실시간 영상확대 칩의 메모리 구조에 관한 연구

여 경 현° , 박 인 규
홍익대학교 전기제어공학과
121-791 서울 마포구 상수동 72-1
parkik@wow.hongik.ac.kr

A study on memory structure of real time video magnifyng chip

Kyung-hyun Yeo° , In-gyu Park
Dept, of Electrical & Control Eng. Honk Ik Univ.
72-1 Sangsu-dong Mapo-gu Seoul 121-791
parkik@wow.hongik.ac.kr

요약

본 논문에서는 영상확대 chip의 video 입력부에 부분화면을 저장할 frame memory의 구조를 개선하고자 하였다. 영상확대 video scaler인 gm833x2는 입력단 측에 frame buffer memory가 필요하게 되지만, 이를 외부에 장착하려면 일반적으로 대용량의 FIFO 메모리를 사용하게 된다. 이것은 dualport SRAM으로 구성이 되며, 메모리 제어를 고가의 FIFO칩에 의존하는 결과를 가져온다. 또한 기존의 scaler chip은 단순히 확대처리만을 하며, 입력 전, 후에 data의 변경 또는 이미지처리가 불가능한 구조가 된다.

본 논문에서는 외부에 필요한 메모리를 내장한 새로운 기능의 chip을 설계하는 데에 있어 필수적인 메모리제어 로직을 제안하고자 한다.

I. 서론

영상확대 chip의 video 입력부에는 부분화면을 저장할 메모리를 필요로 하게 되며, 비디오 신호의 특성상 대용량의 메모리를 필요로 하게 된다.

GENESIS gm833x2 chip은 하드웨어 FIR 필터를 사용하여 고품질의 이미지 확대를 효과적으로 수행한다. 또한 광학적 확대에 가까운 고화질, constant time delay의 실시간 처리능력을 가진 우수한 chip이다. 하지만, 다량의 메모리를 필요로 하며, 더 많은 video format을 지원하거나, 이미지 처리 시스템과 결합된 형태로 발전시키는 등의 응용이 어려워지게 된다.

여기서는 더 나은 기능의 향상된 메모리 제어회로를 제시하고 이를 One-chip에 집적할 수 있도록 하였다. 이를 사용한 Video Scaler Processor chip은 SDRAM을 별도의 제어회로 없이 외부에 장착할 수 있도록 하여 scaler의 기능을 향상시키면서 전체 시스템의 구조를 간단히 할 수 있을 것으로 기대된다.

본 논문에서는 먼저 메모리 제어회로를 포함한 Video Scaler Processor chip의 메모리제어 하드웨어의 구조를 제시하고, 메모리 access model과 제어로직을 소개하고자 한다.

II. gm833x2를 사용한 시스템

gm833x2를 기반으로 한 시스템은 다음과 같다.

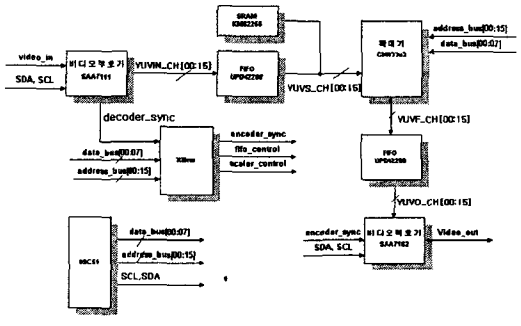

그림1. gm833x3을 기반으로 한 실시간 image 확대 system

gm833x2 확대기는 33 tab FIR 필터를 사용하며

하드웨어로 동작하므로 실시간 처리를 가능하게 한다. video 신호는 line 순으로 순차적으로 나타나기 때문에 33tab의 FIR filter를 가로, 세로로 한번씩 적용하기 위해서는 내부에 33x(처리할 화면의 너비) 만큼의 메모리를 필요로 하며 외부에서는 line을 반복하여 입력하여 주어야 한다. 이때 내부에서 Line Repeat 요구신호를 출력하도록 되어 있다. 또한, 입력화면의 크기와 출력화면의 크기를 설정하여주면 그에 따라 임의의 확대 배율로 확대가 된다.

decoder로부터의 입력 data는 YUV 4:2:2의 16bit 신호이며, 입력화면 크기는 720x480이며, interlace방식이다.

전체 화면중 확대할 부분을 사용자의 요구에 따라 선택하여 입력시켜 주어야 하므로 한 화면이 나오는 시간동안 저장된 부분화면을 입력시켜 주어야 한다. 그러므로 decoder로부터 입력되는 화면의 속도와 scaler로 입력해 주어야 할 화면의 속도는 같지만 같은 시간에 부분을 처리하므로 상대적인 입력속도의 차이가 나게 된다. 이를 위해 FIFO구조의 field memory를 사용하게 된다.

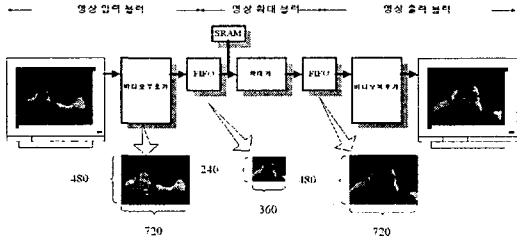

그림2. 기존 system에서의 FIFO의 역할

이 FIFO의 용량은 720x240x8bit=168kByte 이상이 2개 필요하게 되며, scaler chip의 요구사항에 따라 부호기쪽에서 74ns, 확대기 쪽은 약 37ns의 access time이 필요하게 된다. 또한 화면 갈라짐을 방지하기 위해 double buffering을 해야 하므로 앞서말한 168kB의 두 배인 336kB 이상의 메모리가 두개 필요하다.

또 data의 반복을 위해 repeat memory가 필요하다 이는 single port SRAM으로 사용하며 크기는 720 Byte 이상으로 37ns의 access time이 필요하며 Y와 UV용으로 2개 필요하다.

III. 제안한 메모리 구조

위의 기본적인 system에서도 입력부 FIFO가 336kB 이며, 입력지원 이미지의 크기를 확장시키는 경우 더욱 많은 메모리를 집적시켜야 한다. 또한 화면 갈라짐을 방지하기 위해 double buffering을 해야 하므로 두 배인 672kB 이상의 메모리가 필요하다.

또한 scaler 내부에도 tab수 x line길이 x pixel 만큼의 메모리가 필요하며, FIR의 구현에도 많은 로직이 필요하므로 외장할 수 있는 메모리는 외부에 놓는 것이 좋다. 대용량 FIFO대신 SDRAM을 바깥에 위치시키기 위해 하드웨어를 다음과 같이 재구성 하였다. 먼저 비디오 디코더와 스케일러 사이의 FIFO를 SDRAM을 사용하여 대체하도록 한다.

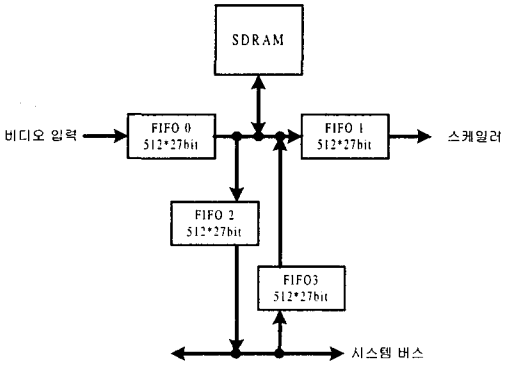

그림3. 새로운 구조에서의 frame 버퍼 메모리 구조

비디오 입력을 SDRAM에 저장하면서 동시에 필요한 스케일러에 입력시켜 주어야 한다. 또한 DSP등의 프로세서에서 data를 접근할 수 있도록 시스템버스측에도 data를 보내주어야 한다.

비디오 입력 pixel clock은 sync 신호, retrace 시간을 포함하여 실제 pixel을 읽는 속도보다 더 빠르게 동작한다. 여기서는 FIFO에 무조건 입력시켰다가 SDRAM제어 로직에서 입력부와 출력부의 pixel clock, 요구 기간등을 고려하여 data를 처리하는 방식을 사용한다.

입력시간은 비디오 디코더에서 나온 720x480의 30fps의 interlace 신호이므로 평균적으로 약 100ns에 한 pixel(16bit) 씩 처리하여야 하며, scaler 측에도 최고 100ns에 한 pixel의 data를 처리해야 한다.

SDRAM으로 저장하고 꺼내려면 늦어도 50ns 안에 SDRAM으로의 전송이 이루어져야 한다. SDRAM의 일반 mode 로는 50ns안에 access가 불가능하므로 여기서는 full-page burst mode로 접근을 하도록 한다. 이렇게하면 256 pixel을 전송하는 데에 clock이 80MHz인 경우 256*1/80㎒=3.2us 까지의 속도가 나오게 되며, 이렇게 하면 시스템 버스에서 data를 read/write 할 수 도 있는 넉넉한 시간을 얻을 수 있다. 그럼에도 FIFO 바깥에서 본다면 여전히 대용량의 FIFO를 사용하는 것 처럼 쉽게 접근할 수 있는 이점이 있다.

위의 구조로 기존의 시스템을 다시 구성하면 다음과 같이 구성될수 있다.

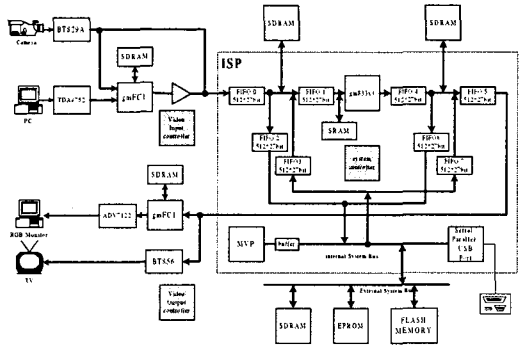

그림4. 전체 system 구조

IV. 메모리제어 방법

FIFO의 역할은 SDRAM의 full-page burst length 인 256만큼의 data를 넘치지 않게 보관해야 하므로 다른 FIFO를 access하는동안 대기하는 시간동안의 입력 pixel수에서 256만큼을 더한만큼의 크기여야 하며, 사용 clock의 사양에 따라 정한다. 여기서는 pixel clock이 13.5㎒ 이고 SDRAM clock이 80㎒라면 다음과 같다. 4개의 FIFO를 access 하는데 걸리는 시간은 256*1/80㎒*4=12.8㎲ 이므로 12.8㎲동안 입력되는 pixel수는 172.8 이다. 그럼 최소 필요한 FIFO의 크기는 173+256=429 이다.

FIFO에 data 가 256이상 차 있다면 SDRAM 전송을 하고 아직 data 가 차지 않았다면 다른 FIFO를 monitor 하는 방식으로 logic을 설계한다.

여기서 memory contorl logic이 SDRAM을 관리하는 방법을 먼저 제안한다.

행과 열의 계산을 용이하기 위해 메모리의 크기를 고정시키고, 위 구조에 알맞은 SDRAM의 memory access model을 제안한다.

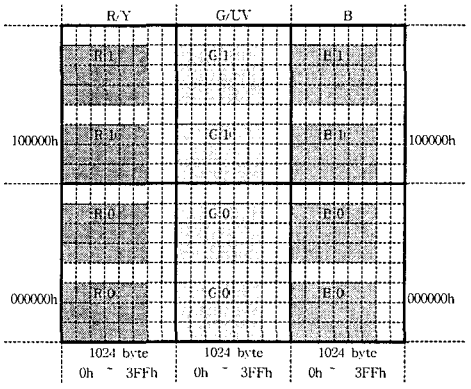

그림5. Memory Access Model

위 그림에서 아래부분(0~100000h)에 한 frame에 해당하는 data가 저장되며, 720x240만큼의 data가 각 두영역으로 (even,odd)저장되며, 현재 아래부분에 새로운 data 가 입력 중이라면, 윗부분에서 출력할 data를 읽어가게 된다. 이렇게 하면 한 화면에서 과거 frame과 현재 frame이 동시에 부분적으로 나타나는 frame tear현상을 막을 수 있다.

R,G,B 각각의 메모리는 pixel의 R,G,B color data 가 다른 pixel과 바뀌지 않도록 서로 같은 Address를 사용하여 동작하도록 한다.

가로축은 pixel 단위이며 세로축은 400h 씩 증가하기 때문에 아래 10bit를 버리면 line수가 된다.

V. 메모리 제어로직 설계

로직구성은 먼저 SDRAM전송을 시작할지를 판단하는 logic을 구성하고 여기서 나온 신호를 이용하여 실제 FIFO에서 SDRAM, SDRAM에서 FIFO로의 전송 logic을 설계한다.

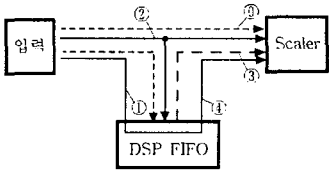

그림6. 전송가능한 경로의 수

먼저 전송가능한 경로를 살펴보면 그림6과 같으며, 각 경로의 조합으로 다음과 같은 mode를 설정하였다.

mode	source	dest.
0	FIFO0	SDRAM
	SDRAM	FIFO1
1	FIFO0	SDRAM
	SDRAM	FIFO2
2	FIFO0	SDRAM
	SDRAM	FIFO1
	SDRAM	FIFO2
3	FIFO3	SDRAM
	SDRAM	FIFO1
4	FIFO0	SDRAM
	SDRAM	FIFO2
	FIFO3	SDRAM
	SDRAM	FIFO1

표1. mode에 따른 전송

각 전송이 가능한지는 FIFO의 flag를 check하여 FIFO에서 256개의 data를 뽑을 수 있는지, 혹은 256개의 data를 입력시킬 수 있는지로 판별할 수 있다.

그림 FIFO0에서 SDRAM으로의 전송에 필요한 timing diagram을 그림7에 나타내었다.

FIFO는 IDT의 72211을 model로 하였으며, SDRAM은 SDRAM은 KM416s8030을 model로 작성하였다. 먼저 SDRAM에 쓸 수 있도록 command를

생성시킨다은 FIFO에서 256개의 data를 차례로 읽으면서 SDRAM에 입력시킨다.

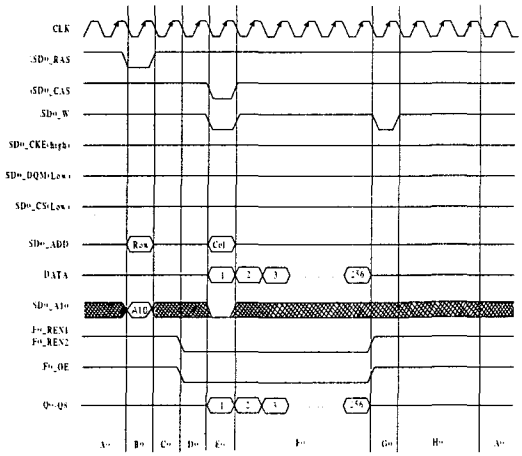

그림6. FIFO0에서 SDRAM으로의 전송 timing diagram

각 전송이 가능하면 그림6과 같이 전송을 시작하며, 위의 전송이 끝나고 나면 다시 전송판별 state로 돌아가게 된다.

각 전송에서 address에 해당하는 값은 자동으로 계산되며, 제어 logic 자체 register의 setting에 따라 변경가능하다. 아래에 화면 사양과 logic 동작에 필요한 register를 정리하였다.

레지스터 이름	description	비고
SD_Colstart[9:0]	CPU가 화면의 폭 시작점의 설정 값을 저장시켜 주는 Register	CPU로 로딩가능
SD_Rowstart[10:00]	CPU가 화면의 높이 시작점의 설정 값을 저장시켜 주는Register	CPU로 로딩가능
SD_Colend[9:0]	CPU가 화면의 폭 끝점의 설정 값을 저장시켜 주는 Register	CPU로 로딩가능
SD_Rowed[10:00]	CPU가 화면의 높이 끝점의 설정값을 저장시켜 주는Register	CPU로 로딩가능
SDRowi[10:0]	SDRAM1의 y축 어드레스 값을 저장시켜 주는Register	자동 reset
SDColi[9:0]	SDRAM1의 x축 어드레스 값을 저장시켜 주는Register	자동 reset

표2. FIFO3에서 SDRAM으로 전송시 필요한 register

외부 controller에서 입력 화면 크기와 출력화면 크기를 setting 한 후 memory control logic에 시작점 끝점에 대한 정보를 SD_Colstart와 같은 register를 통해 setting해 주면 scaler는 일정 영역의 data를 순차적으로 입력받을수 있게 된다.

VI. 결론

지금까지 비디오 scaler chip을 위한 SDRAM의 활용에대해 살펴보았다. 기존의 field memory(FIFO)의 사용은 전체 system을 간단하게 하지만 기능추가등의 확장이 어려워지는 단점이 있다. 본 논문에서 소개한 방법은 단지 메모리 제어 logic만을 포함시키기 때문에 one-chip에 집적시킬 수 있으며, DSP등의 프로세서가 data에 접근할 수 도 있게 하는 확장성을 가지게 된다. 이를 Video scaler와 One-chip에 집적한다면 SDRAM을 별도의 제어회로 없이 외부에 장착하는 것 만으로 frame buffer memory로 활용할 수 있으며 scaler의 기능을 향상시키면서 전체 시스템의 구조도 간단히 할 수 있을 것으로 기대된다.

참고문헌

[1] GENESIS, gm833x2B Datasheet, Mar. 1997

[2] CRC press, the Image Processing handbook, 1999