Journal of the Korea Academia-Industrial cooperation Society
Vol. 15, No. 4 pp. 2363-2369, 2014

http://dx.doi.org/10.5762/KAIS.2014.15.4.2363
ISSN 1975-4701 / eISSN 2288-4688

Full-HD급 PC기반 DVR System 구현을 위한 FPGA 활용에 관한 연구

김기화[1*]
[1]한국산업기술대학교 정보통신학과

A Study on FPGA utilization For PC-based Full-HD DVR System Implementation

Ki-Hwa Kim[1*]
[1]**Department of Information & Communications, Korea Polytechnic University**

요 약 DVR 시스템은 다수의 카메라를 지원하고 채널당 30프레임의 영상을 실시간으로 받을 수 있어야 한다. 따라서 Full-HD급 Multiplexer와 별도의 하드웨어 압축 Codec을 사용하는데, 본 논문에서는 이들을 사용하지 않고 FPGA와 CPU가 가지고 있는 GPU를 이용하여 4채널 Full-HD급 PC기반 DVR의 설계 및 구현 방법에 대하여 기술한다. Multiplexer와 H/W Codec을 사용하지 않는 기존의 방법으로는 실시간으로 채널당 20 프레임 정도의 영상을 획득하는 단점을 가지고 있다. FPGA를 이용하여 다채널의 영상을 실시간으로 획득하는 시스템을 설계하였으며, 소프트웨어로는 Intel Media SDK를 이용하여 영상 압축을 구현하였다. 구현된 제품의 성능 평가 결과, 제시한 요구 성능을 모두 만족하였고, 하드웨어 압축 코덱 디바이스를 제거함으로써 시스템의 실용성을 확인 하였다.

Abstract The DVR system supports multiple cameras and should be able to receive images at 30 frames per channel in real time. Thus, The system is using Full-HD-grade Multiplexer and Hardware compression codec. In this paper, Describing the design and implementation for the 4-channel Full-HD-grade PC-based DVR using FPGA and GPU inside CPU without Multiplexer and Hardware codec. The existing DVR system for Full-HD-grade has drawbacks to acquire images of about only 20 frames per channel in real time. The system to acquire images of multiple channel in real time was designed using FPGA. The software for the system was implemented using Intel Media SDK. At the result of performance evaluation, It was satisfied all for the required conditions. The practicality of the system was confirmed as implementation the system without using hardware compression.

Key Words : Digital Video Recorder, FPGA, Intel Media SDK

1. 서론

CCTV에서 영상의 해상도는 매우 중요한 요소이다. 사건, 사고 현장을 녹화한 영상의 오류나 현실감 부족으로 인한 사건 해결을 하지 못하는 상황을 줄여야 하기 때문에 형체를 파악 할 수 없거나 선명하지 못한 영상은 크게 도움이 되지 않는다.

CMOS센서의 발달로 CCTV 영상의 해상도는 지속적으로 증가할 것이며, 높은 해상도가 구현되기 위해서는 CCTV 카메라, 영상 저장장치, 통합관제 시스템, 영상 분석 기술 및 영상 보안 기술 등 주변 산업 환경이 동시에 발전해 나가야 한다. 뿐만 아니라 고용량 고해상도 영상을 실시간으로 전송 및 처리 할 수 있는 고성능 프로세서가 CCTV 시스템의 핵심이 될 것이다.

HD-SDI(High Definition Serial Digital Interface)는 HD급 방송장비간의 전송 규격으로 디지털영상신호를 압

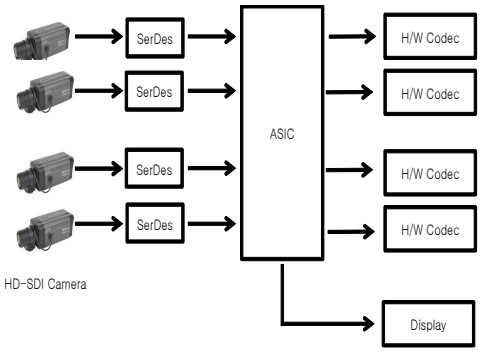
*Corresponding Author : Ki-Hwa Kim(Korea Polytechnic University)
Tel : +82-10-5337-4767 email: ed@sentry.co.kr
Received December 10, 2013 Revised (1st January 13, 2014, 2nd January 20, 2014) Accepted April 10, 2014

축하지 않고, 직렬 신호로 변환하여 동축케이블로 전송하는 방식을 말한다[1]. HD-CCTV는 HD-SDI를 이용하여 고해상도 영상을 압축을 하지 않고 전송을 하기 때문에 압축에 따른 영상 손실이 발생하지 않고, 영상을 아날로그가 아닌 고속의 디지털로 전송하므로 열화에 의한 영상의 노이즈도 존재 하지 않는다. HD-SDI는 기존 아날로그 영상을 전송하는 방식인 동축케이블을 그대로 사용이 가능하지만, 1080p30의 Full-HD 영상 데이터를 고속으로 전송하기 때문에 기본 전송 거리는 아날로그 영상 전송 거리 보다 절반 수준이다. 하지만, 디지털 전송 방식이기 때문에 전용 리피터를 사용하면 영상의 손실 없이 무한대 거리의 영상을 전송 할 수 있는 장점이 있다. Table 1은 HD-CCTV Alliance에서 HD-SDI 규격을 보완하여 발전시킨 전송 규격이다.

[Table 1] HD-SDI Transmit Rule in HD-CCTV Alliance

Ver 1.0	720p25/30, 720p50/60, 1080p25/30(2 mega Pixel)
Ver 2.0	Bi-direction Meta Data & Audio Transmition
Ver 3.0	8 Mega Pixel, 720p240 Higher

HD-CCTV 영상을 초고속으로 전송하고 처리하기 위해 기존의 DVR 시스템에서는 Fig 1과 같이 HD급 영상을 획득 할 수 있는 시리얼 영상데이터를 패러럴로 바꾸어 주는 SerDes 소자와 ASIC화 한 Full-HD급 멀티플렉서(Multiplexer), 그리고 채널 당 1개의 하드웨어 압축 소자를 사용해 왔다.

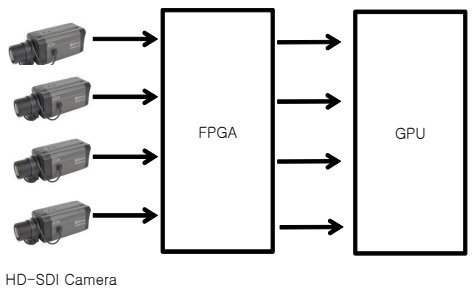


[Fig. 1] Typical Multi-channel DVR System Configuration

채널수에 따라 압축 전용 코덱 소자를 사용해야 하기 때문에 다채널 고해상도 DVR 시스템을 구현하기 위해서는 시스템 자체가 커져야 하고, 가격적인 부담과 시스템 구현에 어려운 문제가 발생한다.

CPU(Central Processing Unit) 제조사들은 현재 CPU의 클럭 주파수를 높이는 것에 집중하기 보다는 내부 코어(Core) 수를 늘리고 그 코어간의 고속 데이터 버스를 구성 하였다. 멀티 코어 형태의 CPU 구조를 갖추어 그 연산을 병렬 처리 구조로 함으로써 성능을 향상해 나가고 있다. 또한, CPU내에 GPU(Graphic Processing Unit)를 내장하여 CPU가 처리해야 하는 데이터를 GPU에서 처리하게 분산시킴으로써 CPU의 부담을 줄이고 데이터를 처리하는데 소요되는 시간을 줄여 처리 능력을 높이고 있다.

이와 같은 PC 시스템 성능의 발전과 더불어본 논문에서 제시한 시스템은 Fig. 2와 같이 외부의 SerDes와 ASIC 멀티플렉서 디바이스를 대체하여 하나의 FPGA(Field Programmable Gate Array)를 사용함으로써 시스템 구성이 간단해지고, 하드웨어 압축 소자를 대신하여 CPU에 내장된 GPU를 사용함으로써 기존 시스템에서 사용되지 않는 불용의 GPU 리소스를 활용하게 된다. 또한, GPU의 성능으로 기존에 20프레임 정도의 압축 성능을 30프레임으로 개선시킴으로써 영상 압축과 저장에 따른 화질 저하를 막을 수 있다.



[Fig. 2] FPGA & GPU Codec Block Diagram

이러한 PC 시스템의 변화를 적용하여 본 논문에서는 FPGA와 GPU를 이용하여 실시간 영상 획득과 하드웨어 압축 코덱의 기능을 대체한 시스템을 제안하고 설계, 구현 하였다.

2. 본론

2.1 일반적인 CCTV에서의 영상 처리 방법

HD-SDI 카메라로 부터 획득 되는 영상 데이터는

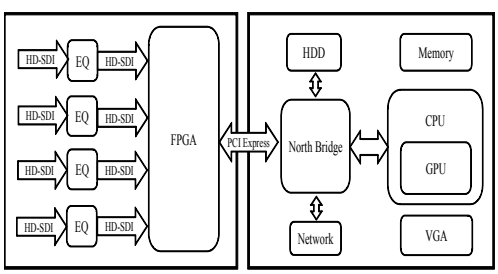
YUV4:2:2 포맷 형식을 따른다[2]. Full-HD급 영상인 1080p30은 132Mbyte/sec의 크기의 YUV형식 데이터를 가지고 있다. CCTV에서 사용되는 일반적인 카메라는 SD급과 마찬가지로 HD급도 YUV4:2:2 영상 포맷을 가진다. YUV4:2:2는 휘도(Y) 성분 4개에 색차 성분U, V 성분을 2개씩 가지는 포맷으로 텔레비전에서 사용하는 영상 포맷이다. YUV4:2:0은 YUV4:2:2에서 사람의 눈이 색차 성분에 둔감하다는 것에 착안하여 색차 성분의 U, V 값을 줄임으로써 처리해야 하는 영상 데이터의 사이즈를 줄이는 방법이다.

CCTV 관련 시스템에서 사용하고 있는 영상 코덱들 중에는 H.264 압축 코덱 형식을 가장 많이 쓰이고 있다. H.264 압축 방식은 기존 MPEG4 Part-10이라 하여 MPEG4에서 파생되어 진화한 압축 코덱이다. 영상의 기준이 되는 프레임을 I(Intra), P(Predictive), B(Bidirectional) 프레임으로 구분하며, I 프레임과 P 프레임 모두를 참조하는 B 프레임을 추가 한 것이 기존 MPEG4와의 차이점이다[3]. 이러한 방식으로 압축에 대한 효율을 높였고, 압축 처리 하는 연산량이 많아지는 단점이 있지만, MPEG4 보다 30% 정도 압축률이 더 좋아져 저장 공간을 줄이는 이점과 영상 화질의 차이로 현재 대부분의 CCTV 제품에서 사용하고 있다.

2.2 실시간 영상 처리를 위한 하드웨어 시스템 설계 및 구현

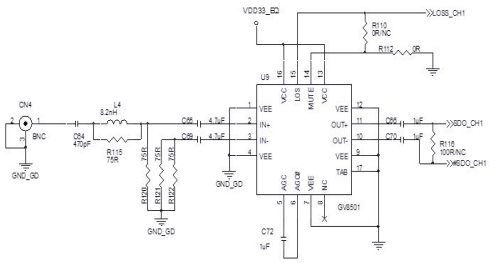
2.2.1 하드웨어 구성

본 논문에서 제안하는 FPGA를 이용한 Full-HD급 DVR 시스템은 Fig. 3과 같이 HD-SDI 영상을 수신하는 이퀄라이저와 FPGA로 구성되다. FPGA 영상 획득 보드는 HD-SDI 카메라와 동축케이블(Coaxial Cable)을 통해 이퀄라이저로 입력된다. 입력된 영상은 FPGA로 전달되고 획득한 Full-HD 영상은 PCI Express Bus를 통하여 GPU로 전송하여 영상을 압축하고 저장하게 된다.



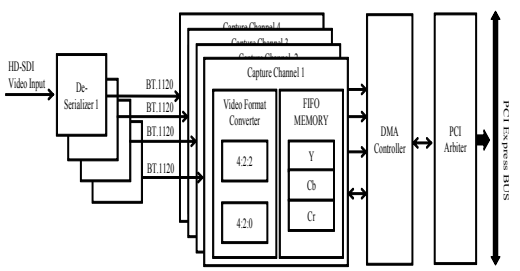
[Fig. 3] FPGA DVR System Block Diagram

HD-SDI 카메라로부터 원거리 전송에 따른 신호의 감쇄를 보상하기 위해 Fig. 4와 같이 이퀄라이저를 사용하여 신호를 보상하였다. 원거리 전송 시 발생되는 신호의 감쇄와 불안정한 동축 케이블의 이음으로 Jitter가 발생될 수 있다. Jitter로 인해 획득되는 영상 신호의 왜곡을 이퀄라이저를 이용하여 제거하고 Re-Clocker로 1.485Gps의 정상적인 HD-SDI 신호로 다시 만들어줌으로써 FPGA에서는 깨끗한 디지털 영상을 획득 할 수 있었다[4].



[Fig. 4] Equalizer Circuit Design

본 논문에서 사용한 FPGA는 Lattice사의 ECP3 계열의 LFE3-70EA이다[5]. LFE3 디바이스는 8개의 SerDes를 가지고 있으며 이 중에 4개는 HD-SDI 영상 신호를 수신하는 De-Serializer로 사용하고 나머지 4개는 PCI Express 4 Lane을 구성하기위해 Serializer로 사용하였다[6]. 영상 수신용 De-serializer는 입력되는 고속의 시리얼 비디오 데이터를 패러럴 데이터로 변환하는 기능을 담당한다. 변환된 영상 데이터를 H.264압축 방식으로 압축하기 위해 Fig. 5와 같이 BT.1120 형식으로 수신되는 YUV4:2:2 데이터를 YUV4:2:0 포맷으로 변환하고, 변환된 영상 라인을 내부 FIFO Memory에 적재하는 기능을 하는 Video Format Converter를 FPGA 내부에 구현하였다[7].



[Fig. 5] Video Format Converter

본 시스템은 4채널로 구성하였기 때문에 Video Format Converter는 총 4개의 Capture Channel을 가지고 있다. Capture Channel은 각각을 독립적으로 구성하여 획득 되는 영상은 채널의 순서와 상관없이 먼저 획득되는 채널을 우선으로 처리 하고, 처리된 영상은 DMA(Direct Memory Access) Controller를 이용하여 각각의 영상을 압축하기 위해 할당한 시스템 메모리로 전송하여 GPU를 통한 압축과 Video Card로 전달하여 디스플레이 한다. 본 논문에서는 CPU내에 있는 GPU를 사용하여 압축하는 방식이기 때문에 Table 2에 기술한 CPU를 사용하여 시스템을 구성하였다.

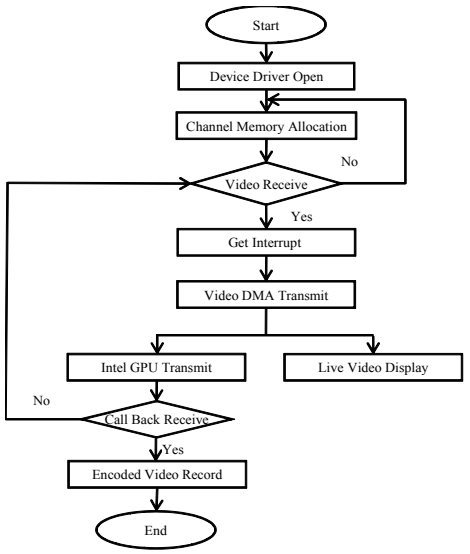
2.2.2 소프트웨어 구현

FPGA 영상 획득 보드를 PC 시스템에서 사용 하기위한 PCI Express Bus 장치를 제어하기 위해 WDM 기반으로 디바이스 드라이버를 구현 하였다[8]. Fig. 6은 Windows OS에 장치를 등록하기 위한 드라이버 소스 프로그램의 일부이다. 장치에 대한 디바이스 이름과 해당 장치를 사용하기 위해 메모리 할당을 요구한다.

```
NTSTATUS NC_AddDevice(IN PDRIVER_OBJECT DriverObject, IN
    PDEVICE_OBJECT PhysicalDeviceObject)
{
    NTSTATUS              status = STATUS_SUCCESS;
    PDEVICE_OBJECT        deviceObject = NULL;
    PDEVICE_EXTENSION     pdx;
    UNICODE_STRING        ntDeviceName;
    WCHAR  DeviceNameBuf[0x100], win32NameBuf[0x100];
    memset(DeviceNameBuf, 0, 0x100*2);
    memset(win32NameBuf, 0, 0x100*2);
    g_dwDeviceCount = InterlockedIncrement(&g_DevCount);
    _snwprintf(DeviceNameBuf, arraysize(DeviceNameBuf),
    NT_DEVICE_NAME, g_dwDeviceCount);
    RtlInitUnicodeString(&ntDeviceName, DeviceNameBuf);
    status = IoCreateDevice(DriverObject, sizeof(DEVICE_EXTENSIO
             N), &ntDeviceName, FILE_DEVICE_UNKNOWN, FILE_
             DEVICE_SECURE_OPEN, FALSE, &deviceObject);
    if(!NT_SUCCESS (status))
    {
        g_dwDeviceCount = InterlockedDecrement(&g_DevCount);
        return status;
    }
```

[Fig. 6] Add Device in WDM Device Driver

Fig. 7은 시스템에서 영상을 획득하는 소프트웨어 흐름을 나타낸다. 프로그램을 실행시키면 각 채널의 영상을 획득하여 저장하는 메모리를 할당하고, 영상에 대한 유무를 판단한다. FPGA로부터 영상에 대한 인터럽트를 받으면 각 영상을 할당된 메모리로 DMA 하여 영상을 획득한 후 압축을 하기위해 GPU로 전달하고, 모니터에 표출하기 위해 그래픽 카드로 전송한다. GPU로부터 영상 압축이 완료되어 인터럽트를 수신하면 하드 디스크에 압축된 영상 데이터를 저장 한다.그리고 저장된 영상을 복원하기 위해서는 날짜 및 시간별로 저장된 압축 영상을 영상 인덱스를 참조하여 GPU로 전달하고, 복원된 영상은 그래픽 카드로 전달하여 저장된 영상을 확인하는 방식으로 구현하였다.



[Fig. 7] Video Process Flowchart

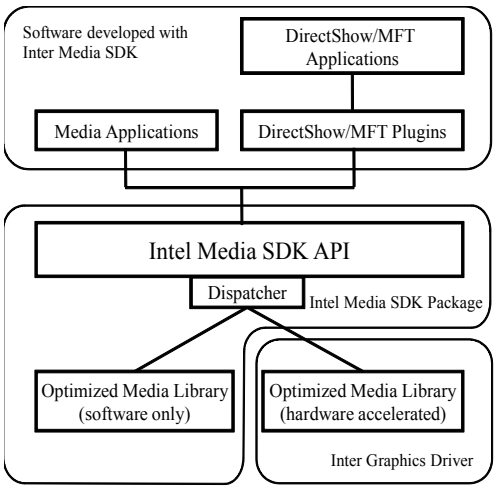
```
BOOL Interrupt_HDVCap(PDEVICE_EXTENSION pdx, DWORD
dwHDVideoStatus, DWORD dwVideoLoss)
{
    BYTE    byCurQCnt, byNextQCnt, byChannel, byHDVideotype,
            byField;
    DWORD   dwField=0, dwData;
    DWORD   dwReadDMAAddress, dwWidth, dwHeight;
    BOOL    bRet = FALSE;
    PVCAP_STATUS_INFO pVCapStatus;
    for(byChannel=0; byChannel<pdx->m_byMaxHDChannel;
    byChannel++) {
      if(((pdx->m_dwVHD_DMAEnable>>byChannel)&0x1) &&
       ((dwHDVideoStatus>>byChannel)&0x1))
      {
        byCurQCn t = pdx->m_byHDVCapQueue[byChannel];
        pVCapStatus = pdx->m_pHDVCapStatus[byChannel]
                      [byCurQCnt];
        if(pVCapStatus->byLock == MEM_UNLOCK)
        {
         dwReadDMAAddress = dwReadRegister(pdx,REG_HD_VIDEO
                            _TYPE);
         byHDVideotype = (BYTE)((dwReadDMAAddress>>
                         (4*byChannel))&0xf);
          if(byHDVideotype<=0x4)
          {
            dwWidth = 1280;
            dwHeight = 720;
            byField = 0;
          }
          else if((byHDVideotype>=0x8) && (byHDVideotype<=0x9))
          {
            dwWidth  = 1920;
            dwHeight  = 1080;
            byField   = 1;
```

```
            }
            else if((byHDVideotype>=0xA) && (byHDVideotype<=
            0xC)) {
                dwWidth   = 1920;
                dwHeight  = 1080;
                byField   = 0;
            }
            pVCapStatus->byLock              = MEM_LOCK;
            pVCapStatus->dwWidth             = dwWidth;
            pVCapStatus->dwHeight            = dwHeight;
            pVCapStatus->byField             = byField;
            pVCapStatus->byChannel           = byChannel + pdx->
                  m_byDeviceID * pdx->m_byMaxHDChannel;
            pVCapStatus->dwTick              = pdx->dwTick;
            KeQuerySystemTime(&pVCapStatus->VCAPTime);
            byNextQCnt = (byCurQCnt+1)%MAX_HDVIDEO_QUEUE;
            SetHDVideoDMATargetAddress(pdx, byChannel,byNextQCnt);
            dwWriteRegister(pdx,REG_VHD_CAPTURE_LENGTH(
                  byChannel), dwWidth*dwHeight);
            pdx->m_byHDVCapQueue[byChannel] = byNextQCnt;
            KeInsertQueueDpc(&pdx->m_DpcObjectHDVCap, NULL,
            pdx);
            bRet = TRUE;
            }
          }
        }
            return bRet;
    }
```

[Fig. 8] Full HD Video Capture Program

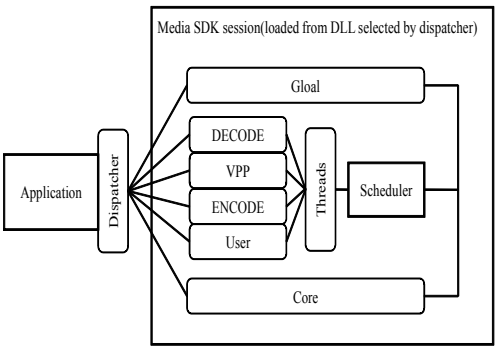
Fig. 8은 Full HD급 영상을 획득하기 위한 프로그램으로 영상의 해상도 크기와 채널을 구분하고 DMA 를 이용하여 영상을 메인 메모리에 적재하는 기능을 수행한다.

하드웨어 압축 코덱 디바이스를 대체하는 GP U를 사용하기 위해서는 Intel사에서 제공하는 Intel Media SDK라는 라이브러리가 필요하다. 이것은 Intel CPU내에 있는 GPU를 핸들링 할 수 있는 고유 함수를 가지고 있어 영상 압축과 복원 및 영상처리에 따른 GPU를 제어 하는데 사용한다[9].



[Fig. 9] Intel Media SDK

Fig. 9에 있는 미디어 디스패쳐는 미디어 SDK의 관문으로 미디어 SDK를 이용하는 프로그램을 연결하는 역할을 한다. 다양한 인코딩과 디코딩 그리고 비디오 전처리 기능을 위한 진입점으로 사용되고, 다중스레드를 이용하여 여러 개의 입력과 출력을 처리 할 수 있다.



[Fig. 10] Media SDK Session

또한, Fig. 10에서 보는 바와 같이 Intel Media SDK는 영상의 Encode 뿐만 아니라 압축을 하기위한 전처리 과정이 필요한 VPP(Video Pre- processing)와 Decode 기능을 사용할 수 있고, 사용자가 임의로 사용 하고자 하는 영상 처리에 대한 부분을 지원하고 있다.

2.3 실험 및 고찰

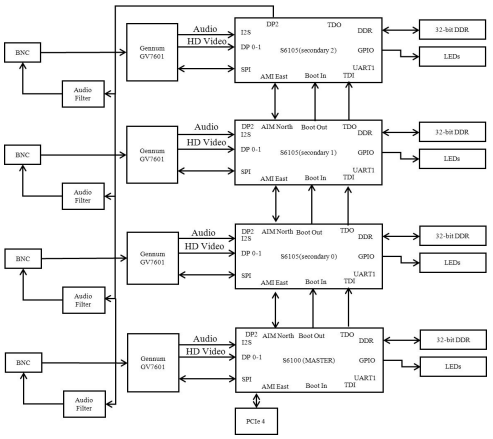
본 논문에서 제시한 시스템의 성능을 검증하기 위해 Table 2와 같은 시스템 기반으로 하드웨어 압축 방식의 보드와 성능 비교를 하였다.

[Table 2] System Specification

Separator	Specification
CPU	Intel Sandy Bridge i7-2600
VGA	Inner VGA (Intel HD2000 Family)
Main Board	ASUS P8H61-MLE
Memory	4GB

하드웨어 압축 방식 시스템은 스트레치사의 S6000기반의 Sentry9004HD 보드이다. S6000 디바이스는 HD급 영상인 720p 영상은 30 프레임의 실시간 압축이 가능하지만, Full-HD급 1080p 영상은 20 프레임 정도 압축 성능을 가지고 있다[10]. Fig. 11과 같이 HD-SDI 영상 신호

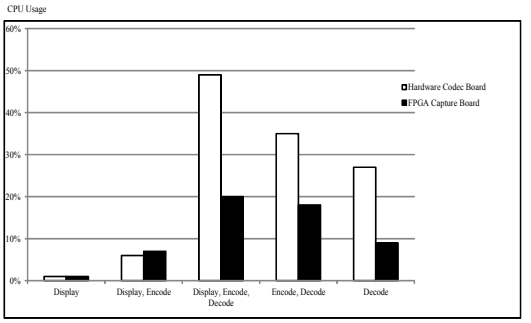
를 수신하고 시리얼 비디오 데이터를 병렬 데이터로 바꾸기 위해 별도의 SerDes칩으로 Gunnum사의 GV7601을 사용 하였다. 이것은 S6000 디바이스 4개를 Cascade 하여 4채널을 구현한 보드로서 개별적으로 획득된 영상을 각각의 S6000 디바이스가 영상을 압축하고, 디스플레이 화면에 맞게 스케일다운 한 후, 4개의 S6000중 마스터 디바이스가 스케일 다운된 영상과 압축된 영상데이터를 받아 PCI Express Bus로 시스템에 전달하는 구조로 되어 있다. S6000 디바이스는 압축 전용 디바이스이기 때문에 압축된 영상을 복원 하기위해서는 CPU를 사용하는 소프트웨어 디코딩 방식을 사용하였다. 그로 인해서 하드웨어 압축 방식을 사용 하였더라도 소프트웨어 방식으로 디코딩을 해야 하기 때문에 시스템의 사양을 낮추면 디코딩에 따른 CPU의 부하가 커지는 문제가 발생하였다.



[Fig. 11] Stretch S6000 Board Block Diagram

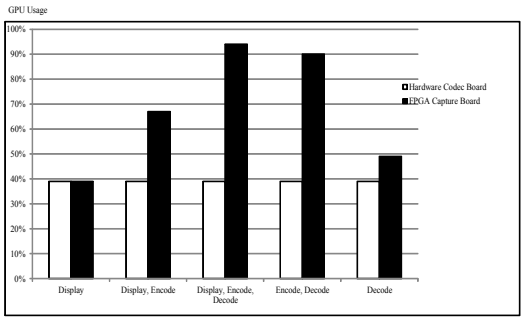
Table 2와 같이 동일한 시스템 사양에서 하드웨어 압축 방식과 FPGA와 시스템내의 GPU를 이용한 방식의 성능 검증 결과 CPU의 사용량은 각 기능별로 Fig. 12과 같은 사용량을 보였다.

획득된 영상을 디스플레이하고 압축 할 때는 두 개의 제품이 동일한 성능을 보였지만, 압축된 영상을 복원 할 때는 하드웨어 압축 방식의 시스템에서 CPU의 사용량이 2배 이상 많아지는 것을 확인 하였다. 영상을 압축 할 때는 하드웨어 압축 전용 디바이스를 사용 하지만 압축된 영상을 복원 할 때는 CPU를 사용하기 때문이다.



[Fig. 12] CPU Usage

Fig. 13은 GPU의 사용량을 나타낸다. 하드웨어 압축 방식은 영상의 압축과 복원, 디스플레이등 모든 기능을 사용해도 39%로 동일한 사용량을 보였다. 이는 GPU가 영상 압축에 대한 기능을 수행하지 않고, 본 논문에서 제시한 시스템과 같이 영상의 디스플레이와 복원하는 기능만 수행하기 때문에 압축을 하지 않더라도 동일한 사용량을 보이는 것이다. 그로 인해 하드웨어 압축 방식은 GPU의 61%는 사용하지 않는다. 이는 GPU의 성능에 따른 시스템 리소스의 사용량이 적을 뿐만 아니라 별도 하드웨어 압축 디바이스를 사용함으로써 GPU의 자원을 낭비하게 되는 것이다. 반면에 FPGA 영상획득방식 시스템은 FPGA와 GPU를 사용함으로써 시스템의 구성도 간단한 구조로 구성 할 수 있었고, 시스템의 자원을 최고 90% 이상 사용을 함으로써 자원의 낭비도 줄일 수 있었다.



[Fig. 13] GPU Usage

3. 결론

본 논문에서는 Full-HD(1920x1080)급 영상을 실시간으로 획득하고, 압축하여 저장 할 수 있는 DVR 시스템

구현에 따른 하나의 방법으로 FPGA를 이용하는 방법을 제시 하였다.

HD-SDI 카메라로부터 받은 HD-SDI 영상 신호를 이퀄라이저(Equalizer)를 통하여 HD-SDI 신호의 짧은 전송 거리에 따른 감쇄를 보상하여 노이즈가 없는 깨끗한 영상을 수신하였고, 별도의 SerDes칩을 사용하지 않고 FPGA내에 구현함으로써 제품의 가격 상승요인을 차단하고, YUV4:2:2 포맷을 YUV4:2:0 포맷으로 변환하여 화질 감쇄를 최소화하는 방법으로 데이터양을 줄여 H.264 코덱에서 사용하기 위한 데이터 포맷으로 변환함으로써 실시간 영상 획득이 가능 하였다.

또한, 기존 하드웨어 압축 방식의 DVR 시스템과 비교하여 FPGA를 이용한 영상 획득으로 Full-HD(1920x1080)영상 사이즈의 획득양이 80프레임에서 120프레임으로 30%의 성능 향상이 검증되었고, Intel Media SDK를 이용한 GPU 를 사용함으로써 CPU에 부담을 주지 않고 Full-HD급 영상 압축이 가능한 것을 실험결과를 통하여 확인 하였다[11].

본 논문에서 제시한 방법으로 샌디브릿지(Sandy-bridge) i7-2600 CPU를 사용 하였지만, 현재 그 후속버전인 아이비브릿지(Ivy-bridge)나 하스웰(Haswell)과 같이 CPU의 처리 속도가 향상되고, 내장된 GPU도 Intel HD Graphics 2000에서 HD Graphics 3000, HD Graphics 4000까지 그 성능을 향상 시키고 있어 향후 FPGA를 이용하여 고해상도의 영상을 획득하고 GPU를 이용한 압축 방식은 성능이 더욱 향상 될 것이라 기대한다. 또한 채널수가 증가할 경우 위에서 제시된 두 가지 구현방법 모두 Decoding에서 GPU를 사용하게 되면 시스템 성능을 한층 높일 수 있을 것이다.

References

[1] SMPTE 274M-2005, Television, "1920 × 1080 Image Sample Structure, Digital Representation and Digital Timing Reference Sequences for Multiple Picture Rates", SMPTE STANDARD, 2005

[2] BT Series Broadcasting service(television) "STUDIO PARAMETERS OF 625 AND 525 LINE PROGRESSIVE SCAN TELEVISION SYSTEMS", ITU-R BT.1358, pp. 1-2, Feb-1998

[3] J.W.Seok, B.H.Kim, J.W.Lee, C.S.Cho, “The Research Trend of the H.264 Technology” Electronics and Telecommunications Trends vol. 21, no. 1, ETRI Journal, pp. 2-10, Feb-2006.

[4] Gennum "GV8501 ActivConnect Multimedia Receiver Preliminary Data sheet 47058-0" pp. 1-8, Feb-2008.

[5] Lattice Semiconductor, “Lattice ECP3 SERDES/ PCS Usage Guide”, pp. 1-17, Aug-2012

[6] Lattice Semiconductor, “PCI Express 1.1 x1, x4 Endpoint IP Core User's Guide”, pp. 10-23, Sep-2010

[7] BT Series Broadcasting service(television) "BT.1120-8, "Digital interfaces for HDTV Studio Signals", pp. 32-38, Jan-2012.

[8] S.J.Sagong, K.H.Park, E.J.Jung, “Writing Windows WDM Device Drivers, pp. 13~51", Acorn publishing Co, 2000-10.

[9] Intel Media SDK "Intel media SDK2013 Developer's Guide - Hardware Accelerated Video On Intel Platforms", INTEL Corporation, pp. 5-7, Dec-2013

[10] Stretch "S6000 Family DataSheet Version1.1", Stretch Inc, pp. 4-17, June-2009.

[11] Wan-Jae Kim, “A Study on the implementation of 4 Channel Full-HD DVR System with FPGA” pp. 21-45, Oct-2013.

김 기 화(Ki-Hwa Kim) **[정회원]**

- 1990년 2월 : 동국대학교 전자계산학과 졸업
- 2002년 2월 : 한국산업기술대학교 전자공학과(석사)졸업
- 2013년 2월 : 한국산업기술대학교 정보통신(박사)수료
- 2000년 6월 ~ 2003년 8월 : 한국산업기술대학교 컴퓨터공학과 겸임교수
- 1999년 4월 : Visual Basic Win32 API Bible 저술(삼양출판사, 1144쪽)
- 1999년 5월 ~ 현재 : (주)시큐인포 대표이사