ISSN: 1226-7244 (Print)
ISSN: 2288-243X (Online)
논문번호 16-03-05

j.inst.Korean.electr.electron.eng.Vol.20,No.3,241~250,September 2016
http://dx.doi.org/10.7471/ikeee.2016.20.3.241

AVM 시스템의 하드웨어 구현에 따른 하드웨어 구조 및 메모리 대역폭 분석

Hardware Architecture and Memory Bandwidth Analysis of AVM System

남 광 민 *, 정 용 진*★

Kwnag-Min Nam*, Yong-Jin Jung*★

Abstract

AVM(Around View Monitoring) is a function of ADAS(Advanced Driver Assistance Systems), which provides a bird's eye view of the surroundings of a vehicle to the user. AVM systems require large bandwidth since they are composed of four input images and require real-time processing for vehicle-embedded environments. Also, the memory bandwidth requirement increases greatly when the resolution of the input data is higher. In this paper, we propose four basic hardware models of AVM systems. The models are decided by whether or not there is a valid data extraction module and an image processing purpose LUT generation module. We analyze the required bandwidth and hardware resource for each model. For verification of the proposed models, we implemented an AVM system using XC7Z045 FPGA and DDR3 memory for VGA and FHD resolution. All four of the proposed hardware model is executed below 33ms, which shows that it can operate in real-time.

요 약

AVM(Around View Monitor)시스템은 ADAS(Advanced Driver Assistance Systems)의 한 종류로 운전자가 차량 주변을 한눈에 파악할 수 있게 도와주는 차량 시스템이다. AVM 시스템은 네 개의 카메라에서 입력받은 데이터를 실시간 처리하기 때문에 요구되는 메모리 대역폭이 크다. 특히 입력 영상의 해상도 증가에 따라 메모리 대역폭 수치가 크게 증가하기 때문에, 필요한 메모리 대역폭에 맞는 하드웨어 구조 설계가 필요하다. 본 논문은 설계에 기틀이 될 AVM 시스템 하드웨어 모델 네 종류를 제시한다. 각 모델은 입력 영상으로부터 유효 데이터를 추출하는 모듈의 유무, 영상처리를 위한 LUT 생성 모듈 유무로 결정된다. 논문에서는 모델 별로 상이한 필요 메모리 대역폭과 하드웨어 자원 사용량이 제시된다. 이를 토대로 설계자의 요구 사항에 맞는 모델을 선택하고 구현할 수 있다. 제시한 하드웨어 모델의 검증을 위해 VGA, FHD급 AVM 시스템을 구현하였다. 구현에는 XC7Z045 FPGA, DDR3가 이용되었으며, 30FPS로 동작한다.

Key words : AVM, ADAS, Memory Bandwidth, Hardware Architecture, Image Processing

* Dept. of Electronics and Communications Engineering, Kwangwoon University

★ Corresponding author

yjjeong@kw.ac.kr / 02-940-5551

※ Acknowledgment

This work was supported by the IT R & D program of Ministry of Trade, Industry and Energy (10049192, Development of a Smart Automotive ADAS SW-Soc for a Self-Driving Car)

Manuscript received Aug. 31, 2016; revised Sep. 29, 2016 ; accepted Sep. 29, 2016

Ⅰ. 서론

ADAS(Advanced Driver Assistance Systems)는 운전자를 도와 보다 편리하고 안전한 운행에 도움을 주는 차량 시스템이다. AVM 시스템은 ADAS의 한 종류로, 차량 주변의 상황을 운전자에게 영상으로 제공한다. AVM 출력 영상에는 사각지대가 없기 때문에 좁은 공간을 지나거나 주차를 진행할 때 도움을 준다. AVM 시스템은 차

Fig. 1. AVM system input image
그림 1. AVM 시스템 입출력 영상

량에 설치된 네 개의 광각 카메라의 영상을 입력받는다. 입력 받은 영상은 일련의 과정을 거쳐 그림 1과 같이 차량의 위에서 수직으로 바라보는 조감도(bird's-eye-view)를 출력한다[1]. AVM 시스템은 LDWS(Line Departure Warning System), PGS(Parking Guidance System) 등의 ADAS에 응용될 수 있다.

빠른 속도로 이동하는 차량과 주변 상황을 고려할 때, AVM 시스템은 30FPS 이상의 실시간 동작 지원이 필요하다. 또한 네 개의 입력 영상 데이터를 처리해야하기 때문에 요구되는 메모리 대역폭이 크다. 특히 영상의 해상도가 높아질수록 데이터의 크기가 급격히 증가하는데, 표 1이 나타내는 바와 같이 FHD급 영상은 VGA급 영상의 6배 이상의 데이터 크기를 가진다. AVM 시스템에서 사용되는 입출력 영상 해상도의 표준 규격은 지정된 바가 없으며, 상용 AVM 시스템의 입력 영상은 VGA급에서 HD급까지 다양하다.

AVM 시스템의 출력 영상은 입력 영상의 각 픽셀이 특정 좌표로 매핑되어 생성된다. 이 때 매핑에 사용할 픽셀 좌표 정보는 입력되는 영상 정보와 관계없이 일정하므로 매번 계산하지 않고 LUT(Look-up Table)에 저장하여 읽어온다. 그러므로 LUT 생성을 위한 계산 과정보다 많은 양의 데이터 전송으로 인해 요구되는 메모리 대역폭 크기가 사양을 맞추는데 핵심이 된다.

Table 1. Image data size and resolution
표 1. 해상도 별 영상 데이터 크기

Type	Resolution	Data size
VGA	640x480	4.69 MB
HD	1280x720	14.06 MB
FHD	1920x1080	31.64 MB

Ⅱ. 관련이론

1. AVM 시스템 동작 과정

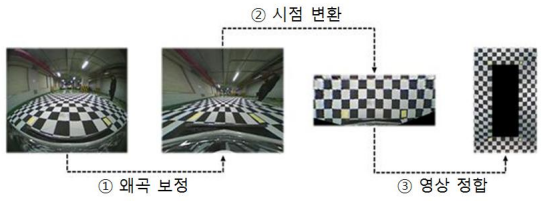

Fig. 2. AVM system process flow
그림 2. AVM 시스템 동작 흐름도

입력된 네 개의 입력 영상은 그림 2와 같이 크게 세 단계의 과정을 거치며 조감도 형태의 출력 영상이 만들어진다. 입력 영상은 광각 렌즈를 거쳐 만들어지므로 사물이 왜곡된 상을 띈다. 첫 번째 과정으로 왜곡된 영상을 평평한 영상으로 보정한다. 조감도를 만들기 위해 보정된 영상이 가진 카메라의 시점을 차량 위쪽으로 끌어올려야 한다. 두 번째 과정에서는 시점 변환을 거치며 선택한 일정 영역을 바라보는 조감도 영상을 만든다. 두 과정을 거쳐 만들어진 네 개의 영상을 출력 영상 크기에 맞게 배치시키고 하나의 AVM 출력 영상으로 만드는 것이 세 번째 과정이다[2].

가. 왜곡 보정 알고리즘

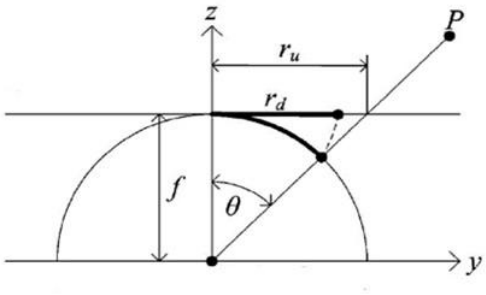

Fig. 3. Distortion correction algorithm model
그림 3. 왜곡 보정 알고리즘 모델

AVM 영상을 획득하기 위한 입력 카메라는 넓은 범위를 촬영할 수 있어야 하기 때문에 광각 렌즈 카메라를 사용한다. 그러나 광각 렌즈는 렌즈의 굴곡에 의한 왜곡특성을 가지고 있다. 본 논문에서는 왜곡 보정을 위하여 광각 사영함수 모델(fish-eye projection function model) 중 일반적으로 AVM 시스템 구성에 가장 많이 사용되는 Equidistant Projection 알고리즘을 사용하였다[3].

Equidistant Projection 왜곡 보정 알고리즘은 그림 3과 같이 렌즈의 곡면에 빛이 굴절되어 물체의 상이 이미지 평면 rd에 투영된다고 가정한다. rd에 맺힌 영상은 왜곡된 영상이며, 렌즈에 의한 왜곡이 없다면 ru에 상이 맺혀야 한다. f는 투영 중심점에서 이미지 평면 사이의 거리이며, f 값이 작을수록 왜곡 정도가 심해진다. 렌즈의 왜곡 보정을 그림 3과 같이 모델링 하면 rd의 좌표 값을 이용하여 ru의 좌표 값을 식 1로 계산하고 그 값을 구할 수 있다[3].

$$r_u = f \cdot \tan\left(\frac{r_d}{f}\right) \quad (1)$$

나. 시점 변환 알고리즘

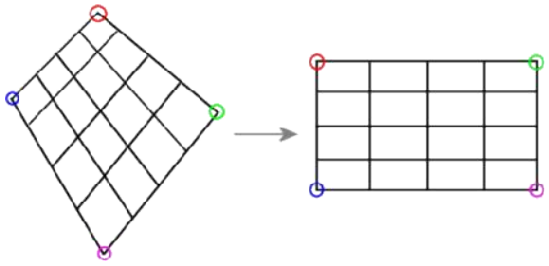

Fig. 4. Perspective transform algorithm model
그림 4. 시점 변환 알고리즘 모델

AVM 출력 영상은 차량을 위에서 아래로 내려다 볼 때의 시점으로 이미지를 만든다. 이러한 영상을 만들기 위해서는 일반적으로 동차좌표계[4] 행렬변환 식을 사용한다. 행렬 변환 수식은 식 2와 같다. 시점을 변환하기 위해서는 먼저 입력 영상의 임의의 4개 좌표를 정하고, 해당 좌표가 각각 출력 영상의 어느 포인트로 가는지 알아야 한다. 식 2에서 (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)가 입력 영상에서의 4개의 좌표이며, (x_1', y_1'), (x_2', y_2'), (x_3', y_3'), (x_4', y_4'), 는 입력 영상에서의 4개 좌표와 대응되는 출력 영상 좌표이다. 입력과 출력 영상에서의 각각 4개의 좌표를 정하면, 식 2를 이용하여 변환 벡터 [a, b, c, d, e, f, g, h]T를 구하게 된다[5].

$$\begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \\ x_3' \\ y_3' \\ x_4' \\ y_4' \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x_1' & -x_1'y_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y_1' & -y_1y_1' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2x_2' & -x_2'y_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2y_2' & -y_2y_2' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3x_3' & -x_3'y_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3y_3' & -y_3y_3' \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4x_4' & -x_4'y_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4y_4' & -y_4y_4' \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \end{bmatrix} \quad (2)$$

2. AVM 시스템 하드웨어 기본 구조

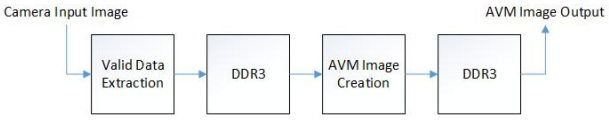

Fig. 5. AVM system hardware process flow
그림 5. AVM 시스템 하드웨어 동작 흐름도

AVM 하드웨어 분석에 앞서, 본 논문에서 사용하고 분석할 기본 구조를 그림 5로, 전체 시스템에서 사용할 DRAM과 FPGA 보드의 수는 각각 하나로 가정한다. 광각카메라로부터 입력 받은 4개의 영상에서 유효 데이터를 추출되어 DRAM에 저장된다. DRAM의 데이터를 읽어 AVM 영상을 생성하고, 다시 DRAM에 저장된 후 출력된다. 유효 데이터 추출, AVM 영상 생성 과정은 대응하는 좌표쌍의 LUT 매핑을 통해 이뤄진다. LUT는

DRAM에서 읽어오거나 생성기를 이용해 만들어진다.

해상도에 따라 데이터 크기 차이가 커서 구체적인 하드웨어 구조와 사양을 잡기가 쉽지 않다. 다음 장에서는 위 구조를 바탕으로 두고, 메모리 대역폭과 하드웨어 자원 사용량을 기준으로 세부 모델을 나눠서 해상도 별 하드웨어 자원 사용량과 메모리 대역폭을 분석한다.

III. AVM 하드웨어 모델 분석

1. 하드웨어 모델 구분

하드웨어 모델 분석을 위한 기본 구조는 그림 6의 구조로 한다. 하드웨어 구조 모델을 나누기 위해서 메모리 대역폭과 하드웨어 자원 사용에 가장 큰 영향을 미치는 두 가지 항목을 선정한다. 첫 번째 항목은 카메라를 통해 입력되는 데이터 중 실제로 사용될 데이터만을 메모리로 전송할 것인가, 혹은 입력 데이터 전체를 전송할 것인가의 여부이다. 두 번째 항목은 데이터 매핑에 사용할 LUT를 미리 저장된 특정 메모리 위치에서 읽어올 것인가, 혹은 LUT 생성기를 하드웨어로 구현할 것인가의 여부이다. 두 항목에서 어떤 선택을 했는가에 따라 그림 6과 같이 총 네 가지의 모델로 구분된다. 이후에 각 모델을 지칭할 때, 유효 데이터를 전송하고 LUT를 생성하면 VG(valid data, generated LUT) 모델, 유효 데이터를 전송하고 저장된 LUT를 읽어온다면 VS(valid data, stored LUT) 모델, 전체 데이터를 전송하고 LUT를 생성하면 RG(raw data, generated LUT) 모델, 전체 데이터를 전송하고 저장된 LUT를 읽어오면 RS(raw data, stored LUT) 모델로 줄여서 부르기로 한다.

유효 데이터를 전송하는 VG, VS 모델은 영상 입력 데이터가 DDR3 메모리에 저장되기 전에 해당 데이터를 뽑아내는 과정을 거친다. LUT 매핑을 위해 하드웨어 자원 사용량은 늘지만 메모리로 전송하는 데이터의 양이 대폭 줄어들기 때문에 메모리 대역폭을 줄일 수 있다.

VG, RG 모델은 사용할 LUT를 구성된 하드웨어로 직접 생성하고, VS, RS 모델은 메모리에 저장된 LUT를 읽어온다. LUT를 생성할 경우 메모리 대역폭이 줄어드는 이점을, 메모리에서 읽어올 경우 하드웨어 자원 사용이 줄어드는 이점을 가진다. VG, VS 모델은 2개의 LUT를 유효 데이터 추출, AVM 영상 제작 과정에서 각각 필요로 한다. RG, RS 모델은 AVM 영상 제작을 위한 하나의 LUT가 필요하며 VG, VS 모델의 LUT와 구조가 다르다. LUT의 자세한 구조는 3. 2절에서 설명한다.

어떤 모델을 선택했느냐에 따라 메모리 대역폭,

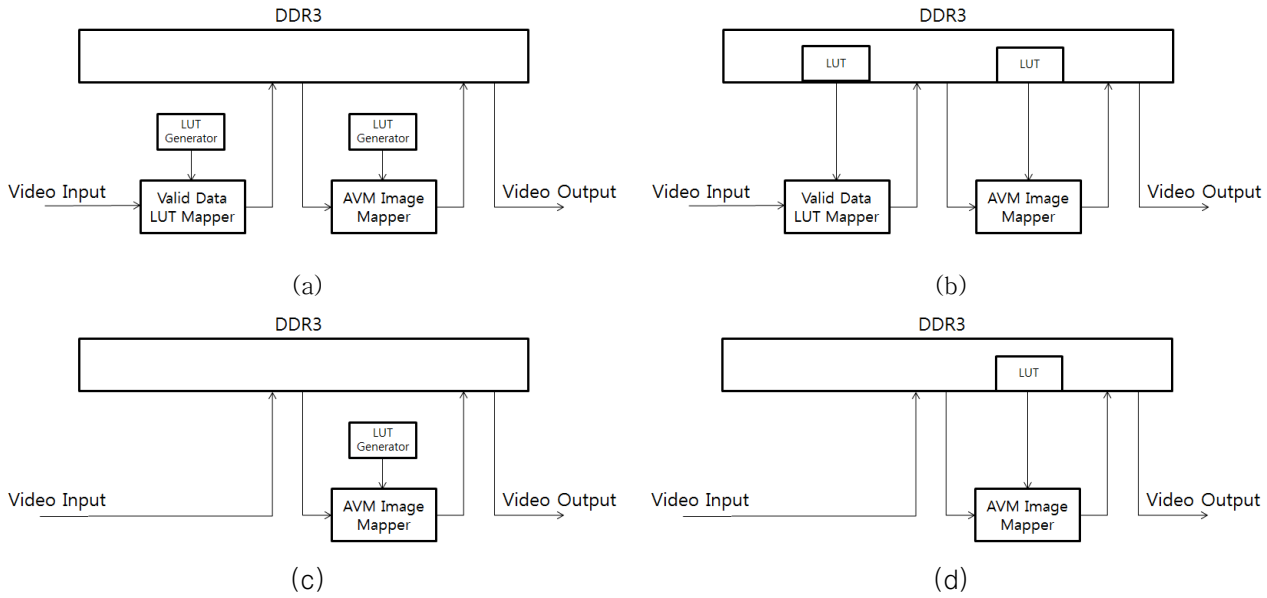

Fig. 6. Hardware model block diagram : (a) VG model, (b) VS model, (c) RG model, (d) RS model
그림 6. 하드웨어 모델 블록 다이어그램 : (a) VG 모델, (b) VS 모델, (c) RG 모델, (d) RS 모델,

Table 2. Required memory bandwidth of each model
표 2. 모델 별 필요 메모리 대역폭

모델명	내용	필요 대역폭		
		VGA	HD	FHD
VG	유효 입력 데이터 쓰기	18.3 MB/s	26.4 MB/s	36 MB/s
	유효 입력 데이터 읽기	18.3 MB/s	26.4 MB/s	36 MB/s
	AVM 데이터 쓰기	23.4 MB/s	39.3 MB/s	57.9 MB/s
	AVM 데이터 읽기	201.9 MB/s	201.9 MB/s	201.9 MB/s
	합계	261.9 MB/s	294 MB/s	331.8 MB/s
VS	LUT 읽기	39 MB/s	60.9 MB/s	87.3 MB/s
	유효 입력 데이터 쓰기	18.3 MB/s	26.4 MB/s	36.0 MB/s
	유효 입력 데이터 읽기	18.3 MB/s	26.4 MB/s	36.0 MB/s
	AVM 데이터 쓰기	23.4 MB/s	39.3 MB/s	57.9 MB/s
	AVM 데이터 읽기	201.9 MB/s	201.9 MB/s	201.9 MB/s
	합계	300.9 MB/s	354.9 MB/s	437.4 MB/s
RG	전체 입력 데이터 쓰기	140.7 MB/s	421.8 MB/s	949.2 MB/s
	전체 입력 데이터 읽기	35.6 MB/s	107.9 MB/s	242.7 MB/s
	AVM 데이터 쓰기	23.4 MB/s	39.3 MB/s	57.9 MB/s
	AVM 데이터 읽기	201.9 MB/s	201.9 MB/s	201.9 MB/s
	합계	401.6 MB/s	770.9 MB/s	1451.7 MB/s
RS	LUT 읽기	59.4 MB/s	78.3 MB/s	115.8 MB/s
	전체 입력 데이터 쓰기	140.7 MB/s	421.8 MB/s	949.2 MB/s
	전체 입력 데이터 읽기	35.6 MB/s	107.9 MB/s	242.7 MB/s
	AVM 데이터 쓰기	23.4 MB/s	39.3 MB/s	57.9 MB/s
	AVM 데이터 읽기	201.9 MB/s	201.9 MB/s	201.9 MB/s
	합계	461 MB/s	849.2 MB/s	1567.5 MB/s

하드웨어 자원 사용량이 달라진다. 표 2는 필요한 메모리 대역폭의 크기를 모델 종류, 해상도 종류에 따라 어떤 데이터 전송에 메모리 대역폭이 쓰이는가를 나타낸다. 데이터 전송의 종류는 LUT 읽기, 입력 데이터 쓰기/읽기, AVM 데이터 쓰기/읽기가 있다. LUT 읽기의 경우, VS 모델의 LUT가 두 개임에도 유효한 데이터만 다루기 때문에 LUT가 하나인 RS 모델보다 필요 대역폭이 더 작다. 유효 입력 데이터를 사용하는 모델은 메모리에 쓰고 읽는 크기가 동일하고, 입력 영상 크기에 비해 그 크기가 매우 작다. 그러나 전체 입력 데이터를 사용하는 모델은 메모리에 저장한 데이터 중 매핑에 필요한 특정 영역을 읽어가기 때문에 쓰고 읽는 크기가 다르며 요구되는 대역폭의 크기가 크다. AVM 데이터 쓰기는 정해진 AVM 영상 출력 크기에 맞게 고정되므로 모델 종류와 상관없이 해상도가 같다면 동일하다. AVM 데이터 읽기는 출력을 위한 모니터 해상도에 맞춰지므로 모델, 해상도 종류에 관계없이 모두 같은 대역폭을 갖는다.

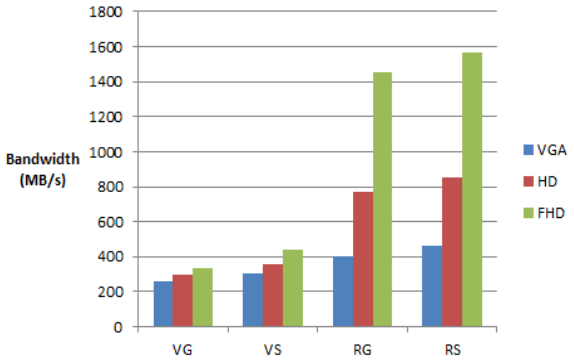

Fig. 7. Required memory bandwidth of each model
그림 7. 모델 별 필요 메모리 대역폭

그림 7은 대역폭 합계를 차트로 나타낸 그림이다. V계열 모델(유효 입력 데이터 사용)은 R계열 모델(전체 데이터 사용)에 비해 필요한 메모리 대역폭이 적으며 해상도가 증가할수록 대역폭 사용의 이점이 커진다. LUT를 생성해서 사용하면 약간의 메모리 대역폭 이점을 가지지만 유효 데이터 사용이 절대적인 영향을 미친다.

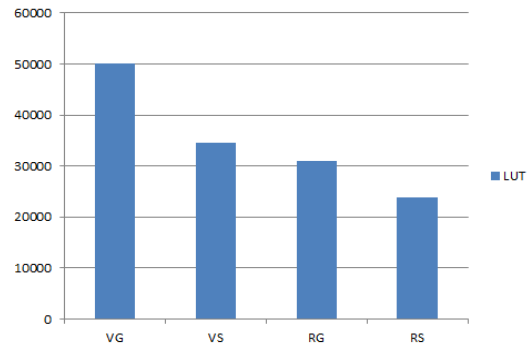

Fig. 8. LUT requirement of each model
그림 8. 모델 별 LUT 사용량

그림 8은 모델 별 LUT 사용량을 나타낸다. LUT 사용량은 LUT 매핑 모듈이 하나이고, 사용하는 LUT를 DRAM에서 읽어오는 RS 모델이 가정 적다. 반대로 LUT 매핑 모듈이 둘이고, LUT 생성 모듈을 가진 VG 모델의 LUT 사용량이 가장 크다. LUT 사용량은 LUT 매핑 모듈이 LUT 생성 모듈에 비해 크다. 사용하는 버퍼와 메모리의 크기, 논리 구조 형태는 RTL 구성과 사용하는 인터페이스에 따라 다를 수 있다.

하드웨어 자원 사용량 역시 LUT 매핑 모듈이나 LUT 생성 모듈이 많을수록 증가한다. 하드웨어 자원 사용량이 많은 모델이 메모리 대역폭은 적게 사용한다. 하드웨어 설계 구현 시 어느 부분에 우선순위를 두느냐에 따라 모델 선택이 달라질 수 있다.

메모리 대역폭 최소화를 우선한다면 VG 모델, 하드웨어 자원 사용 최소화를 위해서는 RS 모델을 선택하는 것이 좋다. 일반적인 경우 메모리 대역폭 요구와 하드웨어 자원 사용이 적은 VS 모델이 가장 적절한 선택이라고 볼 수 있다. 그러나 FHD급 AVM 시스템에서의 RG, RS 모델은 높은 메모리 대역폭이 필요하여 구현에 적절한 선택이 아니다. VGA급 AVM 시스템의 경우 모델 별로 메모리 대역폭 차이가 크게 나지 않아서 구현이 간단한 RG, RS 모델 구현이 좋다.

2. 하드웨어 구현

가. LUT 사용

본 AVM 시스템은 입력된 영상의 데이터를 선별적으로 재배치하여 원하는 형태의 출력 영상을 만드는 방법으로 구현된다. 차량에 설치된 카메라가 손상을 입지 않는 한, 입력 영상의 데이터가 결과 영상에서 출력되는 좌표는 동일하다. 그러므로 대응하는 입력영상 픽셀과 출력영상 픽셀의

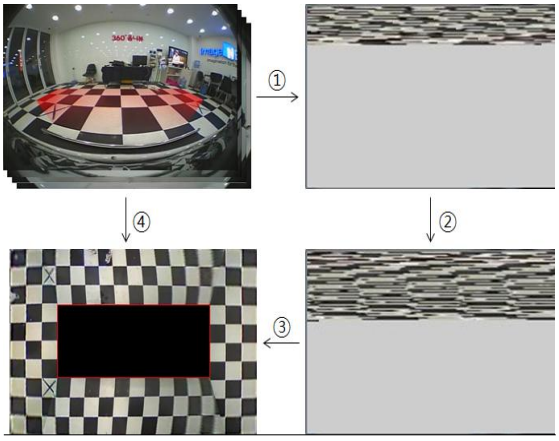

Fig. 9. LUT mapping process flow
그림 9. LUT 매핑 과정 흐름도

비트	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
내용	유효 데이터 위치															
비트	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
내용	유효 데이터 위치					예비										

(a)

비트	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
내용	DDR3 주소															
비트	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
내용	DDR3 주소					반복	예비									

(b)

비트	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
내용	AVM(입력) 영상 x 좌표											AVM(입력) 영상 y 좌표				
비트	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
내용	AVM(입력) 영상 y 좌표					예비										

(c)

Fig. 10. LUT description : (a) Valid data LUT, (b) DDR3 address LUT, (c) AVM coordinates LUT
그림 10. LUT 구조 : (a) 유효 데이터 LUT, (b) DDR3 주소 LUT, (c) AVM 좌표 LUT

좌표쌍을 LUT 형태로 저장해두면, 매 프레임마다 대응하는 좌표쌍 위치를 재연산할 필요 없이 지속적인 사용이 가능하다. 따라서 영상 데이터를 저장 혹은 생성한 LUT로 매핑하여 제시된 하드웨어 모델을 구현한다. 본 논문에서 사용되는 LUT는 총 네 가지 종류이다. FPGA 데이터 전송의 구현 편의를 위해 LUT는 32비트 단위로 정의한다.

그림 9의 ①번 과정에 해당하는 유효 데이터 LUT는 입력 데이터에서 실제로 사용되는 데이터 위치를 저장하고 있다. 그림 10. (a)의 LUT 구조를 가진다. 입력 영상의 픽셀을 차례로 카운트했을 때, 몇 번째 픽셀이 유효 픽셀인지 그 카운트 값을 21비트로 저장하고 있다. 유효 데이터 LUT가 매핑되면 카운트된 유효 데이터는 DDR3에 저장된다.

그림 9의 ②번 과정에 사용된 DDR3 주소 LUT는 DDR3에 저장된 유효 데이터가 AVM 영상에 출력되어야하는 순서대로 해당 유효 데이터의 DDR3 내부 주소를 가지고 있다. 영상 보간을 위해 같은 데이터가 반복 사용될 경우가 있는데, 반복 사용 여부 정보 또한 저장하고 있다. 그림 10. (b)의 LUT 구조를 가진다. 유효 데이터 주소는 21비트로, 반복 여부는 1비트로 저장된다. DDR3 주소 LUT를 적용하면 DDR3 내의 유효 데이터가 출력 순서에 맞춰 필요한 만큼 DDR3에 반복 저장된다.

그림 9의 ③번 과정에서의 AVM 좌표 LUT는 AVM 출력 영상 크기에 맞는 좌표를 저장한다. 그림 10. (c)의 구조로 각각 11비트씩 AVM x, y 좌표를 저장한다. 이 LUT를 매핑하면 최종 AVM 영상이 만들어진다.

그림 9의 ④번 과정에는 입력 좌표 LUT, AVM 좌표 LUT가 사용된다. 입력 좌표 LUT는 전체 입력 영상 좌표에서 출력될 AVM 영상에 쓰일 좌표 정보를 담고 있다. 입력 좌표 LUT는 AVM 좌표 LUT와 같은 구조이다.

유효 데이터를 사용하는 VG, VS 모델은 그림

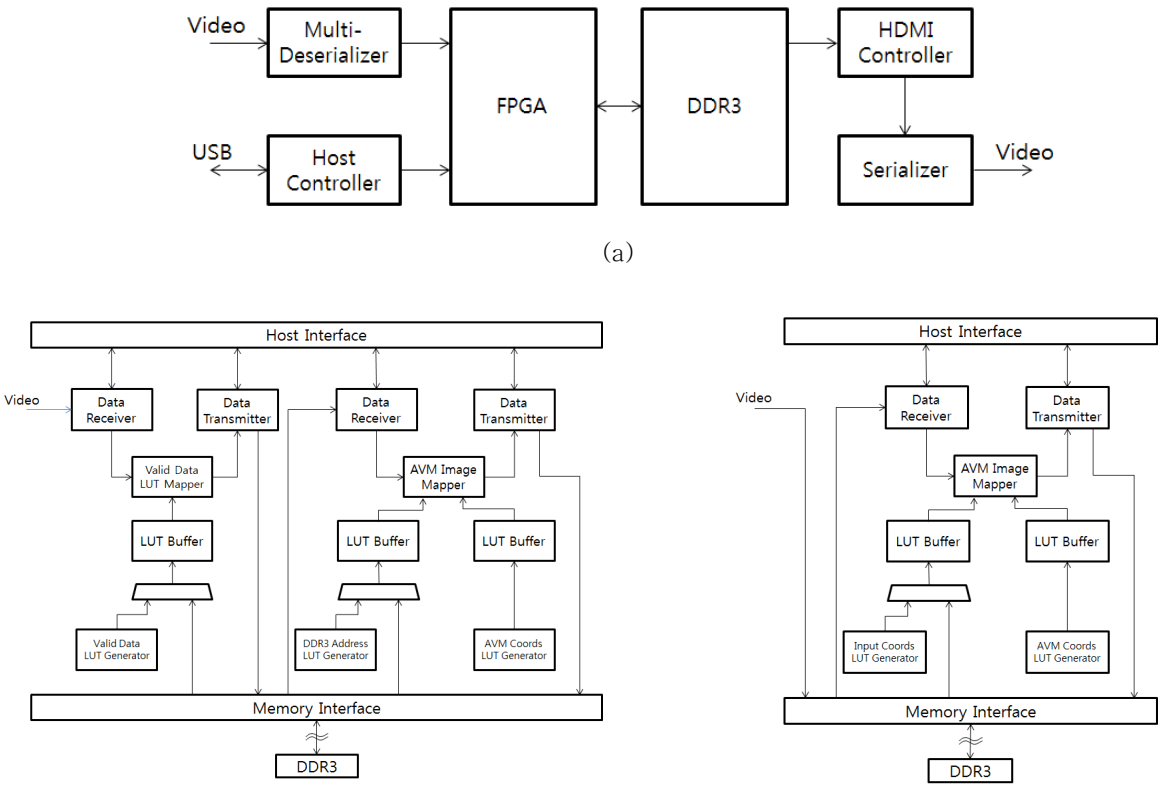

Fig. 11. AVM system block diagram : (a) Whole AVM system , (b) FPGA (VG, VS), (c) FPGA (RG, RS)
그림 11. AVM 시스템 블록 다이어그램 : (a) 전체 AVM 시스템, (b) FPGA (VG, VS), (c) FPGA (RG, RS)

9의 ①-②-③ 과정을 거치며 세 개의 LUT를 사용해 AVM 영상을 생성한다. 전체 입력 데이터를 사용하는 RG, RS 모델은 그림 9의 ④번 과정만을 거치며 LUT 두 개를 사용한다.

나. 하드웨어 구조

그림 11은 전체 AVM 시스템과 FPGA 내부의 하드웨어 구조를 블록 단위로 보여준다. 전체 AVM 시스템의 구조는 그림 11. (a)의 형태로 모든 모델이 동일하다. 촬영된 비디오 데이터는 디지털 데이터로 변환되어 FPGA에 입력되고, FPGA 내부에서 영상처리를 거쳐 만들어진 AVM 영상이 DDR3 메모리에 저장된다. HDMI 컨트롤러 모듈이 저장된 AVM 영상을 주기적으로 읽어 모니터에 출력한다. 호스트는 USB 입출력을 통해 FPGA 영역에 접근이 가능하다. 모델 별로 차이점을 보이는 부분은 FPGA 영역이므로 그림 9. (b), (c)을 통해 자세히 살펴보도록 한다.

그림 11. (b)은 VG, VS 모델의 블록 다이어그램이다. FPGA에 입력된 영상 데이터는 버퍼에 저장된 후, 유효 데이터 LUT가 매핑된다. LUT 정보는 생성되거나 메모리 읽기를 통해 버퍼에 저장되어 제공된다. 뽑힌 유효 데이터는 메모리 인터페이스를 통해 AXI4 버스로 FPGA 외부의 DDR3와 통신한다. DDR3에 저장된 데이터는 DDR3 주소 LUT, AVM 좌표 LUT 매핑을 위해 다시 메모리 인터페이스를 거쳐 읽기 동작을 한다. 매핑을 마치면 AVM 영상이 생성된다. 이 영상은 DDR3 내부의 HDMI 컨트롤러 프레임 버퍼 주소에 저장되어 영상처리 과정을 마친다. 좌표 LUT는 메모리에서 읽어오는 선택지를 두지 않았는데, 필요 메모리 대역폭 대비 하드웨어 자원 사용량이 매우 적기 때문이다. 출력 영상 크기에 맞춰 차례대로 영상 좌표를 생성하는 간단한 연산이다. 호스트 인터페이스를 통해 데이터의 입출력을 조정할 수 있다.

RG, RS 모델의 블록 다이어그램 그림 11. (c)의 구조를 가진다. VG, VS 모델과의 차이는 유효 데이터 추출을 위한 블록이 없다는 것과 LUT의 종류이다. 데이터 이동의 순서와 방법은 흡사하다.

Ⅳ 실험

제시한 AVM 시스템 모델의 검증을 위해 VGA급, FHD급 AVM 시스템을 각각 구현했다. 하드웨어는 Verilog HDL로 구현하였으며, 코드의 합성은 Vivado 2015.2를 사용했다. 설계된 로직은 Xilinx사의 XC7Z045 보드에서 구현하였고, 100MHz 클록으로 동작한다. 출력을 위한 AVM 영상 크기는 VGA급(460x640), FHD급(640x1040)으로 구현하였다.

하드웨어 사용량을 비교하기 위해 네 모델 중 가장 하드웨어 사용량이 큰 VG 모델, 사용량이 가장 적은 RS 모델을 구현하였다. 표 3은 RS 모델로, 표 4는 VG 모델로 VGA급 AVM 시스템을 하드웨어 구현한 결과이다. 두 모델은 60FPS로 실시간 동작한다. BRAM의 대부분은 DDR3에서 AVM Image Mapper로 읽어올 때의 버퍼로 사용되어 모델의 변화에도 큰 차이가 없다. 그러나 VG 모델이 유효 데이터 추출을 위한 추가 블록을 더 가지므로 RG 모델에 비해 FF과 LUT를 많이 사용한다. VG 모델의 LUT 사용량은 RS 모델의 약 2.1배로 앞서 예측한 각 모델 별 LUT 사용량과 일치한다.

Table 3. Hardware implementation result of RS model
표 3. RS 모델 하드웨어 구현 결과

Resource	Utilization	Available	Utilization(%)
Flipflop	23740	437200	5.43
LUT	23871	218600	10.92
BRAM	183.4	545	33.65

Table 4. Hardware implementation result of VG model
표 4. VG 모델 하드웨어 구현 결과

Resource	Utilization	Available	Utilization (%)
Flipflop	90212	437200	20.63
LUT	50129	218600	22.93
BRAM	275.1	545	50.48

표 5는 메모리 대역폭 검증을 위해서 구현한 VS 모델 FHD급 AVM 시스템을 비교하였다. [6]의 경우 6개의 DDR2, 5개의 FPGA 보드를 사용하여 FHD AVM 시스템을 구현하였다. 제안된 AVM 시스템의 경우 1개의 DDR3, 1개의 FPGA

보드를 사용했으며, 메모리 대역폭 사용이 350MB/s로 예측한 메모리 대역폭에 근접한 수치이다. 비교한 논문의 메모리 대역폭은 2188MB/s로 본 논문에서 제시한 모델이 대역폭 크기 면에서 뛰어난 성능을 가진다. VG 모델로 구현한다면 메모리 대역폭 사용을 조금 더 줄일 수 있을 것이다.

Table 5. Performance comparison of AVM system
표 5. AVM 시스템 성능 비교

	Proposed AVM	[6]
System spec	FPGA XC7Z045(x1)	FPGA XC6SLX45(x4) XC6SLX150T(x1)
Input Image	FHD	FHD
Frequency (Mhz)	100	133/ 166
Required memory bandwidth (MB/s)	350	2188

그림 12는 구현된 AVM 영상을 모니터로 출력한 모습이다. 실제 차량에 부착된 카메라의 영상을 입력하기 위해 SD 메모리를 통해 입력 영상을 전달하였다.

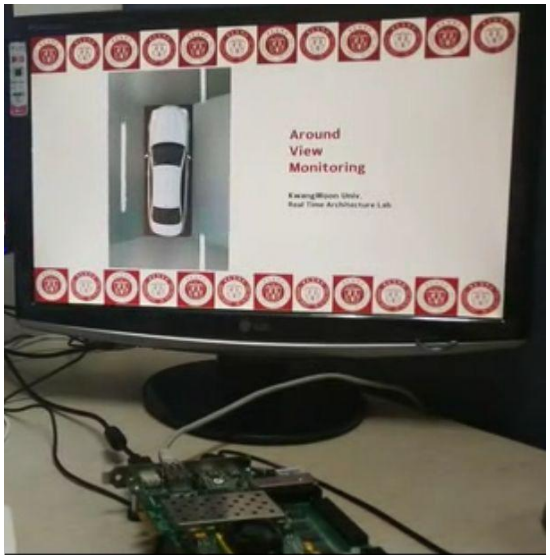

Fig. 12. Experiment environment
그림 12. 실험 환경

V 결론

본 논문에서는 AVM 시스템 구현에 참고가 될 하드웨어 구조와 메모리 대역폭을 해상도 별로 알아보았다. 검증을 위해 실제 차량에 설치된 어안렌즈로부터의 입력영상, 하나의 FPGA 보드와 DRAM을 사용해서 AVM 시스템을 구현했다. 타 논문과의 비교를 통해 보다 낮은 메모리 대역폭과 하드웨어 자원 사용으로 AVM 시스템 구현이 가능함을 알 수 있었다. 따라서 원하는 해상도의 AVM 시스템 설계 시 원하는 사양에 맞춰 구현에 유리한 구조를 참고할 수 있을 것이다.

향후 LDWS, PGS 등의 ADAS를 설계함에 있어 기반 기술로 사용 가능하며, 원하는 해상도에 맞게 AVM 시스템을 구현함에도 큰 도움이 될 것으로 판단된다.

References

[1] Yu-Chih Liu, Kai-Ying Lin, and Yong-Sheng Chen, "Bird's eye view vision system for vehicle surrounding monitoring," *International Workshop on Robot Vision*, pp. 207 - 218, 2008.

[2] Y. Chang, L. Hsu, and O. Chen, "AUTO-CALIBRATION AROUND VIEW MONITORING SYSTEM," *Vehicular Technology Conference*, pp. 1-5, 2013.

[3] H. Ciaran, D. Patrick, and J. Edward, "Accuracy of fish-eye lens models" *OSA Publishing*, Applied Optics Vol. 49, pp.3338-3347, 2010.

[4] Ki-Yong Lee, Joon-Woong Lee, "Ground Plane Detection Using Homography Matrix," *Journal of Institute of Control Robotics and Systems*, pp. 983-988, 2011.

[5] S. Ma, Q. Dong, M. Gao, "Motion and structure from an orthographic and a perspective image," *IEEE International Conference on Automation and Logistics*, pp. 833-837, 2008.

[6] B. Jeon, G. Park, J. Lee, S. Yoo, H. Jeong, "A Memory-Efficient Architecture of Full HD Around View Monitor Systems," *IEEE*

Transactions on Intelligent Transportation Systems, Vol.15, pp. 2683-2695, 2014.

BIOGRAPHY

Kwang-Min Nam (Student Member)

2016 : BS degree in Electronics and Communications Engineering, Kwangwoon University.
2016~ : Course of MS in Electronics and Communications Engineering, Kwangwoon University.

Yong-Jin Jeong (Member)

1983 : BS degree in Control and Instrumentation Engineering, Seoul National University.
1995 : MS, PhD degree in Electronics and Computer Engineering, University of Massachusetts, Amherst
1983~1989 : Research Engineer, ETRI
1995~1999 : Chief Researcher, Samsung Electronics
1999~current : Professor, Dept. of Electronics and Communications Engineering, Kwangwoon Univ.