayleigh fading channel(the red ones), with data block length equal to 512.

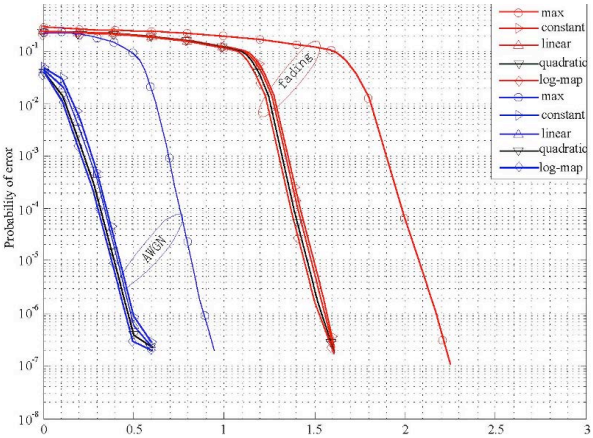


Fig. 14. the BER-SNR curves with different max* operators under AWGN channel(the blue curves) and Rayleigh fading channel(the red ones), with data block length equal to 4096.

For each case, when BER=10^{-5}, the E_b / N_0 is shown in Table 2.

Table 2: E_b / N_0 Required when BER is 10^{-5}

	AWGN		Rayleigh Fading	
	K=512	K=4096	K=512	K=4096
max-log-MAP	1.51dB	0.82dB	2.85dB	2.09dB
constant-log-MAP	1.28dB	0.45dB	2.52dB	1.59dB
linear-log-MAP	1.22dB	0.44dB	2.49dB	1.57dB
quadratic-log-MAP	1.16dB	0.41dB	2.45dB	1.55dB
log-MAP	1.12dB	0.38dB	2.41dB	1.53dB

From Fig. 13 and Fig. 14, the max-log-MAP has the worst performance requiring at least 0.39dB higher (K=512, AWGN) than the log-MAP. Except the max-log-MAP, all other three algorithms (constant-log-MAP, linear-log-MAP and the quadratic-log-MAP) are close to the log-MAP. In particular, The quadratic-log-MAP boasts the narrowest gap with the log-MAP(only 0.02dB when K=4096, Rayleigh fading).

If the total number of data block used in the simulation is increased, the error floor would be reached[15] which is obvious in Fig. 13 when K=512 under AWGN channel. In the error floor, the five curves will converge together. Hence, the algorithm may have a significant impact at low SNR regime and becomes insignificant at high SNR regime. The error-floor phenomenon means that in a software radio implementation, we can choose the algorithm adaptively according to the SNR to reduce the computational complexity.

4.5 Early Stopping Criteria

The results from the previous section were generated under the condition that the decoder stopped the iteration when it converged, namely all the errors in the data blocks were corrected. During the simulation, the transmitted data blocks are known at the receiver. However, the decoder does not know the data, and thus a criteria is needed to stop the iteration. In practice, the BER performance will not be improved endlessly by increasing the number of iteration, so utilize an early stopping criterion will generate a much higher throughput in a software radio implementation[8].

There are many early stopping criteria published[8]. They are proposed based on the cross entropy between iterations or on the sign-difference ratio. In this paper, a stopping criteria based on the log-likelihood ratio is employed. The decoder will stop once the LLRs are greater than a threshold Δ_h ,namely the decoder stops when Equation (10) is satisfied.

$$\min_{1\le k\le K}\{|\Delta_2 k|\} > \Delta_h \tag{10}$$

Obviously, the stopping threshold Δ_h has a significant influence on the performance. If Δ_h is set to be a small number, the decoder will not operate enough iterations. If,

on the contrast, Δ_h is large, then the decoder will run in vain and the throughput will decrease.

BER-SNR curves with different Δ_h are drawn in Fig. 15. As shown in Fig. 15, $\Delta_h = 1$ and $\Delta_h = 6$ are too small and the BER performance suffers under these two conditions. When $\Delta_h = 10$, the BER-SNR curve is almost overlapped with that of perfect decoding. Interestingly, the BER with $\Delta_h = 10$ is sometimes lower than that of ideal stopping. Actually, the output bit sequence of a turbo decoder fluctuates from one iteration to next at times. When the received symbol sequenced is corrupted by the channel seriously, it is impossible for the turbo decoder to correct some bits. However, if the LLR are high, the early-stopping rule will halt the decoder even the BER is not zero. Hence, the stopping criteria can sometimes lower the BER.

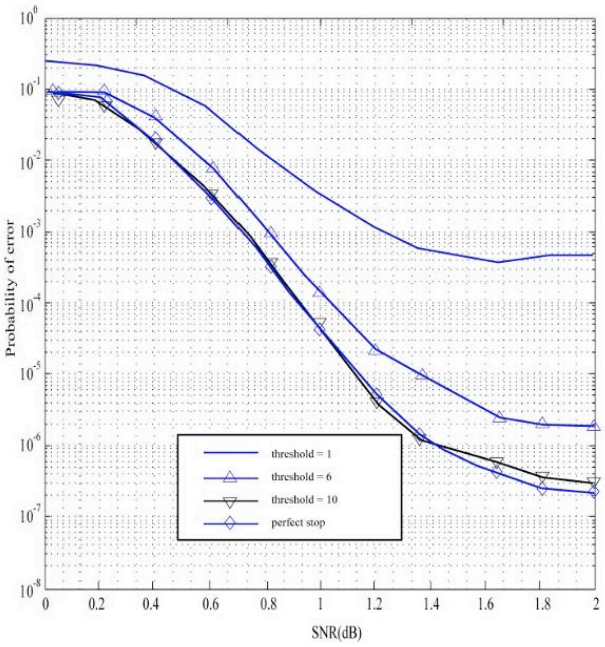


Fig. 15. BER-SNR curve with quadratic-log-MAP decoding in AWGN with different threshold Δ_h (K=512)

4.6 Throughput and Circuit Complexity

Several turbo decoder implementations based on NOC have been published. However, because different degree of flexibility in the computation to be supported always comes at a relevant cost, both in terms of occupied area and achievable throughput, comparison between them tends to be difficult. In Table 3, some multi-core processor based architectures are reported. To conduct a fair comparison, area and throughput of different parts in the NOC are given in table 3. All data have been converted to a 65 nm CMOS technology.

To sum up, the main purpose of table 3 are:1) compare NOC based solutions proposed in this paper with other architectures used in parallel turbo decoders. From this point, comparisons are conducted on the area and throughput. 2) to evaluate the flexibility cost of the NOC architecture. From this point, direct area comparison makes no sense. However, we can get the overhead rate associated with various NOC architectures.

Table 3(part 1): Implementations of Turbo Coder Based on Various NOC Architecture.(All data has been converted to the 65nm CMOS technology), Part 1: results about the NOC Processor.

NOC Architecture	Clock (MHz)	Area (mm^2)	Throughput (Mbps)	Throughput to area ratio
Butterfly[10]	345	1.5	138	92
Benes[6][10]	381	0.96	152	158
Toroidal[10]	200	1.2	162	135
Kautz[10]	200	1.1	164	149
This paper	200	1.13	157	138

Table 3(part 2): Implementations of Turbo Coder Based on Various NOC Architecture.(All data has been converted to the 65nm CMOS technology), Part 2: results about the Routers

Router Architecture	Clock (MHz)	Area (mm^2)	Throughput (Mbps)	Throughput to area ratio
[11]	184	1.5	19.6	13
[12]	-	9.8	91	9.2
This paper	400	15	357	13.8

Table 3(part 3): Implementations of Turbo Coder Based on Various NOC Architecture.(All data has been converted to the 65nm CMOS technology), Part 3: results about the decoders

Decoder Architecture	Clock (MHz)	Area (mm^2)	Throughput (Mbps)	NOC overhead
[13]	150	8.4	75	13%
[14]	250	10.7	187	10%
[15]	352	10	352	11%
this paper	200	7	157	8%

As can be seen from part 1 of table 3, even we use a SDR approach to implement the turbo encoder, the area and throughput is also comparable with the ones listed in Table 3. The message-passing programming model exploits the NOC effectively. Compared with other architecture[10] such as Butterfly, Benes, Toroidal and Kautz, the hardware and software proposed in this paper outperforms them in the aspect of degree of programmability. From part 2 of table 3, the router architecture designed in this paper has the highest throughput and throughput to area ratio[11][12]. Part 3 of table 3 tells us that even the SISO proposed in this paper has smaller throughput compared with[13][14][15], we also have smaller overhead.

5 Conclusions

In this paper, a parallel turbo decoding algorithm is implemented in the pure software radio approach on a NOC multi-core processor. The NOC multi-core processor is evaluated correctly using FPGA. As an example, the Turbo encoder defined in LTE-A is adopted.

During the parallel implementation, the way how to separate the data block in to sub-blocks with proper length is given. The BER-SNR performance of decoder with different interleaver length is simulated in both serial and parallel style. It is shown that there is little performance degradation resulted from the parallel decoding algorithm implemented on NOC multi-core processor.

Compared with other implementations, our solution can finish the turbo decoder with high parallel degree and low overhead complexity. The BER performance is also comparable with the published NOC architectures.

6 Acknowledgements

This work was financially supported by the National Natural Science Foundation of China (No. 61201265, No. 61221004 and No. 61376027).

References

1. C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1," presented at the Technical Program, Conference Record, IEEE International Conference on Communications, 1993. ICC '93 Geneva, 1993(2), pp. 1064–1070
2. H. Lim, Y. Kim, and K. Cheun, "An efficient sliding window algorithm using adaptive-length guard window for turbo decoders," Journal of Communications and Networks, 14(2), 2012, pp. 195–198
3. P. Reddy, F. Clermidy, A. Baghdadi, and M. Jezequel, "A low complexity stopping criterion for reducing power consumption in turbo decoders," presented at the Design, Automation Test in in Europe Conference & Exhibition (DATE), 2011 , pp. 1-6
4. C. Condo, M. Martina, and G. Masera, "A Network-on-Chip-based turbo/LDPC decoder architecture," presented at the Design, Automation Test in Europe Conference Exhibition (DATE), 2012, pp. 1525–1530.
5. M. C. Valenti and J. Sun, "The UMTS turbo code and an efficient decoder implementation suitable for software defined radios," International Journal of Wireless Information Networks, 2001(8), pp. 203–216
6. HU Zhekun and CHEN Jie, "Design of a Message-passing Multi-core System," Journal of Hunan University(Natural Sciences), 40(8) Aug. 2013, pp. 103-109
7. S. Yoon and Y. Bar-Ness, "A parallel MAP algorithm for low latency turbo decoding," IEEE Communications Letters, 6(7),2002, pp. 288–290
8. A. Taffin, "Generalised stopping criterion for iterative decoders," Electronics Letters, 39(13),2003, pp. 993–994
9. M. J. Thul, F. Gilbert, and N. Wehn, "Concurrent interleaving architectures for high-throughput channel coding," presented at the 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03), 2003(2), pp. 613–616
10. M. Martina, G. Masera, H. Moussa, and A. Baghdadi, "On chip interconnects for multiprocessor turbo decoding architectures," Microprocessors and Microsystems, 35(2), 2011 pp. 167–181.
11. F. Gilbert, M. J. Thul, and N. Wehn, "Communication centric architectures for turbo-decoding on embedded multiprocessors," presented at the Design, Automation and Test in Europe Conference and Exhibition, 2003, pp. 356–361.
12. I. Ahmed and C. Vithanage, "Dynamic reconfiguration approach for high speed turbo decoding using circular rings," Proceedings of the 19th ACM Great Lakes symposium on VLSI. ACM, 2009, pp. 475–480.
13. M. May, T. Ilnseher, N. Wehn, and W. Raab, "A 150Mbit/s 3GPP LTE Turbo code decoder," presented at the Design, Automation Test in Europe Conference Exhibition (DATE), 2010, pp. 1420–1425.
14. J.-H. Kim and I.-C. Park, "A unified parallel radix-4 turbo decoder for mobile WiMAX and 3GPP-LTE," presented at the IEEE Custom Integrated Circuits Conference, 2009. CICC '09, 2009, pp. 487–490.
15. P. Urard, L. Paumier, M. Viollet, E. Lantreibecq, H. Michel, S. Muroor, B. Coates, and B. Gupta, "A generic 350 Mb/s turbo-codec based on a 16-states SISO decoder," presented at the Solid-State Circuits Conference, 2004. Digest of Technical Papers. ISSCC. 2004 IEEE International, 2004(1), pp. 424–536