# JPEG 2000을 위한 Tiling 시스템의 구현

# Implementation of Tiling System for JPEG 2000

장원우*, 조성대**, 강봉순***
Won-woo Jang*, Sung-dae Cho**, and Bong-soon Kang***

## 요 약

본 논문은 JPEG 2000에 사용되는 전처리 과정 기능인 타일링 시스템의 구현에 관한 것이다. 설계된 시스템은 JPEG 2000의 표준에 명시되어 있으며, 이미지의 크기 파악, 영역 확장 그리고 이미지 분할 기능을 수행한다. Progressive한 입력을 타일 단위로 분할 및 전송하기 위해서, 입력 이미지를 Frame Memory에 저장한다. 그래서 Verilog-HDL를 사용하여 FSM 방식으로 설계되었으며 최대 5M 이미지를 처리할 수 있다. 또한, 영역 확장을 위한 이미지 크기를 파악하기 위해서 나머지(rem) 연산을 기반으로 한 수식을 만들었다. 이를 이용해서 입력 이미지의 크기 패턴을 파악하는 진리표를 제안한다. TSMC 0.25um ASIC library 환경에서 합성된 gate counts는 18,725로 되었으며 maximum data arrival time은 18.94[ns]를 가진다.

## Abstract

This paper presents the implementation of a Tiling System about Preprocessing functions of JPEG 2000. The system covers the JPEG 2000 standard and is designed to determine the size of the image, to expand the image area and to split input image into several tiles. In order to split the input image with the progressive transmission into several tiles and transmit a tile of this image to others, this system store this image into Frame Memory. Therefore, this is designed as the Finite State Machine (FSM) to sequence through specific patterns of states in a predetermined sequential manner by using Verilog-HDL and be designed to handle a maximum 5M image. Moreover, for identifying image size for expansion, we propose several formula which are based on remainder after division (rem). we propose the true table which determines the size of the image input patterns by using results of these formula. Under the condition of TSMC 0.25um ASIC library, gate count is 18,725 and maximum data arrival time is 18.94 [ns].

***Keywords*** : JPEG 2000, FSM, Tiling, Remainder

## I. 서 론

JPEG 2000 표준은 현재 널리 사용되고 있는 정지영상 표준인 JPEG(Joint Photographic Experts Group)의 단점을 극복하기 위해 2000년대 초반 개발된 최신 정지영상 압축 기술이다 [1]. JPEG 2000은 현재 디지털 카메라 및 모바일 카메라와 데이터 전송 등 여러 분야에서 일반적으로 사용되는 JPEG 표준보다 압축효율 면에서 더 뛰어나며 이미지의 왜곡이 더 적다고 알려져 있다. 또 정지 영상 부호화에 관한 국제 표준화 진행에 따른 요구가 증대됨에 따라 JPEG 2000 표준의 중요성이 날로 증대되고 있다.

그림 1은 각각 JPEG과 JPEG 2000의 성능을 비교하기 위해 Lenna 영상을 사용하여 43:1의 압축 비율로 압축한 것이다 [2]. 그림 1-(a)는 입력 영상이며, 그림 1-(b)는 JPEG을 사용하여 인코딩하고 디코딩한 영상이다. 그림 1-(c)는 동일한 영상을 JPEG 2000을 사용하여 인코딩하고 디코딩한 영상이다. 그림에서 보듯이 Peak Signal-to-Noise Rate (PSNR)과 같은 객관적 수치에 근거하지 않아도 될 정도로 시각적으로 차이가 나타나고 있다. 이와 같이 비교영상에서 보는 바와 같이 동일한 압축 비율을 사용했음에도 불구하고 JPEG 2000이 월등한 성능을 보여주는 것을 알 수 있다. 또한 JPEG 2000은 약 1/150 압축률을 제공하며 저장 공간 및 이미지 전송시간을 줄일 수 있으며 동일 화질대비 3~5배의 압축 효과를 얻을 수 있다.

* 동아대학교 전자공학과 박사과정
** (주) 삼성전자
*** 교신저자 : 동아대학교 전자공학과 부교수
투고 : 2008. 5. 30 수정완료 : 2008. 7. 22
게재확정일자 : 2008. 7. 25
※ 이 논문은 동아대학교 학술연구비 지원에 의하여 연구되었음

(a) (b) (c)

그림 1. JPEG 2000과 JPEG의 성능 비교,
(a) 입력 영상, (b) JPEG, (c) JPEG 2000
Fig. 1. The performance comparison of JPEG and JPEG 2000, (a) Input Image, (b) JPEG, (c) JPEG 2000

JPEG 2000은 입력 영상의 분석 방법으로 Discrite Wavelet Transform (DWT)를 이용한다. 영상의 대부분의 에너지는 저주파 대역에 분포하고 있으므로, 저주파 대역의 데이터는 보존하고, 고주파 대역의 데이터 양을 줄이기 위한 양자화 과정을 수행한다. 양자화 된 계수를 엔트로피 코딩을 하면 압축된 데이터가 구성되게 된다. 그림 2는 JPEG 2000 Encoder 부분의 블록 다이어그램이다 [3].

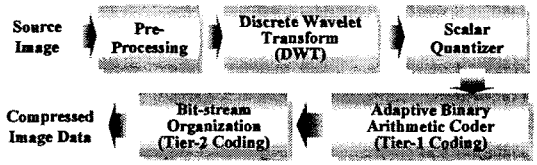


그림 2. JPEG 2000의 Encoder 블록 다이어 그램
Fig. 2. The JPEG 2000 block diagram : encoder

영상을 전처리 과정인 (Tiling , DC level shifting, Color space transform)을 수행한 후 DWT를 수행한다. DWT 변환 필터 계수는 크게 9-7 filter(lossy), 5-3 filter(lossless)의 두 가지로 분류된다. DWT 변환 레벨은 최대 4 레벨까지 있으며, 변환된 계수는 밴드 별로 scalar 양자화를 거쳐 영상의 상태에 따라 Entropy coding하는 Tier-1과정을 수행하게 된다. 그리고 최종적으로 Tier-2 블록이 bit-stream을 구성하여 압축 영상 데이터를 출력하게 된다. Tier-1과 Tier-2를 통합하여 EBCOT라고 한다 [4]. 본 논문이 언급하고자 하는 내용은, 전처리 과정인 타일링(Tiling) 시스템에서, 영역 확장을 위한 이미지 크기 예측을 위한 수식, 이미지 패턴 예측을 위한 진리표, 그리고 하드웨어로 구현 방법에 대해서 언급하려고 한다.

## II. 알고리즘

타일링 기능은 입력 영상을 사각형의 겹치지 않는 영역으로 분할하는 과정을 말하며. 이렇게 분할된 영역을 타일(Tile)이다. 분할된 영상의 모든 타일들은 JPEG 2000 과정인 Color Component Transform, DWT, 양자화, Entropy 코딩을 거쳐 압축되게 되며, 모든 타일들이 각각 압축 과정을 수행한다. 타일들이 가질 수 있는 크기는 2의 지수 승으로 가로 세로가 최소 4에서 최대 1024의 크기를 가진다. 그러나 타일의 크기를 16×16 이하로 타일링을 수행할 경우 너무 작은 크기의 타일로 영상을 분해하게 되고, 이를 각각 JPEG 2000 인코딩과 디코딩을 통해서 이미지로 복원 하였을 때 화질 열화현상이 나타나기 때문이다 [5]. 본 논문에서 주로 사용하고자 하는 고해상도의 입력 영상이므로 굳이 작은 타일로 영상을 분해할 필요가 없으므로 타일의 기본 해상도를 512×512로 설정하여 이미지를 분할한다. 이러한 방법은 가로 세로 해상도가 동일한 타일을 생성시키기도 하지만, 입력해상도에 따라 다양한 예외 경우가 생기게 되기 때문에 이에 대한 예외 처리가 필요하다. 실제 표준 이미지 해상도를 가지고 예를 들어 설명하겠다. 그림 3은 예외의 경우가 없는 3Mega (2048×1536) 영상의 분할과정을 나타내고 있다.

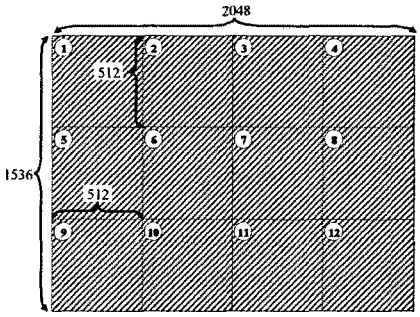


그림 3. 3M (2048×1536) 이미지의 타일링
Fig. 3. 3M (2048×1536) image by Tiling

타일링의 기본 타일 크기는 앞서 언급한 바와 같이 512×512이다. 3M 영상의 경우는 영상의 가로 사이즈는 2048 픽셀이고 세로 사이즈는 1536 픽셀이다. 이를 512×512의 타일링 모드를 적용했을 경우 2048/512=4, 1536/512=3으로 정확하게 나머지 없이 12개 타일로 분할되며, 이는 예외의 경우가 발생하지 않는 경우이다. 영상의 확장 없이 512×512의 영상정보와 함께 수평 동기 신호인 hav와 수직 동기 신호인 vav를 내부에서 생성하여 내보내게 된다. 하지만 위의 3M 영상과는 다르게 5M (2536×1944) 영상은 예외의 경우가 필요하다. 그림 4는 예외의 경우가 발생하게 되는 5M 영상의 분할과정을 나타내고 있다.

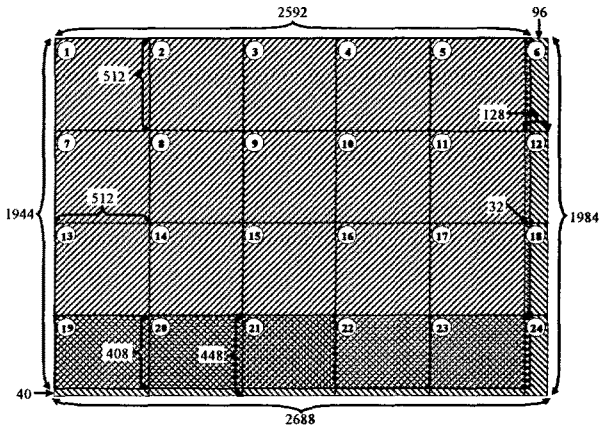


그림 4. 5M (2536×1944) 이미지의 타일링
Fig. 4. 5M (2536×1944) image by Tiling

5M 영상의 경우는 가로 사이즈는 2536 픽셀이고 세로 사

이즈는 1944 픽셀이다. 이를 512×512의 기본 크기를 적용했을 경우, 기본 타일 이외에도 다양한 형태의 예외 경우가 발생한다. 5M 영상의 오른쪽 영역에는 32×512 3개(⑥,⑫,⑱), 영상의 아래에는 512×408 5개(⑲,⑳,㉑,㉒,㉓), 오른쪽 아래에는 408×32 1개(㉔) 발생한다. 이러한 예외 규칙이 적용되는 9개의 타일들을 해결하기 위해서 영상 확장 방법을 사용한다 [6].

입력 영상의 예외 타일이 가로 세로 사이즈가 128 픽셀보다 작게 되면 이는 무조건 128 픽셀로 확장되게 된다. 그리고 128이상이 되면 64의 가장 가까운 배수(예외의 경우 수보다 큰 쪽으로)로 확장되게 된다. 예를 들어 예외 타일의 사이즈가 130×128이라고 가정한다. 이 중 130 이상의 숫자 중에 가장 가까운 64의 배수는 192이다. 그러므로 130×128의 예외 타일은 192×128의 크기로 확장된다. 이 같이 128의 크기 이상으로 예외 타일을 확장하는 이유는 타일링 이후 처리되는 DWT와 EBCOT Tier-1 과정 때문인데 분리된 타일에 DWT를 수행하게 되면 영상의 해상도는 1 레벨인 경우 가로 세로 사이즈가 각각 1/2, 2 레벨인 경우 1/4, 3 레벨인 경우 각각 1/8, 4 레벨인 경우 1/16인 경우로 줄어들게 된다. 어떤 영상의 가로 사이즈가 128이라고 가정한다. 이 영상을 1 레벨 DWT 하면 64, 이를 2 레벨 DWT 하면 32, 이를 3 레벨 DWT 하면 16, 이를 4 레벨 DWT하면 최종적으로 가로 사이즈가 8로 줄어든다. DWT 이후 EBCOT에서 처리되는 코드 블록의 가로 세로 크기는 2의 지수 형태, $2^x$, $2^y$로 표시할 때, x+y<=12으로 제한한다 [7]. 그래서 표준에서 제시하는 EBCOT의 코드 블럭의 크기는 최소 8×8 에서 최대 64×64가 되는 것이다. 또한 Tier-1에서 처리 할 수 있는 타일의 크기 조건이 있으므로 타일링에서 예외의 최소 확장 사이즈를 128로 한 것은 DWT 4 레벨까지 수행하였을 경우에 타일의 크기 제한이 걸리지 않도록 하기 위해서 이다. 표 1은 예외에 해당되었던 타일들을 확장 규칙에 의해 변화된 결과를 나타낸다.

표 1. 각 예외 타일들의 확장 결과
Table 1. Results of the Extended Each Exception of the Tiles

| [illegible] | [illegible] (가로×세로) | [illegible] (가로×세로) | [illegible] |
|---|---|---|---|
| ⑥,⑫,⑱ | 32×512 | 128×512 | 가로 +96 |
| ⑲,⑳,㉑,㉒,㉓ | 512×408 | 512×448 | 세로 +40 |
| ㉔ | 408×32 | 448×128 | 가로 +96<br>세로 +40 |
| 5M 이미지 | 2536×1944 | 2688×1984 | 가로 +96<br>세로 +40 |

위의 언급된 하드웨어 설계를 하기 위해서 확장 규칙에 대한 수식이 필요하다. 이미지가 2차원이기 때문에, 가로와 세로에 대한 고려가 필요하나 두 방향이 같은 속성을 지니기 때문에 가로 방향만 가지고 설명한다. 수식들은 나머지 연산 방법이 사용하였다. 수식 1에서 4는 이미지의 크기가 가질 수 있는 수식이다. 수식 1에서 $x_{width}$는 입력 영상의 가로 사이즈이며, $b_{width}$는 이미 언급된 기본 타일의 가로 길이(512 픽셀)이다. 기본 타일의 크기보다 입력되는 된 타일의 크기가 작을 경우를 파악하기 위한 수식이다.

$$x_{width} < b_{width} \tag{1}$$

수식 2는 나머지(*rem*) 연산 결과와 $min_{size}$과 비교한다. $min_{sizes}$는 확장될 이미지의 사이즈의 최소 단위(128 픽셀)이다. $x_{rem}$은 가로축을 예외 타일의 크기이며 나머지를 이용해서 $x_{width}$을 $b_{width}$으로 나눈 후 나머지이다. 이후 수식에 $y_{rem}$은 세로축을 예외 타일의 크기이며 나머지를 이용해서 $x_{width}$을 $b_{width}$으로 나눈 후 나머지이다. 입력 영상이 128 픽셀보다 작게 되면 이는 무조건 128 픽셀로 확장 시킨다.

$$x_{rem} = rem(x_{width}, b_{width}) < min_{size} \tag{2}$$

수식 3은, 수식 2와 유사하게, 예외 타일의 크기가 0인지 아닌지를 판단하여 $x_{width}$의 값이 $b_{width}$의 배수인지를 판단한다.

$$x_{rem} == 0 \tag{3}$$

수식 4는 예외 타일의 크기에서 $min_{size}$의 값을 빼고 $jump_{size}$와의 나머지 연산을 통해서 배수인지 파악한다. DWT 이후 EBCOT에서 처리하기 위해서, $jump_{size}$는 확장된 이미지의 길이가 64 픽셀을 가진다.

$$rem((x_{rem} - min_{size}), jump_{size}) == 0 \tag{4}$$

수식 1-4들이 가질 수 있는 경우의 수는 16가지이다.

표 2. 입력 이미지에 확장을 위한 진리표
Table 2. True Table for Extending the Image Input

| [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 수식 11 |
| 1 | 0 | 0 | 0 | 1 | 수식 9 |
| 2 | 0 | 0 | 1 | 0 | 수식 9 |
| 3 | 0 | 0 | 1 | 1 | 수식 9 |
| 4 | 0 | 1 | 0 | 0 | 수식 10 |
| 5 | 0 | 1 | 0 | 1 | 수식 10 |
| 6 | 0 | 1 | 1 | 0 | 수식 9 |
| 7 | 0 | 1 | 1 | 1 | 수식 9 |
| 8 | 1 | 0 | 0 | 0 | 수식 8 |
| 9 | 1 | 0 | 0 | 1 | 수식 9 |
| 10 | 1 | 0 | 1 | 0 | 수식 9 |
| 11 | 1 | 0 | 1 | 1 | 수식 9 |
| 12 | 1 | 1 | 0 | 0 | 수식 7 |
| 13 | 1 | 1 | 0 | 1 | 수식 7 |
| 14 | 1 | 1 | 1 | 0 | 수식 7 |
| 15 | 1 | 1 | 1 | 1 | 수식 7 |

그러나 이미지가 확장되는 경우의 수는 $min_{size}$로 확장과 $jump_{size}$로 확장의 경우를 포함해서 5가지의 경우가 발생하며 이에 대한 결과는 수식 7-11과 연결되며 표 2는 언급된 16가지 경우에 대한 진리표이다. 각 개별 수식에 대해서 조건이 맞을 경우에는 값이 1이 되고 틀릴 경우에는 0이 되도록 기

록하여 16가지 경우를 서술하였다.

우선 수식 5와 6은 수식 7-11을 간략화 시키기 위한 서술이다. 수식 5에서 $R_1$은 $x_{width}$를 $b_{width}$으로 나눈 몫을 음수의 방향으로 반올림(*floor*) 또는 소수점 이하 버림 한 것이며 입력 이미지의 가로 타일 개수와 같다.

$$R_1 = floor(\frac{x_{width}}{b_{width}}) \quad (5)$$

수식 6에서 $R_2$는 예외 타일의 나머지에서 $min_{size}$를 빼고 난 뒤 $jump_{size}$를 나눈 몫의 값이 가로 방향으로 최소 몇 번 확장 되는가 알 수 있다.

$$R_2 = floor(\frac{x_{rem} - min_{size}}{jump_{size}}) \quad (6)$$

수식 7은 표 2에서 수식 1과 2가 만족 할 경우, 입력 이미지가 기본 타일 보다 작아서 최소 크기보다도 작을 때, 입력 이미지의 크기를 $min_{size}$로 확장하는 경우이다. $x_{width_modified}$는 확장된 가로 길이 값이다.

$$x_{width\_modifided} = min_{size} \quad (7)$$

수식 8은 표 2에서 수식 1이 만족하고 나머지 수식이 만족하지 않는 경우로, 기본 타일 보다 작으나, 최소 크기보다 크고, $jump_{size}$의 배수가 아닐 때, 이 값으로 치환된다.

$$x_{width\_modifided} = (R_2 + 1) \times jump_{size} + min_{size} \quad (8)$$

수식 9은 표 2에서 절반 이상을 차지하고 있으며, 입력 이미지의 크기를 그대로 사용하여 확장이 필요 없다.

$$x_{width\_modifided} = x_{width} \quad (9)$$

수식 10은 수식 1이 만족하지 않는 경우, 입력 이미지가 기본 타일보다 커서, 타일이 2개 이상이며, 예외 타일에 확장이 최소크기로 필요한 경우이다. 그래서 수식 5에서 얻는 $R_1$

$$x_{width\_modifided} = R_1 \times b_{width} + min_{size} \quad (10)$$

의 값을 이용해서 기본 타일의 길이를 얻을 수 있다.

수식 11은 수식 1이 만족하지 않는 경우, 입력 이미지가 기본 타일보다 커서 타일이 2개 이상이며 예외 타일에 대한 최소 크기 및 추가적인 증가가 필요한 경우이다.

$$x_{width\_modifided} = R_1 \times b_{width} + min_{size} + (R_2 + 1) \times jump_{size} \quad (11)$$

최종적으로 5M 이미지의 전체 사이즈는 가로 2592 픽셀, 세로 1944 픽셀에서 가로 2688 픽셀, 세로 1984 픽셀로 확장되며 확장된 정보에 따라 이후의 처리 단계인 2-D DWT, EBCOT가 동작하게 된다.

일반적인 영상 장치들의 데이터 전송 순서는 Progressive 형태를 가진다. 이것은 좌에서 우로 그리고 위에서 아래의 순서를 가지며 그림 5와 같다. 일반적으로 디스플레이 장치에서 보는 이미지는 Valid Image 영역만이 나타난다. 그러나 영상은 Blanking구간이 포함되어 있기 때문에 시간적으로 여유가 있다. 그래서 Valid영역에서 데이터를 처리하고, Vertical 또는 Horizontal Blanking 영역에서 여러 가지 일들을 수행한다.

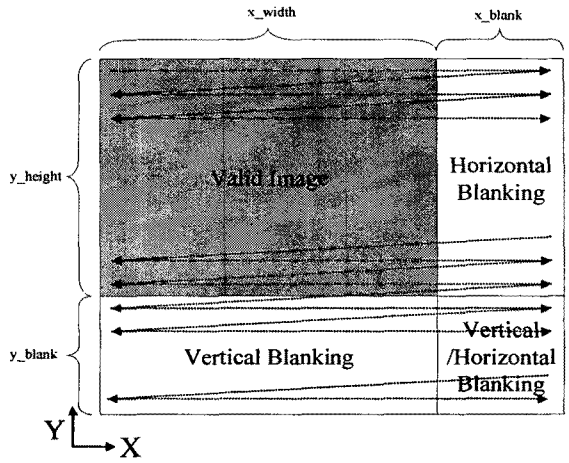


그림 5. Progressive 영상 전송 및 유효 영역 이미지
Fig. 5. Progressive Transmission and Valued Image Area

Progressive 형태의 입력에서는 n 번째 줄의 데이터와 n+1 줄의 데이터 간에는 가로 픽셀 수 만큼 시간 차이가 있다. 그러나 타일링 이후 처리되는 DWT와 EBCOT Tier-1를 위한 데이터는 각 개별 타일, 즉 2차원 데이터 형태를 가져야 한다. 입력되는 타일의 개수가 1개 즉 입력되는 이미지의 크기가 기본 타일(512×512)과 같거나 작을 때에는 타일링 이후 처리 과정으로 Progressive한 순서, 즉 입력되는 순서로 보내면 된다. 반면에 타일의 개수가 2개 이상이라면, 데이터의 입력 순서와 출력 순서와 서로 다르기 때문에 데이터들을 저장할 메모리가 필요하며, 이후 데이터들을 메모리에서 읽어 낼 때, 타일 형태를 만들기 위해서 어드레스 발생에 대해서도 고려가 필요하다.

수식 12-13은 수정된 이미지 크기($x_{width\_modified}$, $y_{height\_modified}$)에 대해서 나머지 연산을 통해서, 확장된 이미지의 가로 세로 나머지($x_{m\_rem}$, $y_{m\_rem}$)를 파악한다.

$$x_{m\_rem} = rem(x_{width\_modified}, b_{width}) \quad (12)$$

$$y_{m\_rem} = rem(y_{height\_modified}, b_{width}) \quad (13)$$

수식 14-15는 가로 세로의 타일 개수에 대해서 파악하기 위한 수식으로 양수의 방향으로 소수점 올림(*ceil*) 연산을 이용하였다. $x_{tile_num_size}$는 확장된 이미지의 가로 타일 개수이고, $y_{tile_num_size}$는 확장된 이미지의 세로 타일 개수를 나타낸다.

$$x_{tile\_num\_size} = ceil(x_{width\_modified} / b_{width}) \quad (14)$$

$$y_{tile\_num\_size} = ceil(y_{height\_modified} / b_{width}) \quad (15)$$

수식 16은 수식 14-15의 결과를 곱해서 전체 타일 개수($tile_{total\_size}$)를 구한다.

$$tile_{total\_size} = x_{tile\_num\_size} \times y_{tile\_num\_size} \quad (16)$$

수식 17은 Progressive한 입력된 것을 최종적으로 타일 단

위로 전송하기 위한 pseude-code이며, 내부의 조건 4가지 (*C1*,*C2*,*C3*,*C4*)는 표 3에서 16가지 경우에 대한 경우를 가진다.

$$
\begin{aligned}
&for\ \ tile_{num}=1:tile_{total\_size}\\
&\quad x_{num}=rem(tile_{num},x_{tile\_num\_size})\\
&\quad y_{num}=floor((tile_{num}-1)/x_{tile\_num\_size})\\
&\quad y_{end}=floor((tile_{total\_size}-1)/x_{tile\_num\_size})\\
&\quad if\ (x_{m\_rem}==0)?\rightarrow C1\\
&\quad if\ (y_{m\_rem}==0)?\rightarrow C2\\
&\quad if\ (x_{num}==0)?\rightarrow C3\\
&\quad if\ (y_{num}==y_{end})?\rightarrow C4\\
&\quad for\ \ i=0:y_{size}\\
&\quad\quad for\ \ j=0:x_{size}\\
&\quad\quad\quad if((j<x_{size\_org}\,|\,x_{size\_org}==0)\&...\\
&\quad\quad\quad\quad (i<y_{size\_org}\,|\,y_{size\_org}==0))\\
&\quad\quad\quad\quad index=x_{width}\times[(y_{m\_rem}-1)\times b_{width}+i]+...\\
&\quad\quad\quad\quad\quad\quad (x_{m\_rem}-1)\times b_{width}+j\\
&\quad\quad\quad\quad tile\_data\_line=ex\_memory(index)\\
&\quad\quad\quad else\\
&\quad\quad\quad\quad tile\_data\_line=0\\
&\quad\quad end\\
&\quad end\\
&end
\end{aligned}
\tag{17}
$$

표 3. 타일 형태의 데이터 출력을 위한 진리표
Table 3. True Table for Tile in the form of output data

| Num | C1 | C2 | C3 | C4 | $x_{size}$ | $x_{size\_org}$ | $y_{size}$ | $y_{size\_org}$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | $B_{width}$-1 | 0 | $B_{width}$-1 | 0 |
| 1 | 0 | 0 | 0 | 1 | $B_{width}$-1 | 0 | $y_{m\_rem}$-1 | $y_{rem}$ |
| 2 | 0 | 0 | 1 | 0 | $x_{m\_rem}$-1 | $x_{rem}$ | $B_{width}$-1 | 0 |
| 3 | 0 | 0 | 1 | 1 | $x_{m\_rem}$-1 | $x_{rem}$ | $y_{m\_rem}$-1 | $y_{rem}$ |
| 4 | 0 | 1 | 0 | 0 | $B_{width}$-1 | 0 | $B_{width}$-1 | 0 |
| 5 | 0 | 1 | 0 | 1 | $B_{width}$-1 | 0 | $B_{width}$-1 | $y_{rem}$ |
| 6 | 0 | 1 | 1 | 0 | $x_{m\_rem}$-1 | $x_{rem}$ | $B_{width}$-1 | 0 |
| 7 | 0 | 1 | 1 | 1 | $x_{m\_rem}$-1 | $x_{rem}$ | $B_{width}$-1 | $y_{rem}$ |
| 8 | 1 | 0 | 0 | 0 | $B_{width}$-1 | 0 | $B_{width}$-1 | 0 |
| 9 | 1 | 0 | 0 | 1 | $B_{width}$-1 | 0 | $y_{m\_rem}$-1 | $y_{rem}$ |
| 10 | 1 | 0 | 1 | 0 | $B_{width}$-1 | $x_{rem}$ | $B_{width}$-1 | 0 |
| 11 | 1 | 0 | 1 | 1 | $B_{width}$-1 | $x_{rem}$ | $y_{m\_rem}$-1 | $y_{rem}$ |
| 12 | 1 | 1 | 0 | 0 | $B_{width}$-1 | 0 | $B_{width}$-1 | 0 |
| 13 | 1 | 1 | 0 | 1 | $B_{width}$-1 | 0 | $B_{width}$-1 | $y_{rem}$ |
| 14 | 1 | 1 | 1 | 0 | $B_{width}$-1 | $x_{rem}$ | $B_{width}$-1 | 0 |
| 15 | 1 | 1 | 1 | 1 | $B_{width}$-1 | $x_{rem}$ | $B_{width}$-1 | $y_{rem}$ |

$tile_{num}$은 전송될 타일의 번호 값이다. $x_{num}$은 입력된 타일의 가로축 번호를 나타내며, $y_{num}$은 입력된 타일의 세로 축 번호를 나타내며, $y_{end}$은 입력된 타일의 전체 세로축 번호를 나타낸다. 세로 방향의 수식들에서 -1을 한 이유는, 이미지의 확장된 부분일 때, 수식의 값이 0 될 수 있기 때문이다.

*C1*는 확장된 이미지의 가로축으로 타일 개수가 1개 인지 그 보다 많은 지를 판별하는 조건이다. *C2*는 확장된 이미지의 세로축으로 타일 개수가 1개 인지 그 보다 많은 지를 판별하는 조건이다. *C3*는 현재 타일이 가로축으로 마지막인지 아닌지를 판별하는 조건이다. *C4*는 현재 타일이 세로축으로 마지막 번째 줄인지 아닌지를 판별하는 조건이다. 이러한 4가지 조건들의 조합으로 인해서 각각 타일에 대한 특성을 파악하게 되고, 이를 통해서 전송될 이미지의 크기를 파악함과 동시에 메모리 어드레스 발생에 이용된다. $y_{size}$는 확장된 이미지의 타일의 세로 길이 값이고, $y_{size_org}$는 원본 이미지의 타일의 세로 길이 값이다. $x_{size}$는 확장된 이미지의 타일의 가로 길이 값이고, $x_{size_org}$는 원본 이미지의 타일의 가로 길이 값이다. 이 4가지 경우를 생성하는 이유는 확장된 부분의 공간을 확보하기 위함이다. ex_memory는 가상의 외부메모리이고, 입력 이미지가 저장된 공간이다. index는 메모리 어드레스의 값이며, 순차적 증가가 아닌 타일 형식의 순번을 가지고 있다. tile_data_line은 최종데이터가 메모리에서 나가는 곳이다.

## III. 시스템의 구성

그림 6는 타일링의 블록 다이어그램이며 하드웨어는 Verilog-HDL을 이용하여 설계하였다 [8]. RGB444YCbCr422 블록은 입력 영상 RGB를 YCbCr 4:4:4를 만들고, 내부 Down-sampling을 통해서 4:2:2로 변환한다. 그리고 외부메모리(Frame Memory)에 저장하기 위해서 결과 데이터를 타일링으로 넘긴다.. 이러한 방법을 통해서 최초 입력 영상의 데이터의 양을 2/3로 줄여서 메모리에 저장할 수 있다.

그러나 JPEG 2000 Encoding 과정에서는 RGB 3채널 데이터를 요구하기 때문에, YCbCr422YCbCr444 블록은 YCbCr 4:2:2을 Up-sampling하여 4:4:4형태로 최종 YCbCr 데이터를 만들어낸다.

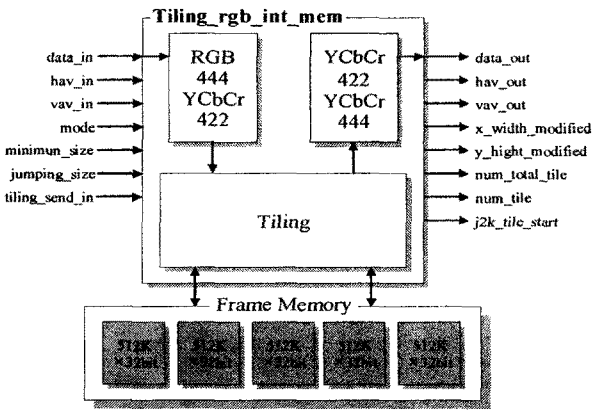


그림 6. 타일링 시스템의 블록 다이어 그램
Fig. 6. Block Diagram of Tiling system

타일링 블록은 입력으로 들어오는 최대 5M 크기의 데이터를 Frame Memory에 저장하고 이미지 데이터 흐름을 총괄하고 각 기능 블록들에 대한 제어 및 정보 신호를 발생 시킨다. 제어 신호인 j2k_tile_start는 새로운 타일이 입력되는 것을 하위 블록에 알려준다. 정보 신호인, x_width_modified와 y_hight_modified는 확장된 이미지의 가로 길이, 세로 길이를 하위 블록에 알려주며, num_total_tile, num_tile_index는 타일링 모드에서 전체 타일 개수와 타일 순번을 알려준다. Frame Memory는 SRAM 512K×32bit를 5개 사용하였으며, 5M 이미지를 처리할 수 있다. 여기서 512K는 depth를 뜻하고 32bit은 하나의 어드레스에 저장되는 용량인 width를 뜻한다. 이미지의 Progressive 형태의 입력과, 크기 또한 달라 질 수 있기 때문에 능동적인 모드 및 여러 가지 모드가 필요하다. 그래서 영상을 분할하지 않고 전체를 전송하는 Bypass 모드와 분할해서 전송하는 타일링 모드를 가지고 있으며, Frame Memory에 데이터를 저장하는 메모리 쓰기 및 이미지 상태 점검 및 확장 모드를 설정하였다. 이러한 모드 별 동작을 위해서 Finite State Machine (FSM)방식으로 설계하였으며, 제어 신호 및 어드레스 신호를 발생시켜서 메모리에 데이터를 저장 및 읽어냄을 수행한다.

FSM은 순차적인 디지털 회로의 상태(State)에 따른 처리 방법을 바꿀 수 있기 때문에, 여러 모드를 가지는 시스템에서 하위 블록을 제어하기 위한 제어 신호를 만들기에 적합하다. 시스템 내부에 현재 상태 레지스터(Current State Register)가 현재 단계 정보를 가지고 있으며, 이 레지스터의 값을 변화 시켜서 다음 단계로 변화 시킬 수 있다. 이러한 FSM를 설계하기 위해서는 상태 다이어그램(State Diagram)을 이용한다 [9]. 그림 7은 설계된 타일링의 상태 다이어그램이다.

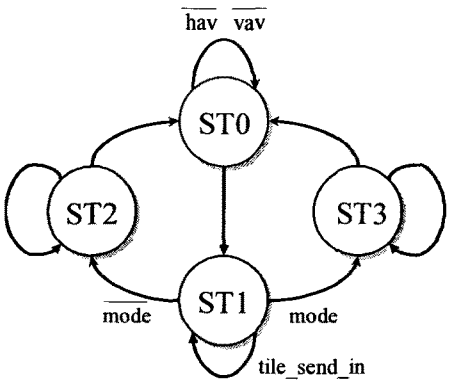


그림 7. 타일링 시스템의 상태 다이어 그램
Fig. 7. State Diagram of Tiling system

단계 0에서는 이미지의 영상 입력을 기다리면서 내부에 사용되는 레지스터들을 모두 다 초기화 시키는 단계이다. 동기신호 중에서 vav신호의 상승 에지를 검출하여 다음 단계로 움직이게 하는 단계이다. 단계 1에서는 입력되는 영상 데이터를 오는 순서대로 Frame Memory에 저장한다. 동기신호 hav와 vav가 0일 때 까지, 즉 1 Frame이 끝날 때 까지 이를 수행한다. 가로 그리고 세로를 세는 레지스터 2개를 사용하여 어드레스를 만들었으며, 동시에 이미지의 크기 또한 측정이 가능하므로 이를 통해서 가변적인 입력의 크기를 알 수 있다. 이미지 입력이 끝난 후에 내부에 지연 회로를 삽입하여 이미지 확장 연산을 수행한다. 내부에 작은 나눗셈기로 내부적으로 몇 개의 타일이 있는지를 구한 후 각각 타일들에 대한 기본 사이즈 정보 저장 레지스터에 타일의 기본 정보를 저장한다. 최종적으로 외부의 mode라는 변수 값에 따라서 bypass 또는 타일링 모드로 가도록 하였다. 단계 2에서는 bypass 모드로서 입력된 이미지를 다시 Progressive 형태로 출력한다. 이 때 출력되는 이미지의 크기는 확장된 이미지의 크기를 출력하고 그리고 blank 시간을 넣었다. 이미지를 모두 다 전송하고 나면 단계 0으로 가서 새로운 이미지를 기다린다. 단계 3에서는 타일링 모드로서 입력된 이미지를 타일 단위로 Progressive 형태로 출력한다. 이 때 출력되는 이미지의 크기는 확장된 이미지의 크기를 출력하고 각각 타일에 대해서 blank 시간을 넣었다. 이미지를 모두 다 전송하고 나면 단계 0으로 가서 새로운 이미지를 기다린다.

제안된 시스템은 Verilog-HDL을 이용하여 설계되었고, 0.25um TSMC cell library로 합성하였다. 표 4은 각각의 블록에 대한 gate count와 동작 속도를 나타내고 있다. 총 gate count는 2-input NAND 게이트를 기준으로 18,725개로 설계되었다.

표 4. 제안된 시스템의 gate counts와 최대지연 시간
Table 4. The gate counts and max delay timing of the system

| ModuleName | Gatecounts | Maxtimings(ns) |
|---|---|---|
| tiling_startb | 1,224 | 5.00 |
| rgb444ycbcr422 | 3,464 | 16.94 |
| ycbcr422ycbcr444 | 1,840 | 8.61 |
| tiling | 12,237 | 18.89 |
| **Total** | **18,725** | **18.94** |
| **FrameMemory** | 512K×32bit×5 = **10,485,760** byte | |

## IV. 결 론

본 논문에서는 이미지의 크기 파악, 영역 확장 그리고 이미지 분할 기능을 수행하고, JPEG 2000 표준을 만족하는 Verilog-HDL로 설계된 타일링 시스템을 제안하고 있다. Progressive한 입력을 타일 단위로 분할 및 이미지 전송을 위해서 1 Frame을 저장이 필요하며, 최대 5M 이미지를 처리할 수 있다. 그래서 Frame Memory를 제어 및 내부 제어 신호 발생을 위해서, 상태 다이어그램을 이용한 FSM 방식을 사용하였다. 이미지 크기 파악 및 영역 확장 알고리즘을 하드웨어로 구현을 위해서 나머지 연산을 기반으로 한 수식을 만들었으며, 이를 이용해서 입력 이미지의 크기 패턴을 파악하는 진리표를 제안하였다. 하드웨어로 구현된 JPEG 2000 Encoder는 소프트웨어 JPEG2000 Codec보다 기능면에서 특성화되고 고

속처리가 가능해진다. 그래서 폰 카메라, 디지털 카메라, 또는 PMP(Portable Multimedia Player)등의 각종 멀티미디어 휴대용 기기에 적용하여 압축효율을 높이고 좀더 좋은 화질을 획득 할 수 있다. 낮은 대역폭을 가지는 회선을 사용하더라도 빠르게 전송할 수 있고, 또한 화질측면에서도 JPEG과 같은 압축률을 사용하더라도 보다 좋은 영상처리결과를 얻을 수 있다.

## 참고문헌


[1] A. Skodras, C. Christopoulos, and T. Ebrahimi, "The JPEG2000 Still Image Compression Standard," IEEE Signal Processing Magazine, Vol. 18, Issues. 5, Sep. 2001

[2] 강봉순, *JPEG 2000 Hardwired Encoder 시스템 개발*, 최종연구보고서, 삼성전자, Dec. 2007.

[3] 우광제, *이차원 이산 웨이블릿 변환을 이용한 영상압축기의 설계 및 구현*, 석사학위논문, 동아대학교, 2003.

[4] ISO/IEC JTC1/SC29/WG1 N390R, "New Work Item: JPEG2000 Image Coding System," Mar. 1997.

[5] T. Acharya and P. Tsai, *JPEG 2000 Standard for Image Compression Concepts, Algorithm & VLSI Architecture*, John Wiley & Sons Inc, 2004.

[6] Information technology-JPEG 2000 image coding system : Core coding system, ITU-T T.800, 2002.

[7] 이민우, *Hardware Implementation of EBCOT Coder for Compressing DWT Coefficient with ISO/IEC 15444-1 JPEG 2000 International Standard*, 석사학위논문, 동아대학교, 2005.

[8] 장원우, 조성대, 강봉순, "JPEG 2000을 위한 tiling시스템의 설계" SoC 학술대회, pp.255~258 , May 2008

[9] D.J. Smith, *HDL Chip Design*, Doone Publications, 1996



**장 원 우(Won-woo Jang)**

2005년 2월 동아대학교 전기전자컴퓨터공학부 전자공학과(공학사)

2007년 2월 동아대학교 전자공학과(공학석사)

2007년 3월~현재 동아대학교 전자공학과 박사과정

관심분야 : VLSI algorithm/architecture design, image/video processing, and wireless communication.



**조 성 대 (Sung-dae Cho)**

1996년 숭실대학교 전자계산학과(공학사)

2000년 미국 Rensselaer Polytechnic Institute 전자컴퓨터공학(공학석사)

2002년 미국 Rensselaer Polytechnic Institute 전자컴퓨터공학(공학박사)

2004년 RPI 영상처리센터 박사후 연구원

2004년~현재 삼성전자 정보통신연구소 책임연구원

관심분야 : Multimedia image processing, color processing, computer vision, and data compression



**강 봉 순 (Bong-soon Kang)**

1985년 연세대학교 전자공학과(공학사)

1987년 미국 University of Pennsylvania 전기공학과(공학석사)

1990년 미국 Drexel University 전기 및 컴퓨터공학과(공학박사)

1989년~1999년 삼성전자 반도체 수석연구원

1999년~현재 동아대학교 전자공학과 부교수

2006년~현재 멀티미디어 연구센터 소장

2006년~현재 2단계 BK21 사업팀장

관심분야 : VLSI algorithm/architecture design, image/video processing, and wireless communication.