FPGA Implementation of Network Coding Decoder

Taeyoon Yoon[†] and Joonseok Park[††],

[†]Samsung Electronics Inc., Suwon, Korea
[††]Department of Computer and Information Engineering, Inha University, Inchon, Korea

Summary
Network Coding enhances performance of multi-cast network system, e.g., P2P(Peer-to-Peer) system. The Receivers in the network that use network coding technique have to decode the received messages which are encoded by intermediate nodes as well as sender node. Hence, decoding operations of network coding system are more complex than those of normal network system; therefore, performance of decoding operations has a significant influence on performance of the entire network system. In this paper, we address the issues regarding implementation of FPGA-based acceleration engines of Gaussian eliminations on finite Galois field. We compare the decoding performance on FPGAs with SW based implementations on P-4 processor and ARM embedded processor. The result shows that FPGA implementation outperforms contemporary widespread P-4 processor and ARM processors by 20 and 140 times, respectively. Utilizing the technique introduced in this paper, an efficient network coding system could be realized with a low level of computing resources.

Key words:
Network Coding, CodeCast, FPGA, Gaussian elimination.

1. Introduction

There has been rapid improvement in the various types of networks, including traditional wired, wireless, and AD-HOC network. There have been modern network technology developments in the area of network, such as hardware devices to perform communication software, communication protocols, etc. Those developments are associated with the diverse spread of data streaming and multimedia applications and services, for example, P2P, VoIP.

These applications of the existing server - client model, would be more suitable in uni-cast, multicast network than conventional broadcast based communications. For broadcast networks, Edmonds-Karp algorithm is known as a routing scheme to obtain maximum transfer efficiency [1]. However, to find the maximum network transmission efficiency in a multicasting routing scheme is known as NP-hard [4]. Therefore, it is necessary to find a new type of routing scheme for multicast network and network coding is recently proposed as a solution for this.

The major difference of network coding from normal routing network is the role of intermediate nodes. When transmitting data, the node in the network coding encodes the data and decodes in the receiving node. However, intermediate nodes in the routing scheme, the received data should be passed without any conversion. Therefore, encoding transferring data is required in intermediate nodes, as well as in a sending node.

Thus, for the implementation of network coding techniques require encoding operations in originating node and intermediate node, and decoding operations destination node, respectively. The processing power for this operation has a significant impact on the performance of overall network system, because the encoding and decoding time reflects the whole data transfer time in the system.

For example, if the transmission time using network coding is faster by time, let say s, than normal data transfer without network coding, and the overhead to encode/decode the transferred data in the intermediate and destination node is bigger than s, the benefits of network coding become negligible. Sending node and intermediate nodes required in the arithmetic coding operations require small and simple computations, however, the decryption operations performed at the destination node, require a considerable amount of complex operations. Therefore, for efficient decoding operations research is needed.

In this paper, network coding techniques for performing fast computation of the decoding method is proposed. Especially the advantages of application specific system such as ASIC (Application Specific Integrated Circuit) or o reconfigurable hardware, FPGA (Field Programmable Gate Arrays), are addressed. Implementation of specialized on-demand system, the results shows that significant improvement in the performance over the general-purpose systems.

This paper is organized as follows. Chapter 2 introduces the related research and Section 3 describes each of the network coding and coding and decoding operations are explained in Section 4 describes the FPGA implementation. And the analysis of experimental results in Section 5 and Chapter 6 describes the conclusion.

2. Related Researches

Since the multicast network coding algorithm based on coding theory is introduced there have been many

Manuscript received December 5, 2010
Manuscript revised December 20, 2010

researches [3]. Based on information theory, network coding can achieve maximum throughput when it uses the data arrived from different paths and again encode the data in the intermediate node. However, the phase information to figure out the optimal solution via identification and coding of the nodes, etc., is yet unknown. Random Linear Coding is propose, as an opportunity to develop various protocols using network coding such as bit torrent (Bit Torrent) system and a form of P2P system AD-HOC networks.[5] [6] [7]. A study on the performance of network coding has been active recently.

There have been some researches to accelerate network protocol processing using FPGAs. In [2] they used FPGAs, for public-key cryptographic protocols using Galois field to achieve high-speed encryption and decryption process. The calculation of a finite field exponent of the encryption decryption process has been presented. Parallel Gaussian elimination using the FPGA in GF(2) (Galois Field (2)) to improve the performance of the crypto system was introduced in [11]. However, their study of GF (2) is only limited to operations in the logic to implementation of Gaussian elimination, therefore it cannot be applied to extended binary field (GF(2n)). In [12] they presented floating-point operations for Gaussian elimination, utilizing the FPGA and GPU acceleration technology.

The contribution of this paper, differs from earlier studies, in that we improved performance by utilizing GF (28) lookup table in the FPGA to accelerate finite field operations. We exploit large amount of parallelism both in computations and take advantage of increased internal bandwidth of FPGAs.

3. Network Coding

3.1 Network coding theory

Fig. 1 is a brief comparison of multicast schemes of the traditional forwarding and network coding. Upper graph shows the multicast using the scheme of data forwarding. In the traditional multicast using forwarding, data should be transferred in packet by packet. As shown in the figure if source node S sends data d1, d2 to the destination nodes, R1, R2, it should transfer d1 and d2 one at a time. Intermediate nodes, K, L, M and N just repeat the data it receives. However, in the figure in down S can send d1, and d2 to different intermediate nodes, K, and L. Then the node M, which receives d1 and d2, can generate new message with input data. For example, bit-wise and operation can make new data block d1+d2. The newly generated data d1+d2 will be transferred to R1 and R2 through another intermediate node N. In the destination nodes, R1, which receives d1, from K, can reconstruct original data d2 with received d1 and d1+d2. In the same manner R2 can reconstruct d1 from received data, d2 and d1+d2.

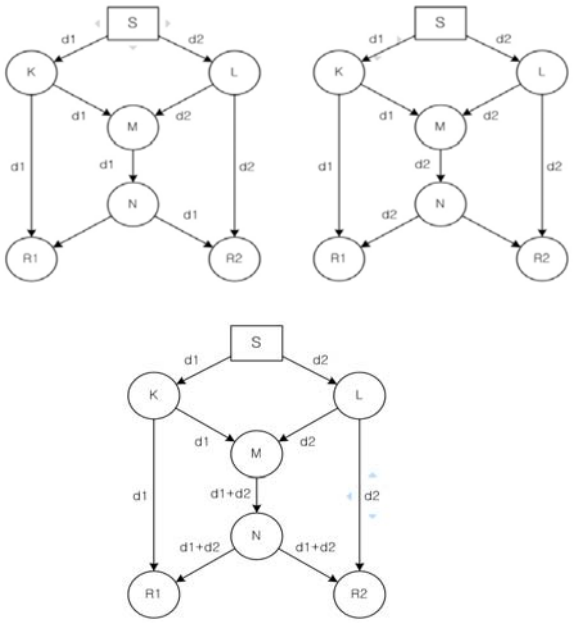

Fig. 1 Multi-cast using forwarding (up) and network coding (down).

In network coding, bit-wise and operation which is triggered in intermediate node M is called encoding. The process which reconstructs original data using received data is called decoding. As explained using Fig1., in the forwarding multi-cast, it requires two stages to send d1 and d2 to the destination nodes, R1 and R2. However, in the network coding multi-cast process will be completed within single phase. In this scenario, data will be transferred more efficiently using network coding than normal forwarding.

To achieve this, sending node partition the original data into block, encode each block and transfer encoded block to the network system. The partitioned data used in the same encoding operation is called as “generation”. All the encoding operations are defined in the finite field operation.

3.2 Encoding

Randomly generated coefficient vectors encoding operation and the operation of the data is divided by the block vectors. K is the size of one block, when the block of n (where n is called the generation size) block [B(1),..., B(n)] can be represented by, the coefficient vector [e (1), ... , e (n)] . In this case, the encoded block is calculated by the following formula. (See equation (1))

$$P(i,j)=\sum_{i=1}^{n} e(j,i)\cdot B(i,j) \qquad (j=1,\cdots,k) \quad (1)$$

In P (j, i), j is the block index and i is the field index within single block. All coefficients which are encoded using arithmetic combination of Finite Galois Field) are randomly selected. It is the mostly beneficial when the sets of each coefficient are mutually independent vectors. When we produce a mutually independent vector coefficient, the minimum of the reconstruction process, at least in the originating node receives data only on the transmitted data can still be obtained. However, it is almost impossible to find the optimal set of coefficients within the ad-hoc network, by identifying the relationship of all participating nodes. Therefore, instead of finding the optimal coefficient vectors with the high cost, using a random coefficient vector by choosing an arbitrary value of the coefficients is more desirable. By using random coefficient, it is possible to decode with high probability [5].

In the encoding process, coefficients are passed along with generated coded block data for decoding process. From the expression below, equation (2), a sending node (or intermediate nodes) in the coding process is presented. Expression under a randomly selected for each factor e, b actually means to transmit data, and n is the number of blocks, k is the block size.

$$\begin{pmatrix} e_{1,1} & \cdots & e_{1,n} \\ \vdots & \ddots & \vdots \\ e_{n,1} & \cdots & e_{n,n} \end{pmatrix} \cdot \begin{pmatrix} b_{1,1} & \cdots & b_{1,k} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,k} \end{pmatrix} = \begin{pmatrix} p_{1,1} & \cdots & p_{1,k} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,k} \end{pmatrix} \quad (2)$$

The data set which will be transferred for multi-cast is partition of above formula. Coded vector set P is transferred with the coding vector e. Hence, each line of 2 dimensional matrixes of e and p will be transferred. To paraphrase, one block of data is consist of the coefficient vector e, whose dimension is n, and coded data vectors, whose size is k. Therefore, n + k-size vector is transmitted.

3.3 Decoding

After the destination node receives more than n number of transport blocks, the decryption operation starts. The received data block contains coefficient vector. The received n vectors forms coefficient matrix. There should be enough coefficient vectors to build inverse coefficient matrix using received vectors. In other words, the independent relationship among the coefficient vector should be met.

In the actual implementation of the system, the received block count should be big enough to guarantee the indecencies to build n by n inverse matrix during decoding process. The process of decoding is captured in the following equation (3).

$$\begin{pmatrix} p_{1,1} & \cdots & p_{1,n} \\ \vdots & \ddots & \vdots \\ p_{n,1} & \cdots & p_{n,n} \end{pmatrix}^{-1} \cdot \begin{pmatrix} p_{1,n+1} & \cdots & p_{1,n+k} \\ \vdots & \ddots & \vdots \\ p_{n,n+1} & \cdots & p_{n,n+k} \end{pmatrix} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,k} \\ \vdots & \ddots & \vdots \\ b_{n,1} & \cdots & b_{n,k} \end{pmatrix} \quad (3)$$

From the node from which to send the data block, [b(1), ... b(n)] will be depicted after the decoding process. To accomplish this, the linear operation to solve equations, this is known as Gaussian elimination.

4. FPGA implementation

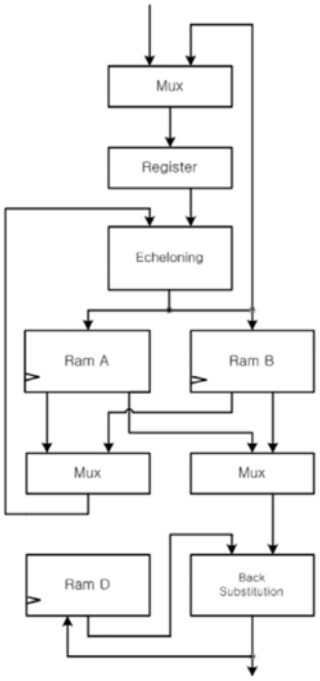

Fig. 2 Block diagram of decoding process

As introduced earlier, the actual implementation of network coding for data transmission to the general network protocols required in addition to creation of a packet originating node (or intermediate node) and destination node. The decoding operation, unlike encoding operation finite field operation, Gaussian elimination, should be used. Therefore decoding process requires a lot of computational capacity. Specialized and parallelized computations for fast decoding operation are implemented in FPGA devices to improve the performance of network coding system.

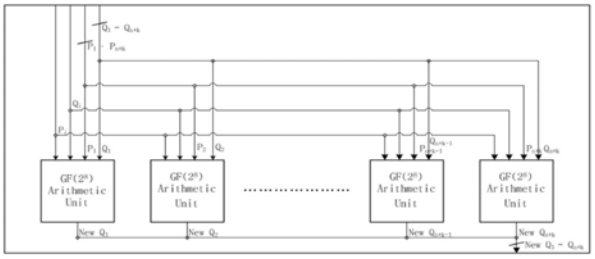

Fig. 3 Internal Structure of Echeloning

Fig. 3 is a block diagram of to perform the decryption. This logic is configured to encode data, coefficients and coded block of a sequential type has a structure. To complete the encoding of the data, decoding operation takes place as many blocks of units are needed. Used to encode the coefficient vector is not independent of each other's creation can be inverted, so the mutually independent n-type should be coded blocks of data can be decrypted. In this experiment, the field size and block size was fixed at 4, and the size of each generation is fixed at 8 byte. The implementation of FPGA logic, block number of each operation is same as the generation size which is 4. One is composed of 8 byte. We assumed that data input interval is 4 cycles and all block data is all independent to each other.

Encoding and decoding all the finite fields to perform the operations defined in the finite field can be divided into two folds - prime fields and binary extension fields. Binary extension field, whose numeric representation is the array of binary values of 0 and 1, it can handle more favorable than that of prime field in dealing with digital equipment. In this experiment, 8-bit binary extension field ($GF(2^8)$) were used. In general, the multiplication in the binary extension field is a set of shift and xor. However, the sequentia implementation to access each bit by shift and xor is faster than arithmetic lookup table. The look up table using the index to log table to table, $GF(2^8)$ is a field operation in the table, is 256 bytes long.

Decoding process can be divided into two phases. As a first step of Gaussian elimination, contrary to the coefficient matrix, is the process of making triangular matrix. The next step is the process of calculating the actual data. Arithmetic operations to perform the first process (Fig. 2) are implemented in the Echeloning logic, the second process, the logic of the arithmetic operations are performed in the back substitution. In Echeloning logic, the destination node decodes the transmitted data block. At this phase, Elements belongs to the same block, same operation is performed. The formula for this is shown as follows.

$$a \bullet \left(P_1^1 \cdots P_{1,n+k}^1\right) - \left(P_1^2 \cdots P_{1,n+k}^2\right) \quad (4)$$

Using these characteristics all elements that belong to a block are computed in parallel. Fig.4 is the block diagram that shows parallelized operations in Echeloning logic block. P, Q, and the index i is determined by the control logic. In Echeloning block of logic operation is performed repeatedly and the corresponding results will be stored into Ram_A and Ram_B alternately block by block, just like the corner turn operation which can exploit maximum memory bandwidth. Both Ram_A and Ram_B will be implemented using internal ram block of FPGA logics. To store the same data we implemented redundant logic to improve the performance of the system using pipelined operations. We applied two stage pipelines – the first stage is echeloning and the second one is Back Substitution. By its nature of pipelining, the overall bandwidth become two times of maximum single phase throughput.

Echeloning logic is the logic to create the superior triangular matrix coefficients and the decoded data values are transmitted to Back Substitution to calculate the original data which the originating node generated. Back Substitution, just like Echeloning logic, all computational elements were implemented in parallel, therefore all data belonged to the same block is performed at the same time. All blocks belonging to generation in order to be decoded belongs to the generation of all the blocks of data are needed. Therefore, after a complete decoding of the block, although, as a result of the decoded output is processed as soon as the block that was designed to save again.

5. Experimental Results

Decoding algorithm and its FPGA implementation is introduced in the previous section. In this chapter, the actual decryption operation on general-purpose systems or embedded systems performance and comparative analysis is described.

5.1 Experimental Environment

Decoding operations of destination node are implemented in the FPGA device. First, the algorithm is described in hardware description language VHDL (Very High-level Hardware Description Language), and then was it is synthesized by Xilinx ® ISE™ 9.1. For the target device Xilinx® Virtex-4 XC4VLX60 used. MGC (Mentor Graphics Co.®) ModelSim 6.5 PE™ simulations were carried out u, it was confirmed that work on the actual FPGA device.

Decryption operations compared to the FPGA implementation results for the performance of a program that performs the same function as C/C++ code was written. The network coding can be applied across multiple networks, wired or wireless network by

considering the situation was running a regular desktop computer. In special circumstances, such as sensor networks and networking, embedded systems are made by considering the circumstances of the processor are widely used as was done in a few ARM-based processors. Experiments on a desktop computer with operating system Linux (kernel-2.6.25) is actually running on the computer. For embedded processor, simulation (Armulator) for Intel ® PXA255, PXA272 processor running on embedded Linux system was equipped with. Table 1 shows the summary of the platforms that SW is actually running on.

Table 1: The Experimental Frameworks.

	Intel® Core2™ Duo E6550	ARM® 926EJ-S	Intel XScale™ PXA255	Intel XScale™ PXA272
Processor	2.33Ghz	276MHz	100~ 400MHz	104~624MHz
Right	32K/32K(L1) 4M (L2)	8K / 8K	32K / 32K	32K / 32K
Measure	Wall-time measure	armulato r	Wall-time measure	Wall-time measure

The ARM926EJ-S processor is manufactured in the 0.13μm process, and [9] are refered to identify speed and cache size of each processor. In the three types of embedded system environment was conducted, all of them are widely used embedded processors. The experiment is performed in a real embedded system environment and the experiments were performed using the simulator.

In the experiment using the simulator, the processor's memory access time to the math equation shown below was. Calculation and memory access time, system bus speed of 100MHz, Micron ® 48LC64M4A2 SDRAM (256 MBit, 100MHz) was assumed.[8]

5.2 Experimental results

Table2. shows the FPGA implementation results of decoding operations. Each of the system in order to measure the speed of the decryption processing of data required for 32byte measure execution time compared to the performance. Run time and recovery time was measured from the point input data is encrypted to the time after the arrival in the destination node through a process of decoding the original data. In each platform of Table 3, the time taken to perform the decryption operation is shown. Network coding for the purpose of decryption using the FPGA outperforms the contemporary P-4 processor (Intel ® E6550) in desktop PC, by about 22 times. It outperforms the embedded ARM926EJ-S processor, about 150 times, PXA255 about 520 times more, PXA272 approximately 300 times.

Table 2: The Synthesis results of FPGA design

Target Device	Xilinx®XC4VLX60
Slices Flip Flop	1,675/53,248 (3%)
4 input LUT	19,583/53,248 (36%)
Max. Frequency	50.7 (MHz)
equivalent gate count	166,080

FPGA-based implementation of decoding shows a very significant performance improvement. Compared with SW version that performs Gaussian elimination sequentially. Performance achieved by FPGA can be seen as two folds of sources. First, eight blocks of data in is computed in parallel by hardware configurations. Second, the pipelined execution of the two independent steps - enchloning and back substitution, which ultimately achieves twice of original throughput.

Table 3: The performance comparison

platforms	(micro sec)	Performance (base : Intel P4)
FPGA (Xilinx®XC4VLX60)	0.32	x 22.2
Intel® E6550	7.1	x 1.0
ARM926EJ-S	48.1	x 0.15
Intel® PXA255	166.8	x 0.04
Intel® PXA272	96.3	x 0.07

5.3 Analysis and Future Research

In this paper, we have shown that FPGA design of the decoding operations in network coding is adequate to handle in hardware. To build the FPGA design, the number and size of the block, the generation of block size should be fixed at the FPGA configuration time. The each block should be independent to correctly encode and decode, as well. If the generation of a fixed block size generally assumed in the two SW-based systems can be low, but compared to the number, size, use only the limited hardware resources in the transport system is very useful.

According to the Earlier studies, SW-based network coding techniques for the finite size and, depending on the

size of the block and the generation of the performance of network coding system was changed.

In this study, we used look-up table for field operations in FPGAs. In this context, if the size of finite field gets bigger, the more logic is required to implement fintite field operations in FPGA. However, the field size increases, the same amount of data needed to be reduced in proportion to block size, and apply Gaussian elimination to reduce the size of the matrix and the operation of the FPGA can be less based on logic can be. Therefore, in order to optimize the implementation, it is necessary to adjust field and block size to implement decoding logics into FPGAs, which only has limited resource. The matrix size of the FPGA at run-time effects, because parallel processing of a single increase in proportion to the size of column size unit. The exact analysis is going to continue through future research.

6. Conclusion

In this paper, decoding operation of network coding has been implemented on the FPGA, which network can significantly affect the on the performance of the network coding system. The results of SW implementations in general-purpose and embedded computing systems are compared with FPGA-based hardware acceleration system. Implementation in FPGA is mainly focusing on Galois Finite Field operations. Our implementation exploit table loop up method to parallelize row by row operations of Gaussian elimination, and addressed pipelined operation to improve performance. The results of network coding and decoding operations processing software to hardware to handle compared to 22 times in the general purpose computing system and improvement of up to 150 times of wide-spread contemporary embedded systems.

References

[1] J. Edmonds and R. M. Karp, "Theoretical improvements in algorithmic efficiency for network flow problems," Journal of the Association for Computing Machinery. Vol. 19, No.2, pp.248-264, Apr. 1972

[2] G. Orlando and C. Paar, “A super-serial Galois fields multiplier for FPGAs and its application to public-key algorithms,” Proc. of the IEEE Symposium on Field-Programmable Gate Array. 1999

[3] R. Ahlswede, N. Cai, S. R. Li, and R. W. Yeung, "Network information flow," IEEE Transactions on Information Theory. Vol.46, Issue.4, pp.1204-1216, 2000.

[4] K. Jain, M. Mahdian and M. R. Salavatipour, "Packing Steiner trees," Proc. of 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp.266-274, 2003.

[5] P. Chou, Y. Wu, and K. Jain, “Practical Network Coding,”Proc. of Allerton Conference on Communication, Control, and Computing, October 2003.

[6] C. Gkantsidis and P. Rodriguez, “Network Coding for Large Scale Content Distribution,” Proc. of IEEE INFOCOM 2005, March 2005.

[7] J.S. Park, D. S. Lun, Y. Yi and M. Gerla, “CodeCast: a network coding based ad hoc multicast protocol,” in IEEE Wireless Communications, vol. 13, no. 5, pp. 76–81, 2006

[8] B. Jacob, S. W. Ng, D. T. Wang, Memory Systems : Cache, DRAM, Disk, Morgan Kaufmann, 2007

[9] ARM9EJ-S Technical Reference Manual. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.set.arm9/index.html

[10] A.Bogdanov and M. C.Mertens, "A Parallel Hardware Architecture for fast Gaussian Elimination over GF(2),".Proc. of 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines(FCCM). pp.237-248, 2006

[11] S. Che, J. Li, J.W. Sheaffer,K. Skadron and J. Lach, "Accelerating Compute-Intensive Applications with GPUs and FPGAs," Proc. of IEEE Symposium on Application Specific Processors(SASP). pp.101–107, 2008.

Taeyoon Yoon received the B.S. in Geometric Information engineering and M.S. degree in Computer Engineering from Inha University in 2007 and 2010, respectively. He is now with Samsung Electronics Inc. His research interests are in the area of embedded system and HW/SW co-design.

Joonseok Park received the B.S.in Mathematics from Sogang University in 1997, M.S. degrees and Ph.D in Computer Science from University of Southern California in 2000 and 2004, respectively. During 2004-2006, he stayed in SoC Lab of Samsung Electronics, Inc. as a senior Engineer. He is an assistant professor in Inha University, School of Computer and information Engineering