JKIICE Journal of the Korea Institute of Information and Communication Engineering

한국정보통신학회논문지(J. Korea Inst. Inf. Commun. Eng.) Vol. 19, No. 2 : 386~392 Feb. 2015

영상압축코덱을 위한 효율적인 스캔변환기 설계

이건중[1] · 류광기[2*]

A Design of Efficient Scan Converter for Image Compression CODEC

Gunjoong Lee[1] · Kwangki Ryoo[2*]

[1]Research Institute, HT, Inc, Seoul 137-927, Korea

[2*]Department of Information and Communication Engineering, Hanbat National University, Daejeon 305-719, Korea

요 약

영상압축코덱은 데이터를 일정한 블록크기로 나누어 처리하며 블록크기로 나누어진 데이터는 필요에 따라 처리 순서가 바뀌게 되므로 블록단위의 순서 재배열을 위해서는 블록의 크기에 해당하는 데이터를 메모리에 저장한 후 새로운 순서로 읽는다. 처리 속도를 유지하기 위해서는 두 개의 메모리를 이용하여 입력 데이터를 저장하는 동시에 이전에 저장된 데이터를 읽는 방법을 사용한다. 본 논문에서는 단일 메모리를 적용한 불규칙한 입출력 순서 변환의 경우에도 주소의 변화가 유한한 갱신 횟수 안에 반복되는 예측 가능한 규칙성을 가짐을 보이고 하드웨어 구현을 위한 효율적인 방법을 제안한다. 제시한 방법은 HDL로 설계하여 TSMC 0.18 CMOS 공정 라이브러리를 이용하여 합성하였고 다양한 입출력 순서변환 스캔블록에 대해 40%이상의 면적 절감효과가 있음을 확인하였다.

ABSTRACT

Data in a image compression codec are processed with a specific regular block size. The processing order of block sized data is changed in specific function blocks and the data is packed in memory and read by a new sequence. To maintain a regular throughput rate, double buffering is normally used that interleaving two block sized memory to do concurrent read and write operations. Single buffering using only one block sized memory can be adopted to the simple data reordering, but when a complicate reordering occurs, irregular address changes prohibit from implementing adequate address generating for single buffering. This paper shows that there is a predictable and recurring regularity of changing address access orders within a finite updating counts and suggests an effective method to implement. The data reordering function using suggested idea is designed with HDL and implemented with TSMC 0.18 CMOS process library. In various scan blocks, it shows more than 40% size reduction compared with a conventional method.

키워드 : 영상코덱, 스캔, JPEG, MPEG, HEVC

Key word : Image Codec, scan, JPEG, MPEG, HEVC

접수일자 : 2014. 12. 01 심사완료일자 : 2014. 12. 26 게재확정일자 : 2015. 01. 12

*** Corresponding Author** Kwangki Ryoo(E-mail:kkryoo@gmail.com, Tel:+82-42-821-1710)
Department of Information and Communication Engineering, Hanbat National University, Daejeon 305-719, Korea

Open Access http://dx.doi.org/10.6109/jkiice.2015.19.2.386 print ISSN: 2234-4772 online ISSN: 2288-4165

Ⅰ. 서 론

영상신호처리는 공간적 신호를 처리하는 분야로 화질 개선이나 압축, 영상 인식 등의 효율적 처리를 위해 인접한 데이터를 블록 단위로 나누어 처리 한다. 블록 단위로 나누어진 데이터는 필요에 따라 서로 다른 기능을 하는 연속적인 신호처리 과정을 거친다. 이 과정에서 블록 내의 데이터는 일반적으로 좌에서 우, 그리고 위에서 아래로 진행되는 기본적인 순서에 따라 처리된다. 그 중 영상압축분야는 대량의 데이터를 일정한 속도로 유지하면서 실시간으로 처리를 해야 하는 특성을 갖고 있다. 또한 저전력 동작을 위해 가능한 낮은 주파수에서 동작할 수 있도록 하드웨어를 구현해야만 한다.

영상압축과정에서 데이터의 처리 순서를 변환하기 위해서는 처리하고자 하는 블록의 데이터를 블록 크기의 메모리에 저장한 후 다시 읽어 내는 과정을 거쳐야 한다. 실시간 영상신호의 특성상 연속적으로 새롭게 입력되고 있는 데이터의 손실을 막기 위해 별도의 버퍼 메모리를 사용한다. 그림 1은 그와 같은 경우 데이터 손실을 막기 위해 처리하고자 하는 블록의 크기와 동일한 크기의 메모리 2개를 사용하여 서로 순서를 바꿔가며 입출력 동작을 수행하는 이중 버퍼링 방식 보여준다[1].

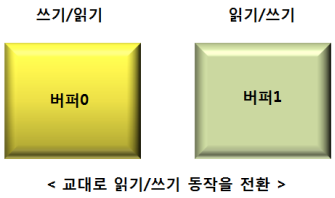

그림 1. 이중 버퍼링을 이용한 입출력 순서변환
Fig. 1 Input/output ordering using dual buffer

이러한 과정의 두 가지 예로 영상압축에서 공간적 영역의 데이터를 주파수 영역으로 바꾸는 이차원 변환의 데이터 전치블록과, 엔트로피 코딩을 위해 중요도에 따른 순서로 데이터를 재배열하는 지그재그 스캔 과정을 들 수 있다.

첫 번째 예로 이차원 변환과정의 데이터 전치과정은 그림 2에서와 같이 데이터 블록의 가로와 세로 방향을 바꿔서 순서를 재배열한다[2]. 재배열 전후의 순서가 규칙적이고 간단하기 때문에 단순한 주소 비트의 재배열만으로도 구현이 가능하다.

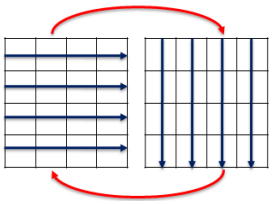

그림 2. 이차원 변환과정의 데이터 전치
Fig. 2 Data transposition of 2-dimensional processing

두 번째 예는 엔트로피 코딩을 위한 중요도 순의 데이터 재배열이다[3]. 변환과정과 양자화를 거친 영상 데이터는 엔트로피 코딩을 거치기에 앞서 중요도가 높은 순으로 재배열을 거친다. 지그재그 스캔이 대표적인 예로 변환을 거친 데이터는 공간적으로 좌측 상단 방향으로 눈에 민감한 저주파 데이터가 모이는 점을 고려하여 데이터를 재배열 한다.

본 논문에서는 실시간 영상압축코덱에서 작은 면적으로 구현 가능한 임의의 불규칙한 블록 데이터의 순서 변환방법을 제시한다. 제시한 방법은 HDL을 이용하여 구현하였으며 이전 방법과 비교하여 그 효용성을 검증하였다.

Ⅱ. 본 론

2.1. 단일 메모리를 이용한 데이터 입출력 순서 변환의 기본 개념 및 문제점

그림 3에서 볼 수 있듯이 동시에 이루어지는 입출력 데이터 간의 자리바꿈을 이용하는 것이 기본 개념이다. 입출력 순서가 규칙적이고 간단한 경우 입출력 순서열은 규칙성을 갖고 반복적으로 이루어지기 때문에 용이하게 구현할 수 있다. 이 경우 처리하고자 하는 블록 크기의 메모리 한 개만 있으면 되므로 작은 면적의 구현이 가능하다[4]. 반면 불규칙한 입출력 순서의 변환이 일어나는 경우 단일 메모리를 방법을 적용하면 블록의 데이터를 한번 처리할 때마다 접근해야 하는 주소열의 순서가 새로운 불규칙한 순서열로 바뀌기 때문에 적절한 주소발생기를 구현하기가 어렵다.

그림 4와 표 1은 8x8 지그재그 스캔에서 단일 메모리를 사용하는 방법을 사용한 예이다. 첫 번째 블록데이터는 비어 있는 메모리에 저장되기 때문에 0, 1, 2, 3,

…의 순서대로 주소접근이 이루어진다.

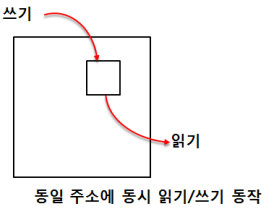

그림 3. 단일메모리를 이용한 입출력 순서변환
Fig. 3 read/write sequence of single memory

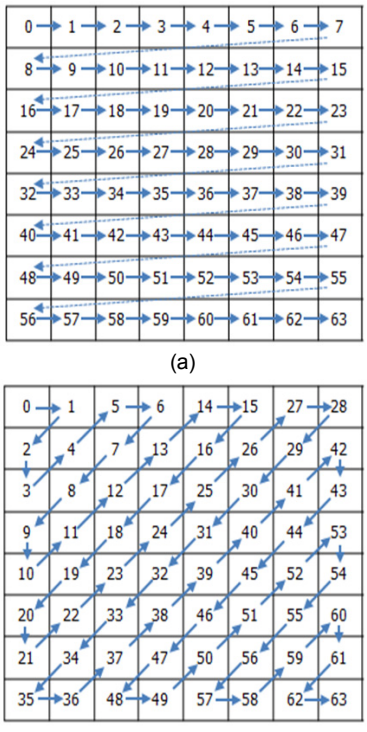

그림 4. 지그재그 8x8 스캔의 (a) 입력순서 (b) 출력순서
Fig. 4 (a) Input (b) output sequence of zigzag 8x8 scan

그 다음 블록 데이터가 입력될 때의 메모리 주소접근은 앞서 기록된 순서인 0, 1, 2, 3, …을 지그재그 순서로 변환하여 이루어지기 때문에, 0, 1, 17, 12, …의 순서로 주소가 접근된다. 두 번째 갱신은 앞서의 0, 1, 17, 12, …를 다시 지그재그 순서로 재접근하기 때문에 0, 1, 19, 18, …의 순서열이 나오게 된다. 이처럼 단일 메모리를 사용할 경우 기본 입출력 순서는 고정되어 있어도 실제 접근하는 주소는 블록데이터를 한번 처리할 때 마다 새로운 주소열 패턴으로 나타남을 알 수 있다.

표 1. 단일메모리를 적용시킨 8x8 지그재그 스캔 입출력순서 변환의 주소열 변화패턴
Table. 1 Address pattern of input/output sequence change in 8x8 zigzag scan using single memory

갱신 횟수	주소								…						
0	0	1	2	3	4	5	6	7	…	58	59	60	61	62	63
1	0	1	8	16	9	2	3	10	…	61	54	47	55	62	63
2	0	1	17	12	24	8	16	32	…	55	39	51	46	62	63
3	0	1	19	18	27	17	12	35	…	46	36	45	44	62	63
4	0	1	33	26	6	19	18	56	…	44	57	37	30	62	63
5	0	1	42	13	3	33	26	53	…	30	60	50	21	62	63
6	0	1	15	11	16	42	13	31	…	21	47	52	48	62	63
7	0	1	5	25	12	15	11	28	…	48	51	38	58	62	63
8	0	1	2	20	18	5	25	7	…	58	45	43	61	62	63
9	0	1	8	40	26	2	20	10	…	61	37	23	55	62	63
10	0	1	17	29	13	8	40	32	…	55	50	34	46	62	63

2.2. 이전 연구에서의 처리방법 및 제약

불규칙하고 매번 새롭게 갱신되는 주소를 효율적으로 발생시키기 위해 데이터의 입력순서와 출력순서간의 상대적 관계는 변하지 않는 점을 이용한다[5]. 블록처리 시, 새롭게 정해지는 순서를 별도의 주소 저장공간에 순서대로 기억시키고, 그 주소저장 공간의 주소를 이용하면 고정된 입출력 순서의 상대적 관계를 이용하여 출력 순서를 지정할 수 있다.

이 방법은 입출력 순서간의 변환형태가 일정한 이상 어떤 임의의 순서열에도 대응할 수 있는 장점이 있다. 하지만, 처리해야 하는 블록의 크기가 커질수록 주소를 저장해야 하는 공간의 크기도 상대적으로 커져 면적을 줄이는 효용성이 떨어진다. 표 2는 처리해야 하는 블록의 크기에 따른 저장공간의 크기를 나타낸다.

표 2. 블록의 크기에 따른 주소 저장공간의 크기
Table. 2 Memory space according to the block size

블록크기	버퍼크기	주소저장공간 크기	버퍼대비 크기
4x4	12x16	12x4	4/16 = 0.25
8x8	12x64	12x6	6/12 = 0.5
16x16	12x256	12x8	8/12 = 0.67

표 3. 8x8 지그재그 스캔에서 처리순서에 따른 물리적 주소의 변화패턴
Table. 3 Patterns of physical address sequence in 8x8 zigzag scan process using single memory

갱신횟수	주소																													
0	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	0	1	8	16	9	2	3	10	17	24	32	25	18	11	4	5	12	19	26	33	40	48	41	34	27	20	13	6	7	14	21	28
2	0	1	17	12	24	8	16	32	19	27	35	20	26	25	9	2	18	33	13	42	29	58	22	49	6	40	11	3	10	4	48	7
3	0	1	19	18	27	17	12	35	33	6	56	40	13	20	24	8	26	42	11	15	14	61	41	59	3	29	25	16	32	9	58	10
4	0	1	33	26	6	19	18	56	42	3	53	29	11	40	27	17	13	15	25	5	4	55	22	54	16	14	20	12	35	24	61	32
5	0	1	42	13	3	33	26	53	15	16	31	14	25	29	6	19	11	5	20	2	9	46	41	39	12	4	40	18	56	27	55	35
6	0	1	15	11	16	42	13	31	5	12	28	4	20	14	3	33	25	2	40	8	24	44	22	36	18	9	29	26	53	6	46	56
7	0	1	5	25	12	15	11	28	2	18	7	9	40	4	16	42	20	8	29	17	27	30	41	57	26	24	14	13	31	3	44	53
8	0	1	2	20	18	5	25	7	8	26	10	24	29	9	12	15	40	17	14	19	6	21	22	60	13	27	4	11	28	16	30	31
9	0	1	8	40	26	2	20	10	17	13	32	27	14	24	18	5	29	19	4	33	3	48	41	47	11	6	9	25	7	12	21	28
10	0	1	17	29	13	8	40	32	19	11	35	6	4	27	26	2	14	33	9	42	16	58	22	51	25	3	24	20	10	18	48	7
11	0	1	19	14	11	17	29	35	33	25	56	3	9	6	13	8	4	42	24	15	12	61	41	45	20	16	27	40	32	26	58	10
12	0	1	33	4	25	19	14	56	42	20	53	16	24	3	11	17	9	15	27	5	18	55	22	37	40	12	6	29	35	13	61	32
13	0	1	42	9	20	33	4	53	15	40	31	12	27	16	25	19	24	5	6	2	26	46	41	50	29	18	3	14	56	11	55	35
14	0	1	15	24	40	42	9	31	5	29	28	18	6	12	20	33	27	2	3	8	13	44	22	52	14	26	16	4	53	25	46	56
15	0	1	5	27	29	15	24	28	2	14	7	26	3	18	40	42	6	8	16	17	11	30	41	38	4	13	12	9	31	20	44	53
16	0	1	2	6	14	5	27	7	8	4	10	13	16	26	29	15	3	17	12	19	25	21	22	43	9	11	18	24	28	40	30	31
17	0	1	8	3	4	2	6	10	17	9	32	11	12	13	14	5	16	19	18	33	20	48	41	23	24	25	26	27	7	29	21	28
변화패턴 크기	1	1	8	17	17	8	17	8	8	17	8	17	17	17	17	8	17	8	17	8	17	8	2	17	17	17	17	17	8	17	8	8
갱신횟수	주소																															
0	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
1	35	42	49	56	57	50	43	36	29	22	15	23	30	37	44	51	58	59	52	45	38	31	39	46	53	60	61	54	47	55	62	63
2	56	15	59	53	60	52	23	57	14	41	5	34	21	50	30	45	61	54	38	37	43	28	36	44	31	47	55	39	51	46	62	63
3	53	5	54	31	47	38	34	60	4	22	2	49	48	52	21	37	55	39	43	50	23	7	57	30	28	51	46	36	45	44	62	63
4	31	2	39	28	51	43	49	47	9	41	8	59	58	38	48	50	46	36	23	52	34	10	60	21	7	45	44	57	37	30	62	63
5	28	8	36	7	45	23	59	51	24	22	17	54	61	43	58	52	44	57	34	38	49	32	47	48	10	37	30	60	50	21	62	63
6	7	17	57	10	37	34	54	45	27	41	19	39	55	23	61	38	30	60	49	43	59	35	51	58	32	50	21	47	52	48	62	63
7	10	19	60	32	50	49	39	37	6	22	33	36	46	34	55	43	21	47	59	23	54	56	45	61	35	52	48	51	38	58	62	63
8	32	33	47	35	52	59	36	50	3	41	42	57	44	49	46	23	48	51	54	34	39	53	37	55	56	38	58	45	43	61	62	63
9	35	42	51	56	38	54	57	52	16	22	15	60	30	59	44	34	58	45	39	49	36	31	50	46	53	43	61	37	23	55	62	63
10	56	15	45	53	43	39	60	38	12	41	5	47	21	54	30	49	61	37	36	59	57	28	52	44	31	23	55	50	34	46	62	63
11	53	5	37	31	23	36	47	43	18	22	2	51	48	39	21	59	55	50	57	54	60	7	38	30	28	34	46	52	49	44	62	63
12	31	2	50	28	34	57	51	23	26	41	8	45	58	36	48	54	46	52	60	39	47	10	43	21	7	49	44	38	59	30	62	63
13	28	8	52	7	49	60	45	34	13	22	17	37	61	57	58	39	44	38	47	36	51	32	23	48	10	59	30	43	54	21	62	63
14	7	17	38	10	59	47	37	49	11	41	19	50	55	60	61	36	30	43	51	57	45	35	34	58	32	54	21	23	39	48	62	63
15	10	19	43	32	54	51	50	59	25	22	33	52	46	47	55	57	21	23	45	60	37	56	49	61	35	39	48	34	36	58	62	63
16	32	33	23	35	39	45	52	54	20	41	42	38	44	51	46	60	48	34	37	47	50	53	59	55	56	36	58	49	57	61	62	63
17	35	42	34	56	36	37	38	39	40	22	15	43	30	45	44	47	58	49	50	51	52	31	54	46	53	57	61	59	60	55	62	63
변화패턴 크기	8	8	17	8	17	17	17	17	17	2	8	17	8	17	8	17	8	17	17	17	17	8	17	8	8	17	8	17	17	8	1	1
최대공약수	136																															

2.3. 새롭게 갱신되는 입출력 순서열 패턴의 유한성 및 반복성

블록크기의 단일 메모리를 사용하는 입출력 데이터 순서변환은 입력순서와 출력순서가 고정된 대응관계를 유지한다. 또한 1회의 블록처리에서 모든 주소는 한번만 그리고 빠짐없이 접근된다는 특성이 있다. 그리고 고정된 대응관계에 따라 처리되기 때문에 n번째 처리된 물리적 주소는 다음 차례에는 항상 고정된 m번째 순서로 처리가 된다. 이러한 물리적 주소의 처리순서 변화는 전체 블록 내 데이터의 수를 최대값으로 하는 고정된 형태의 변화패턴으로 나타난다. 표 3은 8x8 지그재그 스캔에서 각 물리적 주소가 접근되는 순서의 고정된 변화패턴을 보여주고 있다. 빗금 친 부분이 각 물리적 순서의 고정된 변화패턴을 나타낸다.

한편, 물리적 주소에 따른 각각의 변화 패턴은 n번째 처리된 주소는 다음 번에는 항상 고정된 m번째 처리된다는 특성이 있다. 또한 옵셋값으로 구분되는 동일한 순서열로 그룹화 된다.

표 4는 이런 특성을 보이는 8x8 지그재그 스캔에서 5-2-8-17-19-33-42-15 형태의 접근순서 변환패턴을 갖는 5번째 저장공간의 예이다. 표에서 볼 수 있듯이, 변화패턴에 속하는 모든 물리적 주소공간들은 5-2-8-17-19-33-42의 접근순서 변환패턴을 따르고 시작하는 순서만 다르다. 예를 들어 표 4의 2번 주소공간은 1이라는 옵셋값을 갖고 기본패턴인 5-2-8-17-19-33-42의 두 번째 숫자인 2부터 시작되는 2-8-17-19-33-42-15의 변화패턴을 가진다. 마찬가지 방법으로 표 4의 주소저장공간 8은 2의 옵셋값을 갖고 기본패턴의 세 번째 숫자인 8로 시작되는 8-17-19-33-42-15-5의 순서로 접근순서가 변한다.

또한, 표 3을 보면 각각의 물리적 저장공간마다 일정한 크기의 접근순서변화패턴으로 접근이 이루어짐을 알 수 있다. 이것을 이용하면 전체블록의 처리순서가 몇 번의 주소열 갱신 후에 처음의 순서로 돌아오게 되는지 구할 수 있다. 이 과정은 다음과 같이 설명할 수 있다. 각 주소저장공간은 고유한 변화패턴의 크기만큼의 전체블록처리가 이루어지면 다시 처음의 접근순서를 갖는다. 따라서 각각의 모든 변화패턴 크기의 최소공배수에 해당하는 횟수만큼의 블록처리가 이루어지면 모든 주소저장공간이 처음의 처리순서와 동일한 처리순서를 갖게 된다. 예를 들어 8x8 지그재그 스캔의 경우 각 변화패턴의 크기는 표 4에서와 같이 1, 2, 8, 17이며 최소공배수는 136이므로 136번의 블록처리과정을 거치면 처음의 순서열로 돌아온다.

같은 방법으로 H.264나 HEVC에 사용되는 여러 스캔 방법에 대하여 단일 메모리를 사용하는 방법을 적용하였을 때의 주소변화패턴과 처음의 순서열로 돌아오는 최소횟수인 최소공배수를 구하면 표 5와 같다.

표 4. 동일 접근순서 변화패턴(5-2-8-17-19-33-42-15)을 보이는 주소 그룹군
Table. 4 Address group with same pattern of access order (5-2-8-17-19-33-42-15)

물리적 주소	갱신횟수에 따른 접근순서의 변화							
	갱신횟수							
	0	1	2	3	4	5	6	7
5	5	2	8	17	19	33	42	15
2	2	8	17	19	33	42	15	5
8	8	17	19	33	42	15	5	2
17	17	19	33	42	15	5	2	8
19	19	33	42	15	5	2	8	17
33	33	42	15	5	2	8	17	19
42	42	15	5	2	8	17	19	33
15	15	5	2	8	17	19	33	42

표 5. 스캔에서 반복되는 주소변환패턴의 크기와 최소공배수
Table. 5 Length of repeated address pattern and its LCM

스캔 종류	변환패턴의 크기	최소공배수
지그재그 4x4	1, 3, 6	6
지그재그 4x4 field	1, 2, 6	6
대각선 4x4	1, 4, 6	12
지그재그 8x8	1, 2, 8, 17	136

2.4. 제안하는 효율적인 주소발생기의 구현방법

표 4에서와 같이, 특정 주소가 접근되는 순서는 전체 블록이 한번 처리될 때마다 고정된 패턴을 따라 변화한다. 또한 동일 패턴에 속한 물리적 주소의 처리순서는 시작하는 순서만 다를 뿐 같은 변화패턴을 보이며 바뀐다. 이와 같은 특성을 이용하여 범용으로 사용할 수 있는 주소발생기의 구조를 그림 5와 같이 나타내었고 다음과 같이 구현하였다. 먼저 각각 블록의 개별 데이터를 세는 픽셀카운터를 준비한다. 8x8 블록의 경우 0-63까지의 값을 갖는다.

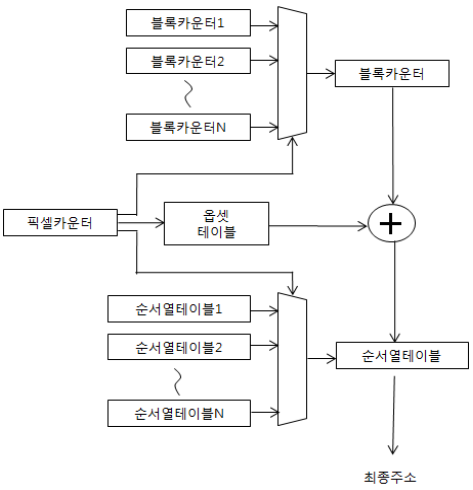

그림 5. 주소발생기의 구조
Fig. 5 Architecture of address generator

그리고 각각의 주소 변화패턴마다 갱신횟수를 측정하는 블록카운터를 설정한다. 이 카운터는 픽셀카운터가 데이터가 처리될 때마다 증가하는 것과는 달리 전체 블록이 한번 처리될 때마다 증가한다. 또한 이 때 변화패턴의 크기가 카운터의 최대값이 된다. 예를 들어 표 4의 5-2-8-17-19-33-42-15 변화패턴은 0-7의 값을 갖는 블록카운터를 사용한다. 이 때 같은 패턴을 갖는 그룹은 동일한 블록카운터를 사용하고, 각각은 고유한 옵셋값으로 구분된다. 한편 블록카운터는 0, 1, 2, 3, ..과 같이 기본적인 증가순서를 따른다. 반면 실제 변화패턴은 고유한 변환순서를 갖기 때문에 카운터와 실제 변화패턴을 대응시키는 테이블이 필요하다.

그림 5의 순서열 테이블이 이에 해당한다. 표 4의 예를 보면 0-1-2-3-4-5-6-7의 일반 카운터를 실제 변화패턴으로 바꿔주기 위하여 5-2-8-17- 19-33-42-15의 순서열 테이블이 사용된다. 이렇게 정해지는 블록카운터와 옵셋 값을 더하고 순서열 테이블을 이용해 대응값을 찾으면, 그림 5에서와 같이 최종주소를 발생시킬 수 있다. 8x8 지그재그 스캔의 경우를 예로 들면 다음과 같다. 표 3에서 9번째 처리순서(표 3에서의 8번 주소)의 경우를 보면 8-17-19-33-42-15-5-2의 주소변화패턴을 보인다. 이것은 앞서 3번째 처리순서(2번주소)의 변화패턴 2-8-17-19-33-42-15-5에 대해 옵셋값 1을 갖는 변화패턴으로 볼 수 있다. 이 때 4번째 갱신 후, 9번째 처리순서(표 3에서 주소 8)에 해당하는 주소를 찾는다고 가정해 보자. 해당하는 기본 주소변환패턴이 8-17-19-33-42-15-5-2 이므로, 갱신회수 4에 옵셋값 1을 더한 5번째 숫자인 42를 찾을 수 있고, 이것은 표 3에서 찾을 수 있는 값(주소 8, 갱신횟수 4) 42와 일치함을 알 수 있다.

Ⅲ. 구현 및 검증

그림 6은 출력 순서가 바뀌는 것을 고려하여 작성한 지그재그 8x8 스캔 블록에 대한 테스트 벡터의 예이다. 정상적인 입출력 대응관계를 고려하여 정상적인 순서변환이 이루어지면 출력데이터는 항상 0, 1, 2, 3, ... 이 나오도록 하였다.

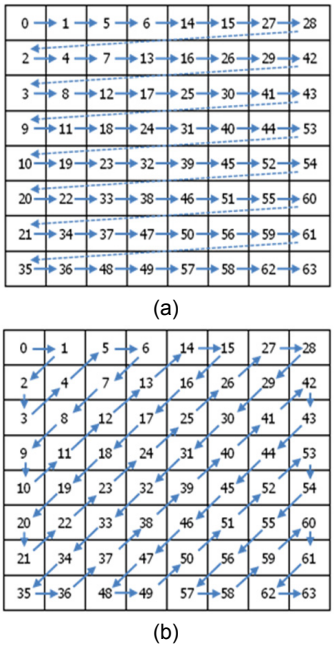

그림 6. 지그재그 8x8 스캔 블록의 테스트 벡터 (a) 입력데이터 : 0, 1, 5, 6, ····· (b) 출력 데이터 : 0, 1, 2, 3, ·····
Fig. 6 Test vector of zigzag 8x8 scan block (a) Input data : 0, 1, 5, 6, ····· (b) output data: 0, 1, 2, 3, ·····

제시한 방법은 Verilog HDL로 설계 및 검증을 한 후 TSMC 0.18um CMOS 공정 라이브러리로 합성하여 타이밍과 면적을 확인하였다. H.264, HEVC에 사용되는 지그재그 4x4, ,대각선 4x4 스캔에 대해서도 같은 방법

을 적용하여 설계하였다. 기존의 방법과 비교한 결과를 표 6에 나타내었다. 평균 43.5%이상 게이트수가 감소하는 효과를 얻을 수 있었다.

표 6. 다양한 스캔 블록에서의 구현결과
Table. 6 Implementation results in various scan blocks

구분	게이트수		
	기존방법	제안한 방법	감소율
지그재그 4x4	4,847	2,634	45.7%
대각선 4x4	4,838	2,730	43.6%
지그재그 8x8	21,976	12,931	41.2%

Ⅳ. 결 론

일반적으로 데이터 블록의 단순한 실시간 입출력 순서 변환은 처리하고자 하는 블록크기의 메모리만을 이용하여 동시에 입출력 데이터를 교환하는 방법으로 구현해 왔다.

본 논문에서는 그러한 방법을 적용하기 어려웠던 불규칙한 형태의 입출력 순서 변환에 있어서 갱신되는 순서열 사이에 일정한 규칙성과 반복성이 있음을 밝혔다. 그리고 그러한 원리를 이용하여 어떤 순서 변환의 경우에도 보편적으로 적용시킬 수 있고 효율적인 하드웨어 구현 방법을 제시하였다. 제시한 방법은 HDL 설계와 ASIC 라이브러리 적용을 통해 면적이나 처리속도 면에서 기존의 방법에 비해 효율적임을 보였다.

감사의 글

본 연구는 교육부와 한국연구재단의 지역혁신인력양성사업(NRF-2012H1B8A2025862)과 미래창조과학부 및 정보통신산업진흥원의 해외인재스카우팅사업(NIPA-HB616-13-1001)의 지원으로 수행되었습니다.

REFERENCES

[1] Junho Kim, "Design and Verification of High-Performance JPEG IP using SoC Platform SoC Platform," *Proceedings of IEEK summer conference*, Vol 30, No 1, pp. 87-88, June 2007.

[2] Jun Rim Choi, et. al., "A 400MPixel/s IDCT HDTV by Multibit Coding and Group Symmetry," *Solid-State Circuits Conference*, 1997. Digest of Technical Papers. *43rd ISSCC*., pp. 262-263, San Francisco, CA, Feb. 1997.

[3] Jian-Jiun Ding, et. al., "Context-based adaptive zigzag scanning for image coding," *Visual Communications and Image Processing*, 2011 IEEE, pp. 1-4, Nov. 2011.

[4] Minjung Lim, "Architecture of Real-time JPEG Input Buffer," *Journal of IEEK*, Vol 39-SD, No 2, pp. 7-13, Feb. 2002.

[5] Gunjoong Lee, "Efficient hardware Design for Zigzag Scanning using Indirect Memory Addressing Scheme," *Proceeding of IEEK Conference*, pp. 87-88, Dec 2011.

이건중(Gunjoong Lee)

1995년 서강대학교 공과대학 전자공학과 공학사
2013년 한밭대학교 정보통신전문대학원 정보통신공학과 공학석사
2013년~현재 ㈜에이치기술 ASIC설계팀 연구원
※관심분야 : SoC 플랫폼 설계 및 검증, 멀티미디어 코덱 설계

류광기(Kwangki Ryoo)

2000년 한양대학교 대학원 전자공학과 공학박사
1991년~1994년 육군사관학교 교수부 전자공학과 전임강사
2000년~2002년 ETRI 시스템IC설계팀 선임연구원
2010년~2011년 Univ of Texas at Dallas 방문교수
2003년~현재 한밭대학교 공과대학 정보통신공학과 교수
※관심분야 : 공학교육, SoC 플랫폼 설계 및 검증, 멀티미디어 코덱 설계