Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 2839797, 8 pages
https://doi.org/10.1155/2018/2839797

Research Article

Architecture for Collision-Free Communication Using Relaxation Technique

Saeed ur Rehman,[1,2] Rizwan Akhtar,[1] Zuhaib Ashfaq Khan,[2] and Changda Wang[1]

[1]*School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China*
[2]*COMSATS University Islamabad, Attock Campus, Pakistan*

Correspondence should be addressed to Changda Wang; changda@ujs.edu.cn

Received 1 July 2018; Revised 25 September 2018; Accepted 18 October 2018; Published 4 November 2018

Guest Editor: Sarmadullah Khan

In today's world we are surrounded by world of smart handheld devices like smart phones, tablets, netbooks, and others. These devices are based on advance technologies of multiple-input and multiple-output, Orthogonal Frequency Division Multiplexing (OFDM), and advance data reliability techniques such as forward error corrections. High data rates are among the requirements of these technologies for which turbo and low density parity check codes (LDPC) are widely used in these standards. In order to get high speed, we need multiple and parallel processors for the implementation of such codes. But there exists a collision problem as a consequence of parallel processor. This problem results in increase latency and increase of hardware complexity. In this work an approach for collision problem is presented in which network relaxation technique is used which is based on a fast clique detection. The proposed approach results in high throughput in terms of latency and complexity. Furthermore, the proposed solution is able to solve the collision problem by connecting network optimization for achieving high throughput.

1. Introduction

In modern era of an advancement in technology, different types of smart devices like smart phones, tablets, etc. are communicating with each other nodes. These devices are based on advance technologies which consist of multiple-input and multiple-output, OFDM, and advance data reliability techniques such as forward error corrections [1–9]. High data rates are among the requirements of these technologies for which turbo and low density parity check codes are widely used in these standards. In order to get high speed we need multiple/parallel processors for the implementation of such codes. Such codes are developed by using multiple processors which are linked with the data storage memory elements via connecting networks.

In turbo codes we have an algorithm which is based on iteration for the computation that improved the overall performance. So to exploit these codes with their iterative nature is a challenging task which is overcome by the use of parallel multiple processors. The required throughput is obtained by the use of multiple processors in parallel architecture. This solution increases the throughput as the latency of the system becomes the latency of the constituent subblocks [3]. Moreover, the cost of the system and the complexity are increased due to parallel nature of the architecture.

In Section 2 the state of the art is given. Section 3 gives the problem formulation. The proposed work is given in Section 4 with the pedagogical example in Section 5. Experiment and results are presented in Section 6 whereas Section 7 concludes the paper.

2. State of the Art

Different parallel processing elements depending on the interleaving law may access the same memory block simultaneously as shown in Figure 1. This architecture consists of processors linked with banks. The controller consists of control bits for connecting network and bits for address sequences of the banks. There exists memory access problem in such architectures known as the "collision" problem [7].

For solving collision problem, the existing solutions can be categorized into two families. The first family includes

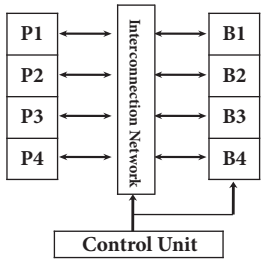


Figure 1: General architecture.

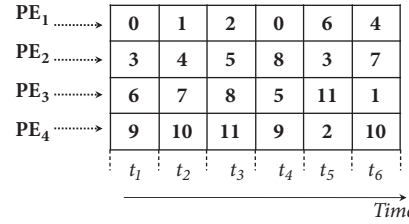


Figure 2: Memory access schedule.

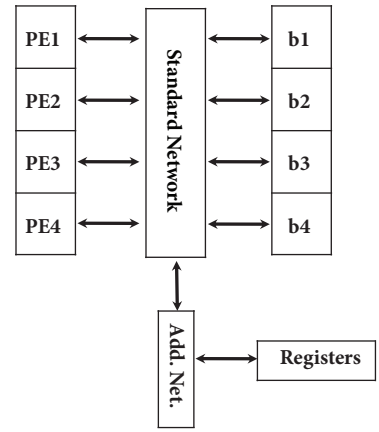


Figure 3: Architecture generated by memory adaptation.

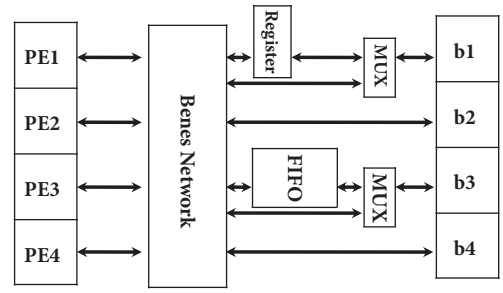


Figure 4: Architecture generated by time relaxation.

solutions like [10–14] in which the problem is solved without any changes in the standard architecture; i.e., these approaches use a standard network/memory. In [10], that is proposed with the network option but limited to certain codes. Metaheuristics approach [12] overcame this limitation but not the polynomial time. The polynomial is introduced in [13].

Works of [15–19] are of the second category which offers certain adaptation in the architecture for the storage and/or connecting network mechanism. Due to the adaptation in the real architecture the results may lack throughput in terms of latency and complexity. Reference [15] proposed to use buffers in the BENES network [20] for solving collisions. Reference [16] used advance network techniques with dedicated mechanism of routing. The resultant solution needs buffering for such mechanism as well. The work of [17] presents a network-on-chip interconnection architecture. This network allows supporting the access conflicts thanks to dedicated routing algorithm. Reference [18] is based on use of additional memory (memory adaptation) when there is no other existing solutions. But, this approach generates a huge amount of additional registers leading to expensive architectures, especially for LDPC. Since the goal of [18] is to respect a user defined network, this constraint impacts this final architectural cost on a larger scale.

Our proposal is to introduce degrees of liberty not in the memory architecture, but in the network. Hence, we will target the original source of the conflict, and we will be able to generate strongly optimized architectures. This concept will be later referred in this paper as network adaptation.

3. Problem Formulation

We will demonstrate a multiple processor parallel architecture having **P** processors which are denoted by PE which is linked with B banks of memory where number of processors are equal to number of banks. We have the size of storing data referred as D.

Figure 2 shows the example to explain the above-mentioned approach. Processors are considered as 4 which is equal to the number of banks. The time to process the data is represented in columns of the table. In first time the four processors will need 0, 3, 6, and 9 for processing as shown in Figure 2. This is a typical way to show turbo decoding. The main goal is to find collision-free mapping with specific laws and number of processors with the option of interconnection network.

The state-of-the-art architecture [12] is shown in Figure 1. The control bits will be generated in very irregular pattern having limited optimization. However, the regular pattern for such cases can be very useful for optimized results.

The regularity pattern can be seen in [18] but the finale architecture will have overhead cost due to inclusion of the logical memory elements and its connecting network (see Figure 3).

In [15], these could be avoided in which BENES network is considered in the architecture along with additional buffer as shown in Figure 4, called time relaxation resulting in latency.

In this work, an approach is presented in which a special technique is used which results in high throughput in terms of latency and complexity. The proposed solution is able

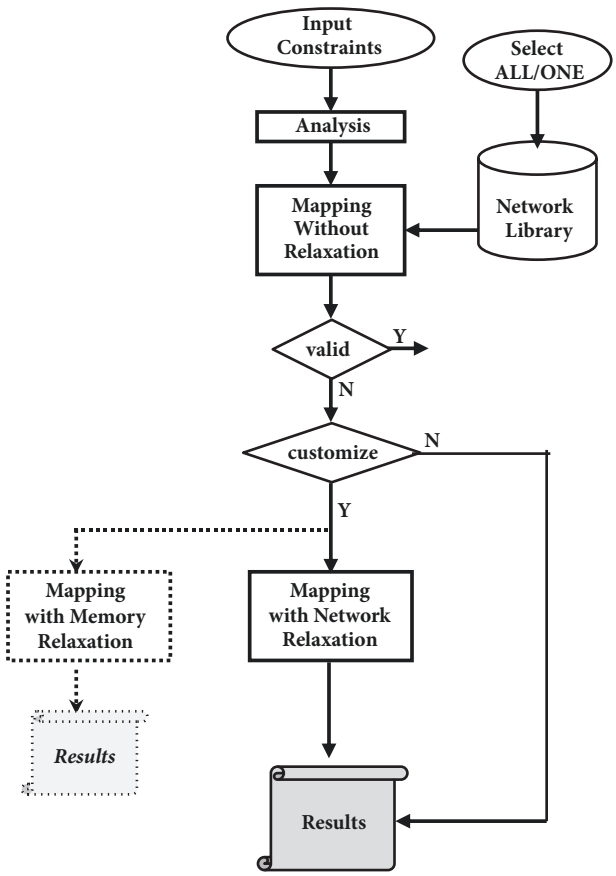


Figure 5: Proposed design flow.

to solve the collision problem by the connecting network optimization for achieving high throughput.

4. Proposed Approach

In [18], it is observed that the selected connection network has a vital role in the complexity of the proposed architecture. The proposed technique aims to use the principle based on both memory mapping space and the network, as can be seen in Figure 5. The figure presents a design space exploration for a given memory mapping problem.

The input constraint is addressed in the first step which includes laws, number of processors, and network selection. It is to determine whether a collision-free in-place memory mapping solution may exist or not for this set of constraints. This will provide useful information for the memory mapping step of our flow. The second step is to apply appropriate algorithm targeting a connecting network stored in the library. This mapping step is performed with a modified version of the mapping algorithm with memory relaxation proposed in [18]. The modifications aim to see all the collision-free mapping solution space and to stop the mapping algorithm as soon as a register should be added because of unsolvable memory mapping conflicts. If the algorithm is stopped for this specific reason, which means the selected interconnection network is too much constrained for the interleaving law.

Then, if the designer validates the customization of this interconnection network, our memory mapping algorithm with network relaxation is applied.

In the end, the cost is calculated and the results are generated. We repeat all the above process for all the available networks one by one. The user can restrict the library to explore only a subset of network(s); otherwise all the networks are explored one by one. We also integrate for experimental purpose the memory mapping approach based on memory relaxation, as given on Figure 5.

4.1. Analysis through Fast Clique Detection Algorithm (FCD). Memory mapping problem could either be based on [10] like in turbo decoders called *in-place* or be based on [18] like in LDPC decoders called *dual-memory* mapping. A fast clique detection algorithm is then proposed which determined the category of the problem.

4.1.1. Algorithm Principles. For a particular block of $\boldsymbol{L}$ data with the parallelism $\boldsymbol{P}$, our approach first constructs $\boldsymbol{L} * \boldsymbol{L}$ adjacency matrix and then fills the matrix according to the data access: if two data are accessed in the same time (e.g., data 1, 2, 3, and 4 at cycle t_1 in Figure 2), then a link between them will be stored in this adjacency matrix. This model is in fact a well-known approach used to store conflict graphs.

Next, our fast clique detection algorithm explores this matrix: if the number of interconnected data elements in a selected row is lower or equal to $\boldsymbol{P}$, then this means that data is interconnected to less than $(\boldsymbol{P}+\boldsymbol{1})$ data, so the corresponding clique, if it exists, will include less than $(\boldsymbol{P}+\boldsymbol{1})$ data, so *in-place* memory mapping remains possible; on the contrary, if this data is interconnected to at least $(\boldsymbol{P}+\boldsymbol{1})$ data (i.e., at least $(\boldsymbol{P}+\boldsymbol{1})$ of these data are involved in a common clique), this means that additional memory will be needed to keep an *in-place* memory architecture or a *dual-memory* mapping must be used. Since our goal is only to detect the existence of a clique involving at least $(\boldsymbol{P}+\boldsymbol{1})$ data, the FCD algorithm stops. In this case, we know that, in order to target architecture with no additional memory, an in-place memory mapping is not possible.

4.2. Mapping without Relaxation. Mapping without relaxation is to find a valid memory mapping for a given problem based on networks available in the library or on a user selected network. This mapping step is performed through our modified version of [18] algorithm.

The available networks in the library are barrel shifter, *BS*, butterfly, *BF*, and BENES, *BEN*. In case of exploration of all the network of the library, a memory mapping is performed targeting the lower cost network in the library (i.e., *BS* in our case), and then all other network as long as a conflict free memory mapping is possible with no relaxation. Then the lower cost architecture is proposed to the user. If not memory mapping is possible, then no solution can be proposed to the user at this step. It must be noticed that *BS* or *BF* does not guarantee a valid memory mapping for a given problem, but a fully connected *BEN* network can always give a valid memory mapping.

```
Set_perm    // permutations that exists in the network
Map // The matrix of mapping
Col_i;      // The Map column which is selected
Pcol_i;  // Permutation Col_i
Dpcol_i; // One data of Pcol_i
Algorithm NetworkMap (set_Perm, Map, Startup)
Coli = SelectColumn(Map);
PColi = SelectValidPerm(set_Perm, Coli, Map);
If ((PColi is not empty) and not (ALLMAP(MAP))) Then
    Do
      DPColi = Select&RemoveFirstPerm(PColi);
      MapColumn(Coli, DPColi);
      Startup = FALSE;
      NetworkMap (set_Perm, Map, Startup);
      If ((Startup = FALSE) and not(ALLMAP(MAP))) Then
         RemoveColumn(Coli, DPColi);
      End if;
    While ((Startup = FALSE) and (PColi is not empty) and
         not(ALLMAP(MAP)));
    If ((Startup = FALSE) and not(ALLMAP(MAP))) Then
        AddNewNetworkComp(set_Perm);
        EraseMap(Map);
        Startup = TRUE;
        NetworkMap (set_Perm, Map, Startup);
    End if;
End if;
```

Pseudocode 1

4.3. Mapping with Network Relaxation. Mapping with network relaxation gives the liberty to change the network architecture to find a valid memory mapping by adding network components. If mapping is not possible with the selected standard network, certain additional network will be added to the standard network which will result in a valid mapping. The mapping adaptation is given by adaptation the network selected in the beginning with the addition in terms of some elements such as switches and/or multiplexers. This aim to take advantage of [10, 12] for high throughput cost and latency, in such a way that the application used in [15, 18] can be targeted.

For a standard network, the mapping algorithm finds a memory mapping according to the permutation available for the network in the library. If the mapping is not possible for the targeted network an additional network component (e.g., switch) is added to the network. The new interconnection network architecture composed of standard network with additional switch generates new set of permutations. The mapping algorithm is applied based on the new set of permutations. This approach keeps on adding new network component until a collision-free mapping is obtained.

Figure 6 shows the architecture with network relaxation which is composed of original connecting network having adaptation of additional components. As memory adaptation is achieved so optimized architecture can be obtained as compared to [15] or [18], since memory elements are more costly than MUX/switches.

The pseudocode of the proposed approach is given as shown in Pseudocode 1.

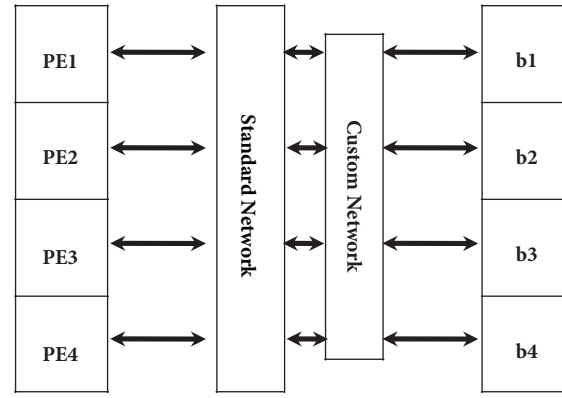


Figure 6: Architecture generated by network relaxation.

5. Pedagogical Example

The proposed approach presented in Section 3 is explained with a small example of size **L**=12 and parallelism **P**=4 from high speed packet access (HSPA) turbo decoders architecture is considered.

The whole design space is explored using all the networks one by one available in the library. The data access pattern is shown in Figure 7.

5.1. Analysis. First of all the clique test is carried out in order to find out whether the collision problem is based on in-place or double memory mapping. As $\boldsymbol{L} = \mathbf{12}$ and $\boldsymbol{P} = \mathbf{4}$, construct

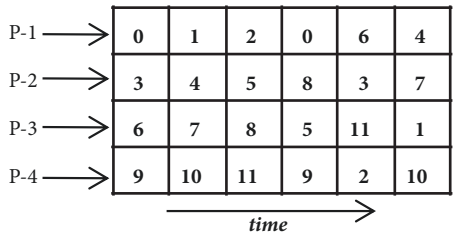


Figure 7: Pedagogical example.

	0	1	2	3	4	5	6	7	8	9	10	11
0	x			x		x	x		x	x		
1		x			x			x			x	
2			x	x		x	x		x			x
3	x		x	x			x			x		x
4		x			x			x			x	
5	x		x			x			x	x		x
6	x		x	x			x			x		x
7		x			x			x			x	
8	x		x			x			x	x		x
9	x			x		x	x		x	x		
10		x			x			x			x	
11			x	x		x	x		x			x

Figure 8: Clique test matrix.

12*x*12 matrix and fill the matrix according to the data access. In Figure 8, the complete matrix is shown filled according to all the access of Figure 7.

Now starting from the first row as it can be seen that 6 data elements are connected with each other which are greater than ***P***, so we will find the number of combinations and determine the clique. After checking all the rows no such clique is found so the problem is based on in-place memory mapping.

The considered example is shown in Figure 9 such that data can be mapped according to the described mapping algorithm.

5.2. Mapping without Relaxation. The mapping algorithm [10] is applied based on all the networks available in the library one by one.

Firstly, the data matrix is mapped based on BS. For considered example collision-free solution is not possible as none of the *BS* permutations are valid for mapping as shown in Figure 10(a). The next available network in the library to find the mapping is *BF*. The whole matrix is mapped successfully for BF network. The resultant mapping is shown in Figure 10(b). Finally, memory is mapped with *BEN*.

The given problem cannot be mapped using *BS* so network and memory relaxation methods can be considered to find collision-free mapping with targeted network.

5.3. Memory Relaxation. The memory relaxation approach adds registers in the architectures to solve the data with collision and memory locations when there is no other solution. Figure 11 shows a complete mapping consist of registers.

5.4. Network Relaxation. Network relaxation gives the liberty to modify the network architecture. As shown in Figure 10(a), for the last column none of the permutation from BS is supported. So an additional network component is added to the standard BS network and new permutations are generated. Continue the mapping process by using the current new set of permutations. The resultant mapping is shown in Figure 12.

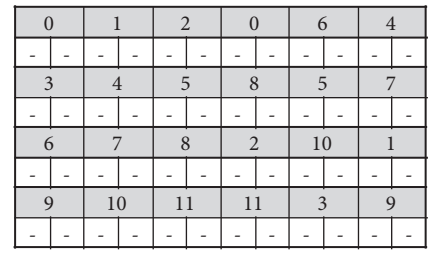
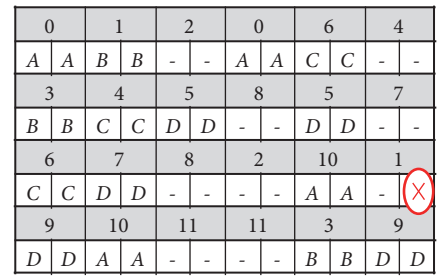
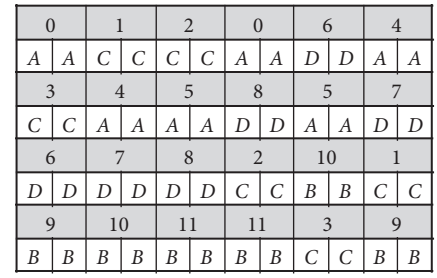


Figure 9: Matrix representation.



(a) Targeting barrel shifter



(b) Targeting BF

Figure 10: Mapping without relaxation.

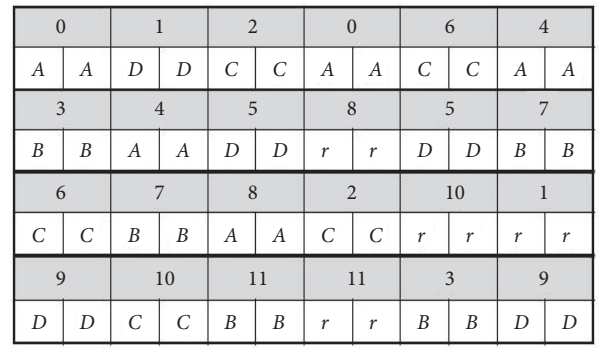


Figure 11: Mapping with memory relaxation.

Table 1: Area and latency comparison.

	Area [15]	Area [12]	Area Proposed NW relax.	Latency [15]	Latency [12]	Latency Proposed NW relax.
L=40	21386	31666	15849	44	20	20
L=120	78786	129246	64639	134	60	60
L=800	690096	1090696	545364	1000	400	400
L=1560	1456386	2574486	1343224	1560	780	780
L=2240	2090656	3696656	1848344	2800	1120	1120

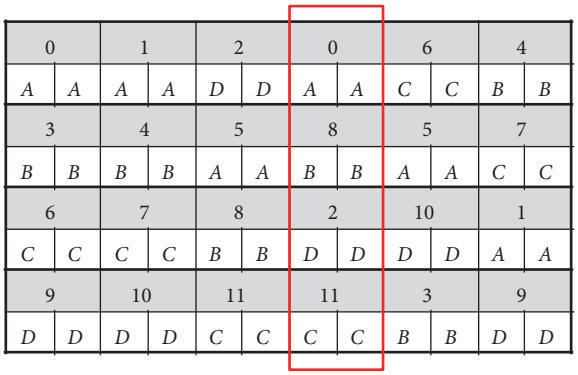
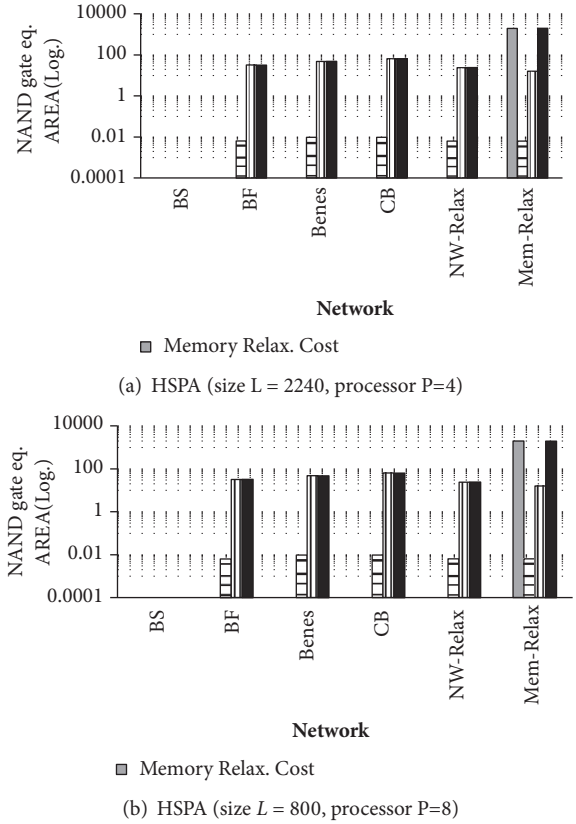


Figure 12: Mapping with network relaxation.



(a) HSPA (size L = 2240, processor P=4)

(b) HSPA (size L = 800, processor P=8)

Figure 13: Cost comparison for different networks.

6. Experiments and Results

The experiments and results performed are presented in this section. STMicroelectronics 90 nanometer technology is used in these experiments to determine the NAND-gate equivalent area of various of the architecture.

This approach can be used for any type interleaver with any number of selected processors for parallelism. We have considered implementation of interleavers that are in use of HSPA Evolution [1].

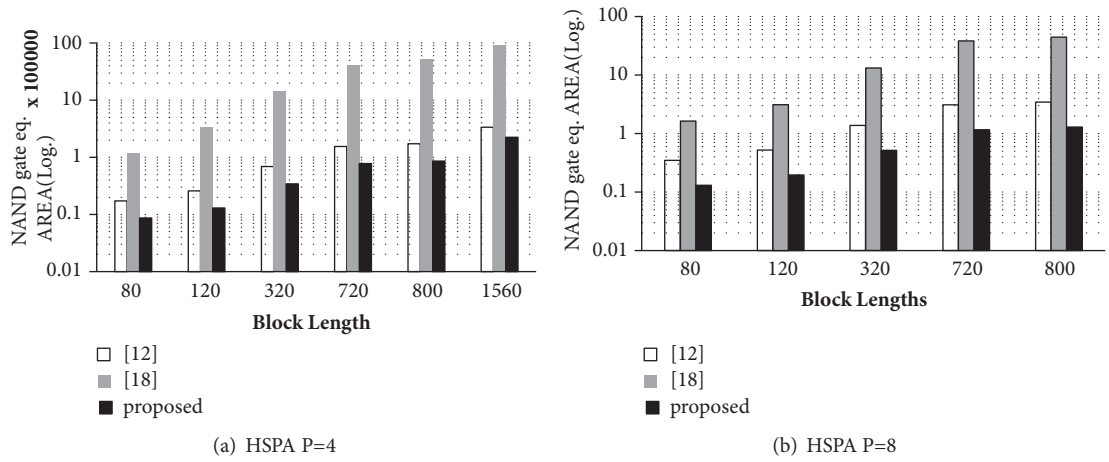
Table 1 shows assessment of the network adaptation approach with different other approaches presented in the literature. The results are shown as comparison of NAND-gate equivalent area and latency in terms of cycles for many HSPA block sizes $\boldsymbol{L}$ with number of processors $\boldsymbol{P} = 4$.

The network adaptation approach has above half of lesser area having similar latency as compared to [12] and 20% area is lesser with reduction of 55% in latency as compared to [15]. Hence, the network adaptation approach can be used to reduce the cost significantly as compared to [12] and it reduces the latency compared to [15].

In order to explore the proposed approach an example of HSPA with Block length of 2240 is considered. Figure 13 shows results for the given problem based on BS, BF, and BENES network excluding the memory cost which is considered same for all the results. Figure 13 also includes results with memory and network relaxation methods.

As the given problem cannot be mapped using *BS*, the cost for memory relaxation using a *BS* is very high as compared to newly proposed network relaxation method. Moreover, we have taken different block sizes using HSPA standard with number of processors $\mathbf{P} = 4$ & $\mathbf{P} = 8$. Figure 14(a) shows the results for four processors and Figure 14(b) shows 8 processors. The result clearly shows that the network adaptation method gives optimal results. Network relaxation for a targeted network can reduce the cost of controller 400 times as compared to existing memory adaptation approach.

In Figure 15, the comparison of the proposed approach is presented with [12, 18, 19]. The results showed significant reduction is cost as compared to these approaches. In [21], the authors proposed a solution for the standard LTE which is based on multistage connecting network of barrel shifter. For 4 processors, two stages of barrel shifter with modification need 3 bits while for $\mathbf{P} = 8$, three stages of modified *BS* need 7 bits. Experiments of the proposed generalized approach are able to find some block lengths that support standard *BS*.



(a) HSPA P=4 (b) HSPA P=8

Figure 14: Cost comparison for different block lengths.

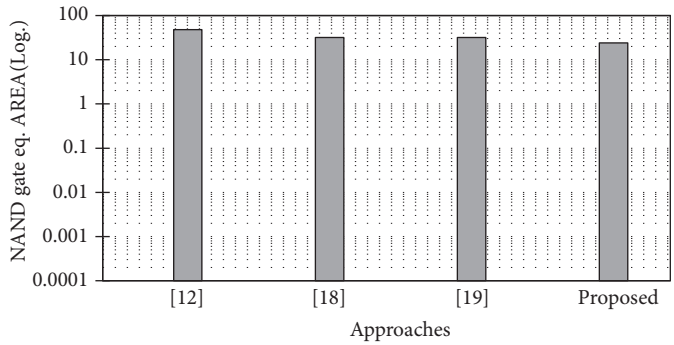


Figure 15: Cost comparison for different block lengths.

Table 2: Block lengths supported by BS.

P=4	160,200,240,320,360,480,528,800,880, 960,1440,1600,2240,2880,3520,4160, 4480,5760,4160
P=8	416,480,800,1600,2240,3520,3584, 4608,4480

The standard barrel shifter uses 2 bits for $\mathbf{P} = 4$ and only 3 bits for $\mathbf{P} = 8$. This will reduce the network controller cost up to 133%. Table 2 shows *LTE* block lengths that support *barrel shifter*.

7. Conclusion

In this work we have presented an approach in which a special technique is used which results in high throughput in terms of latency and complexity. The proposed solution is able to solve the collision problem by the connecting network optimization for achieving high throughput. The result clearly showed that the network adaptation method gives optimal results. Network adaptation for a targeted network can reduce the cost of controller 400 times as compared to existing memory adaptation approach. It can be noted that the network controller cost mostly affects the total cost of the architecture; therefore, in future the controller cost could be reduced to get more optimized results. Furthermore, other targeted networks shall be studied in order to get optimized results.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Science Foundation of China under Grant 61672269 and the Jiangsu Provincial Science and Technology Project under Grant BA2015161.

References

[1] Technical Specification Group Radio Access Network; Evolved Universal Terrestrial Radio Access; Multiplexing and Channel Coding (Release 8), *3GPP Std. TS 36.212*, Dec. 2008.

[2] 3GPP, "Technical specification group radio access network multiplexing and channel coding (FDD)" (25.212 V5.9.0). June 2004.

[3] *DVB DocumentA122*. "Frame structure channel coding and modulation for the second generation digital terrestrial television broadcasting system (DVB-T2)," 2008.

[4] A. Giulietti, L. Van Der Perre, and M. Strum, "Parallel turbo coding interleavers: Avoiding collisions in accesses to storage elements," *IEEE Electronics Letters*, vol. 38, no. 5, pp. 232–234, 2002.

[5] *IEEE P802.11n/D5.02, Part 11*. "Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Enhancements for Higher Throughput", July 2008.

[6] *IEEE P802.16e, Part 16*. "Air Interface for Fixed and Mobile Broadband Wireless Access Systems," Amendment 2: Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands, and Corrigendum 1, Feb. 2006.

[7] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low density parity check codes," *IEEE Electronics Letters*, vol. 32, no. 18, pp. 1645-1646, 1996.

[8] A. I-Gutierez, J.-B. Mourade, S.-P. fletschinger et al., "DAVINCI Non-Binary LDPC codes: Performance and Complexity Assessment," in *Proceedings of the future Network & Mobile Summit, Italy*, June 2010.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near-Shannon limit error-correcting coding and decoding: Turbo codes," in *Proceedings of the IEEE International Conference on Communications*, vol. 2, 1993.

[10] C. Chavet, C. Phillippe, M. Eric et al., "Static Address Generation Easing: a Design Methodology for Parallel Interleaver Architectures," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 1594–1597, Dallas, March 2010.

[11] L. Dinoi and S. Benedetto, "Variable-Size Interleaver Design for Parallel Turbo Decoder Architectures," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1833–1840, 2005.

[12] A. Tarable, S. Benedetto, and G. Montorsi, "Mapping interleaving laws to parallel turbo Low density parity check decoder arch," *IEEE Transactions on Information Theory*, 2004.

[13] A. H. Sani, P. Coussy, and C. Chavet, "A first step toward on-chip memory mapping for parallel turbo and LDPC decoders: a polynomial time mapping algorithm," *IEEE Transactions on Signal Processing*, vol. 61, no. 16, pp. 4127–4140, 2013.

[14] T.-C. Wu, C.-M. Lee, and C.-K. Wang, "Construction of parallelized-decoding LDPC codes," in *Proceedings of the 2016 IEEE International Symposium on Circuits and Systems, ISCAS 2016*, pp. 425–428, Canada, May 2016.

[15] N. When, *SOC-Network for Interleaving in wireless Communications*, MPSOC, 2004.

[16] O. Muller, A. Baghdadi, and M. Jézéquel, "ASIP-based multiprocessor SoC design for simple and double binary turbo decoding," in *Proceedings of the Design, Automation and Test in Europe, DATE'06*, Germany, March 2006.

[17] H. Moussa, A. Baghdadi, and M. Jézéquel, "Binary de Bruijn on-chip network for a flexible multiprocessor LDPC decoder," in *Proceedings of the 45th DAC annual conference*, p. 429, Anaheim, California, June 2008.

[18] Ar. Briki, Ch. Cychile, and C. Philipe, "A Memory Mapping Approach for Network and Controller Optimization in Parallel Interleaver Architectures," in *Proceedings of the 2013 IEEE Workshop on Signal Processing Systems (SiPS)*, 2013.

[19] E. Nieminen, "On quadratic permutation polynomials, turbo codes, and butterfly networks," *Institute of Electrical and Electronics Engineers Transactions on Information Theory*, vol. 63, no. 9, pp. 5793–5801, 2017.

[20] V. E. Benes, *Mathematical theory of connecting networks and telephone traffic*, Academic Press, New York, NY, USA, 1965.

[21] C.-C. Wong and H.-C. Chang, "Reconfigurable turbo decoder with parallel architecture for 3GPP LTE system," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 7, pp. 566–570, 2010.