특집논문-11-16-3-02

독립적 홀로그램 화소 연산 방식의 고성능 디지털 홀로그램 생성기의 하드웨어 구조

이 윤 혁[a], 서 영 호[b], 최 현 준[a], 김 동 욱[a)‡]

A New Architecture of High-Performance Digital Hologram Generator based on Independent Calculation of a Holographic Pixel

Yoon-Huyk Lee[a], Young-Ho Seo[b], Hyun-Jun Choi[a], and Dong-Wook Kim[a)‡]

요 약

본 논문에서는 고속으로 디지털 홀로그램을 생성할 수 있는 하드웨어구조를 제안하였다. 수정된 컴퓨터 생성 홀로그램(computer-generated hologram, CGH) 알고리즘을 이용하고, 기존의 한 화소에 대한 홀로그램 전체 화소를 연산하는 방법이 아니라 객체 전체 화소에서 홀로그램의 한 화소씩 연산하는 방법을 선택하여 홀로그램 한 화소씩 계산하고 바로 출력 하여 메모리 병목 현상을 제거하기 위한 파이프라인 기반의 하드웨어 구조를 제안하였다. CGH 알고리즘을 바탕으로 입력부, 연산부, 및 정규화부로 구성된 디지털 홀로그램 생성기의 구조를 제안하였고, 이를 효율적인 하드웨어로 구현하였다. 객체의 화소만 저장하여 반복 사용하기 때문에 메모리의 사용량을 줄일 수 있었다. 제안한 하드웨어는 세로 방향으로 확장을 하여 동작을 병렬화시킬 수 있다. 제안한 하드웨어는 1K의 광원에 대해 HD급 홀로그램을 초당 약 87장을 생성할 수 있었다.

Abstract

In this paper, we proposed a hardware architecture to generate digital holograms at high speed. It used the modified computer-generated hologram (CGH) algorithm and adapted the pipeline-based hardware to be able to remove memory bottleneck problem. It uses not the method which generates a hologram by accumulating intermittent holograms but the one which independently generates a pixel of a final hologram and uses the appropriate CGH algorithm for the selected method. Based on the CGH algorithm we proposed the architecture of the digital hologram generator which consists of input interface part, calculating part, and normalizing part. The hardware can decrease memory usage because it repeatedly use object light sources which is stored in the internal buffer. It is also operationally parallelized by vertically adding unit cells. It can generate 86 frames of HD digital hologram per 1 second for 1K light sources.

Keyword : digital hologram, computer-generated hologram, VLSI, hardware design, parallel architecture

---

a) 광운대학교 전자재료공학과
Department of Electronic Materials Engineering, Kwangwoon University
b) 광운대학교 교양학부
College of Liberal Arts, Kwangwoon University
‡ 교신저자 : 김동욱(dwkim@kw.ac.kr)
※ 본 연구는 지식경제부, 방송통신위원회 및 한국산업기술평가관리원의 산업원천기술개발사업(정보통신)의 일환으로 수행하였음. [KI002058, 대화형 디지털 홀로그램 통합서비스 시스템의 구현을 위한 신호처리 요소 기술 및 SoC 개발]
・접수일(2011년3월16일),수정일(2011년5월11일),게재확정일(2011년5월11일)

Ⅰ. 서 론

홀로그래피는 1948년 Gabor에 의해 최초로 제안된 이래 3차원 정보를 기록할 수 있다는 특징 때문에 많은 연구자들의 관심을 끌어왔다. 기존의 홀로그래피는 홀로그램 필름에 3차원 정보를 기록하고, 현상된 필름을 사용하여 3차원 물체를 복원하는 방식을 사용함으로써 그 응용이 크게 제한되었다. 이러한 단점을 극복하기 위한 새로운 접근방법으로 1966년 이후 많은 연구자들이 컴퓨터에 의한 홀로그램(computer-generated hologram, CGH)의 제작을 연구해 오고 있다[1]. 이 기술은 물체파(object wave)와 기준파(reference wave)의 간섭에 의해 생성되는 간섭항을 계산함으로써 현실에서는 불가능한 이상적인 특성을 가진 부품을 제작하거나 특성시험 등을 위해 개발되었다[2][3].

CGH를 이용하여 한 프레임에 해당하는 홀로그램을 생성하기 위해서는 많은 연산량과 시간이 소요되기 때문에 고속의 연산방법이 필요하다. 이런 고속 CGH를 위한 여러 알고리즘들이 개발되어 왔다[4~6]. MIT Media lab의 Spatial Imaging Group(지금은 Object-based Media Group)은 고속 CGH를 위한 연구를 가장 먼저 시작한 연구그룹이다[4]. 여기서는 HPO(horizontal-parallax-only) 방식의 CGH 기법을 이용해 디지털 홀로그램을 생성하는 연구를 수행하였다. 이 연구에서는 LUT(Look-up Table)방식과 parallel supercomputer를 이용해 10,000 point의 light source 영상으로부터 1초에 한 장씩 디지털 홀로그램(해상도: 6M)을 생성하였다. 일본 Nihon대학의 Yoshikawa교수는 root연산을 Taylor 전개를 통해 변형한 후 CGH 수식의 근사화를 통해 고속화 알고리즘을 정리하였다[5]. 이 연구는 최근 연구되고 있는 고속 CGH 알고리즘들의 이론적 토대가 된다는 점에서는 상당히 가치가 있지만, 연산속도 측면에서는 큰 의미가 없다. 일본의 Chiba 대학의 연구팀은 Yoshikawa교수의 이론을 변형시켜 x축에서 반복덧셈만을 수행하여 CGH를 계산하는 알고리즘을 제안하고 FPGA를 이용한 하드웨어로 구현하였다[6]. 이와 같이 CGH 연산량이 너무나 방대하기 때문에 실제로 소프트웨어로 CGH를 실시간으로 처리하는 것은 불가능하고 하드웨어로 구현되어야 하며, 지금까지 CGH를 위해 하드웨어를 활용한 많은 연구가 진행되어 왔다[7~14]. 이러한 연구들은 GPU 기반의 소프트웨어 방식[7~11]과 FPGA 기반의 하드웨어 칩 구현 방식[6][12~14]으로 나누어진다. GPU를 이용한 방식은 FPGA 기반의 방식에 비해서 구현이 비교적 쉽고 개발기간이 짧다는 장점이 있다. FPGA를 이용하면 구현과정이 매우 복잡하고 개발기간이 오래 걸린다는 단점이 있다. 또한 한번 구현하면 구조를 변경하거나 성능을 개선하기 어렵다. 그러나 GPU 방식에 비해서 성능은 수십에서 수백배 가량 높은 성능을 갖는다. 특히 GPU에 대한 연구가 최근에 활발히 이루어지고 있다. 싱가폴대[9]는 CGH 수식을 exponent 함수를 이용한 복소형태로 변환한 후에 연산을 분리하는 알고리즘을 제안하였다. 분리된 항을 각각 LUT로 만든 후에 연산을 고속화시켰고, 이를 nVidia의 GPU로 구현하였다. 1,000(1K)개의 object point를 갖는 객체에 대해 1024×768크기의 홀로그램을 0.3초당 한 장씩 생성할 수 있었다. 중국 Zhongshan 대학의 Wang[10]은 3D mesh-model을 기반으로 GPU를 이용해 CGH를 수행한 연구를 발표했다. 또한 일본 Chiba대의 Shimobaba[11]는 AMD의 GPU를 기반으로 하여 이전 연구에서 제안한 알고리즘[6]을 사용하면서 GPU 프로그래밍 기법을 활용하여 고속화하였고, HD크기의 홀로그램을 0.31초당 한 장씩 생성할 수 있었다. [12]에서는 4개의 Xilinx FPGA (XC2VP70)를 사용하는 전용 PCB 보드를 제작하여 Fresnel Transform CGH를 구현하였다. 홀로그램의 x축 해상도만큼의 단위 연산기를 병렬로 배열(1,408개)하는 구조를 가지고 166MHz의 클록 주파수에서 한 프레임의 홀로그램을 0.0679초에 생성할 수 있다. 최근에는 CGH를 연산하기 위한 전용 연산 시스템인 HORN-6 특수 컴퓨터가 제안되기도 하였다[13]. 또한 100% 파이프라인(pipeline) 구조를 기반으로 하는 CGH 프로세서가 제안되었다[14]. Fresnel 변환을 수행하기 위한 CGH Cell의 하드웨어 구조를 제안한 후에 이를 확장하여 CGH Kernel을 구성하였고, 이를 다시 확장하여 CGH 프로세서를 구현하였다. [14]의 하드웨어는 [12]보다 최대 87.32%의 높은 성능을 갖는다. [15]의 논문에서 사용한 하드웨어는 1920×1080 크기의 HD급의 홀로그램을 생성할 수 있다.

[15]의 논문에서 구현한 하드웨어는 순수한 홀로그램을 계산하는 셀 기반이기 때문에 계산 후 출력을 고려했을 때

메모리 병목 현상으로 인한 여러 장의 홀로그램 구현 시 지연시간이 생긴다. 또한 한 홀로그램을 계산 할 경우 모든 객체의 광원에 대하여 반복 덧셈을 해야 하기 때문에 HD급에 맞는 메모리 공간이 필요하다. 본 연구팀은 이전 연구[14][15]에서 제안된 하드웨어에서 여러장의 홀로그램 구현을 위하여 메모리 스케쥴링과 자원을 줄이기 위한 알고리즘을 재구성한 후 파이프라인 기법을 적용하여 고성능의 CGH 프로세서를 제안하였다. 본 연구에서는 이전 연구에서 제안된 것보다 동영상 홀로그램 생성에 더욱 우수한 성능을 가질 수 있게 하기 위하여 메모리 사용에 있어서 최적화할 수 있는 하드웨어 구조를 새롭게 제안하고자 한다.

본 논문은 다음과 같이 구성되어 있다. 2장에서는 CGH의 원리와 하드웨어 구현을 위해 수정된 CGH 알고리즘을 설명한다. 3장에서는 제안한 하드웨어 구조를 설명하고 4장에서는 구현결과를 보이고 기존 연구와 비교한다. 마지막으로 5장에서는 결론을 맺는다.

II. 컴퓨터 생성 홀로그램

1. CGH 알고리즘

홀로그램은 광학계를 이용하여도 취득할 수 있지만 광학계 자체를 수학적으로 모델링한 연산에 의해서 구할 수도 있다. 이러한 수학적인 연산을 통해 얻어진 홀로그램을 컴퓨터 생성 홀로그램(computer-generated hologram, CGH)이라고 한다. 여러 종류의 CGH가 있지만 본 논문에서는 “위상(phase)” 방식을 사용하는데, 이것은 객체에서 CCD로 입사되는 파에서 위상 성분만을 이용하여 홀로그램을 생성하는 것으로, 그 증명[3]은 본 논문에서 다루지 않는다.

파면의 간격이 시간과 공간에 걸쳐 일정하게 유지되는 가간섭성(coherent) 광이 어떤 형태를 가진 물체에 부딪쳐 반사될 때 반사된 광의 파면은 물체의 형태에 비례하여 달라진다. 즉 파면의 위상 변화가 물체의 형상에 따라 변화하게 된다. 그러므로 물체의 형상에 관한 좌표의 정보가 있으면, 형상에 따라 변화하는 파면의 위상변화는 기하광학적인 광선추적에 의해 쉽게 계산된다.

CGH는 식 (1)과 같이 정의되는데 앞서 설명한 것과 같이 홀로그램의 위상으로부터 홀로그램의 강도(I_α)를 얻는 방법이다. 여기서 N은 3차원 객체의 광원수를 뜻한다. k는 참조파의 파수로 $2\pi/\lambda$로 정의되고 λ는 사용된 파의 파장을 나타낸다. x_α와 y_α는 홀로그램내의 위치를 뜻하고 x_j, y_j, 및 z_j는 3차원 객체의 위치를 나타낸다. p는 픽셀의 크기를 나타낸다.

$$I_\alpha = \sum_j^N A_j \cos\left(k\sqrt{(px_\alpha - px_j)^2 + (py_\alpha - py_j)^2 + z_j^2}\right) \quad (1)$$

식 (1)에서 제곱근은 $x_{\alpha j}$, $y_{\alpha j} << z_j$의 조건인 경우에 식 (2)와 같이 Fresnel 근사를 통해서 근사될 수 있다[5]. $x_{\alpha j}$와 $y_{\alpha j}$는 각각 $x_{\alpha j} = x_\alpha - x_j$ 및 $y_{\alpha j} = y_\alpha - y_j$로 정의된다.

$$\sqrt{(px_\alpha - px_j)^2 + (py_\alpha - py_j)^2 + z_j^2} \cong z_j + \frac{p^2}{2z_j}(x_{\alpha j}^2 + y_{\alpha j}^2) \quad (2)$$

식 (2)를 이용하여 식 (1)을 다시 정리하면 식 (3)과 같다.

$$I_\alpha = \sum_j^N A_j \cos\left(k\left(z_j + \frac{p^2}{2z_j}(x_{\alpha j}^2 + y_{\alpha j}^2)\right)\right) \quad (3)$$

2. 수정된 CGH 알고리즘

본 절에서는 앞 절에서 설명한 CGH 알고리즘의 효율을 높이기 위해 제안된 방법을 소개한다[6]. 먼저 식 (3)을 θ_H 이용하여 정의하면 식 (4),(5) 같고, 이를 θ_Z와 θ_{XY}를 이용하여 식 (6)와 식 (7)으로 정의된다[14].

$$I_\alpha = \sum_j^N A_j \cos(\theta_H) \quad (4)$$

$$\theta_H(x_{\alpha j}, y_{\alpha j}, z_{\alpha j}) = 2\pi(\theta_Z + \theta_{XY}) \quad (5)$$

$$\theta_Z(z_j) = \frac{z_j}{\lambda} \quad (6)$$

$$\theta_{XY}(x_{\alpha j}, y_{\alpha j}, z_{\alpha j}) = \frac{p^2}{2\lambda z_j}(x_{\alpha j}^2 + y_{\alpha j}^2) \quad (7)$$

홀로그램 평면의 한 열에서 제일 첫 번째 화소위치로부터 d번째 화소위치의 θ_{XY}를 구하면 식 (8)과 같다. 식 (8)에서 $\frac{p^2}{2\lambda z_j}(x_{\alpha j}^2+y_{\alpha j}^2)$는 초기에 $d=0$일 때 연산되는 항에 해당하고, 는 $\frac{p^2}{2\lambda z_j}(2dy_{\alpha j}+d^2)\,d=0$일 때 위치와의 θ_{XY}와 차이다. 즉 세로방향으로 이동함에 따라서 그 만큼의 값이 차이나는 것이다.

$$\begin{aligned}&\theta_{XY}(x_{\alpha j}, y_{\alpha j}+d, z_{\alpha j})\\&=\frac{p^2}{2\lambda z_j}(x_{\alpha j}^2+y_{\alpha j}^2)+\frac{p^2}{2\lambda z_j}(2dy_{\alpha j}+d^2)\end{aligned}\tag{8}$$

이 수정된 CGH 알고리즘은 하나의 세로줄 단위로 연산이 이루어진다. 먼저, 세로줄의 첫 번째 홀로그램 위치($d=0$)에 대해 연산을 수행하고 그 이후의 위치($d>0$)에 대해서는 세로줄이 끝날 때 까지 이전 위치에서 연산된 결과(Γ_{d-1})에 일부 값($\Gamma_1+(d-1)$)을 보정하여 해당 위치에서의 홀로그램 값(I_α)을 구한다. 세로줄의 첫 번째 위치에서 연산되어야 하는 과정은 식 (9)과 같다.

$$I_\alpha=\sum_j^N A_j\cos\left(2\pi\left(\frac{z_j}{\lambda}+\frac{p^2}{2\lambda z_j}(x_{\alpha j}^2+y_{\alpha j}^2)\right)\right)(d=0)\tag{9}$$

(d=1)의 위치부터 사용되는 연산 식은 식 (10)과 같다. 식 (9)에서 Γ_d는 식 (11)과 같고, Γ_1와 Δ는 식(12) 및 식 (13)과 같다[8][9].

$$I_\alpha=\sum_j^N A_j\cos\left(2\pi\left(\frac{z_j}{\lambda}+\frac{p^2}{2\lambda z_j}(x_{\alpha j}^2+y_{\alpha j}^2)+\Gamma_d\right)\right)(d\geq 1)\tag{10}$$

$$\Gamma_d=\Gamma_{d-1}+\Gamma_1+(d-1)\Delta,\quad(d\geq 1)\tag{11}$$

$$\Gamma_1(y_{\alpha j}, z_j)=\frac{p^2}{2\lambda z_j}(2y_{\alpha j}+1)\tag{12}$$

$$\Delta=\frac{p^2}{\lambda z_j}\tag{13}$$

식 (8)과 (9)를 이용하여 객체의 첫 번째 광원으로 중간 홀로그램을 구하여 저장하고, 두 번째 광원으로 두 번째 중간 홀로그램을 구한 후 첫 번째 중간 홀로그램과 더한다. 객체를 구성하는 광원의 수만큼 이러한 과정을 반복하여 최종적인 홀로그램을 생성한다[14][15].

III. 제안한 하드웨어의 구조

1. 병렬화 수식의 유도

본 절에서는 제안하고자 하는 연산 방식에 대해서 설명한다. 앞서 설명한 CGH 생성 수식을 새롭게 재구성하여 병렬 연산에 적합하게 한다. Γ_d는 식 (14)으로 정의할 수 있고, d 값을 증가시키면서 Γ_d를 구하면 식(15)과 같이 Γ_1과 $\frac{p^2}{\lambda z_j}$에 대한 일반항으로 정리할 수 있다[14][15].

$$\Gamma_d(y_{\alpha j}, z_j)=\frac{p^2}{2\lambda z_j}(2dy_{\alpha j}+d^2)\tag{14}$$

$$\Gamma_d(x_{\alpha j}, z_j)=\frac{p^2}{2\lambda z_j}(2dx_{\alpha j}+d^2)=d\Gamma_1+\frac{d(d-1)}{2}\Delta\tag{15}$$

하드웨어의 효율성을 고려한다면 식 (15)은 식 (16)와 같이 변형하는 것이 유리하다. 두 식이 동일한 것으로 보이지만 하드웨어 구현 측면에서는 차이점을 갖는다. 하드웨어의 측면에서 식 (15)은 3개의 곱셈기와 1개의 덧셈기가 필요하지만 식 (19)는 2개의 곱셈기와 1개의 덧셈기가 필요하다. 즉, 곱셈기의 개수가 1개 감소한다. 1개의 곱셈기 자체의 감소는 중요하지 않게 생각될 수 있다. 그러나 CGH 셀이 홀로그램의 열의 개수(예를 들어 1080개)만큼 사용된다는 것을 고려한다면 중요한 요소로 고려될 수 있다. $(d-1)$은 열의 개수만큼 사용된다면 각 셀마다 일정한 값이 되고 Δ'은 Δ를 LUT(Look-up Table)방식을 사용하기 때문에 LUT를 만들 때 Δ'으로 만들면 자원 사용에 차이는 없다.

$$\Gamma_d=d\left\{\Gamma_1+(d-1)\Delta'\right\}\tag{16}$$

$$\Delta'=\frac{\Delta}{2}\tag{17}$$

식 (17)는 식 (10)과 동일한 결과를 가지는 식이지만 식 (10)과 달리 한 열의 초기 값($\theta_{XY_{d=0}}, I_1$)이 주어진 이후에는 그 열의 임의의 위치에서 홀로그램 계수 생성이 가능하다는 장점을 갖는다. 이러한 장점은 하드웨어 구현 시 다양한 병렬 화를 가능하게 해줄 수 있다. 뿐만 아니라 홀로그램 평면에서 임의의 영역에 대한 홀로그램을 선별적으로 구할 수도 있어서 다양한 활용이 가능하다.

2. 하드웨어 구조

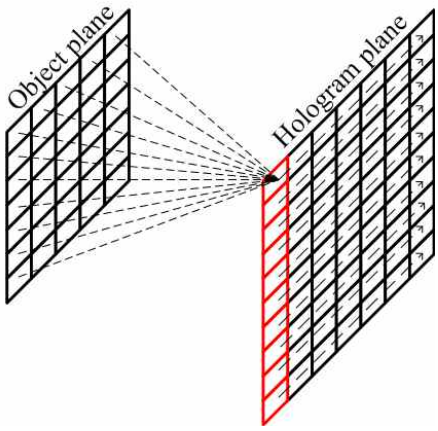
2.1 전체 하드웨어의 구조 및 동작

본 장에서는 새롭게 유도된 CGH 수식을 바탕으로 하여 새로운 하드웨어 구조를 제안한다.

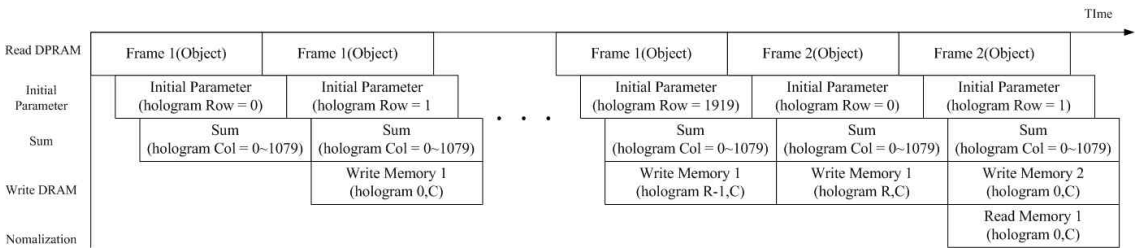
그림 1(a)와 같이 홀로그램 평면의 한 화소를 연산하기 위해 모든 광원에 대하여 연산을 하면 그림 1(b)와 같이 광원의 총 화소 수만큼의 주기를 가지면 한 세로줄에 대한 연산을 할 수 있다. 다음 세로줄을 연산중에 이전 세로줄에 연산되었던 값들은 외부 메모리에 저장할 수 있다. 따라서 홀로그램 평면에 대한 병렬화할 시에 생기는 메모리 병목 현상을 제거 할 수 있다. 또한 메모리 자원의 사용량을 고려하였을 때 객체의 한 광원에 대하여 모든 홀로그램 평면의 화소를 연산하는 방법은 홀로그램 평면의 화소 수만큼 자원을 소모한다. 하지만 한 홀로그램 평면에 대하여 객체의 모든 광원을 가지고 연산을 할 경우 객체의 화소 수만큼만 자원을 사용하면 된다. 보통 홀로그램 평면이 객체평면에 비하여 더 고화질이 요구되므로 저장 자원을 줄일 수 있다.

2.2 세부 하드웨어의 구조 및 동작

제안한 하드웨어의 구조는 그림 2에 나타내었다. 하드웨어의 셀은 입력받은 객체의 광원을 받아서 저장시키는 입력 인터페이스, 실제 CGH연산을 하는 CGH 연산기 그리고 연산된 홀로그램을 디스플레이 할 수 있게 256단계로 규격



(a)



(b)

그림 1. 하드웨어 구현을 위한 연산 방식
Fig. 1. Operational method for hardware development

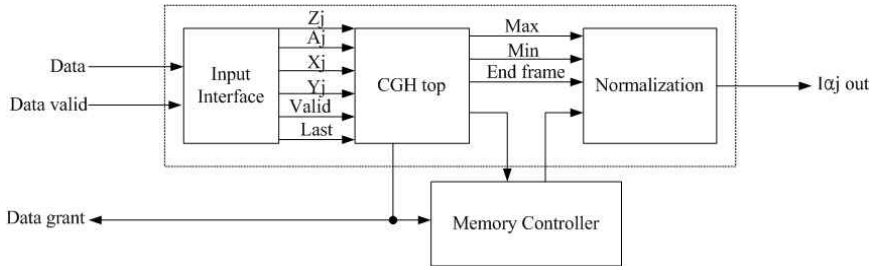


그림 2. 제안한 하드웨어 구조
Fig. 2. The proposed architecture of hardware

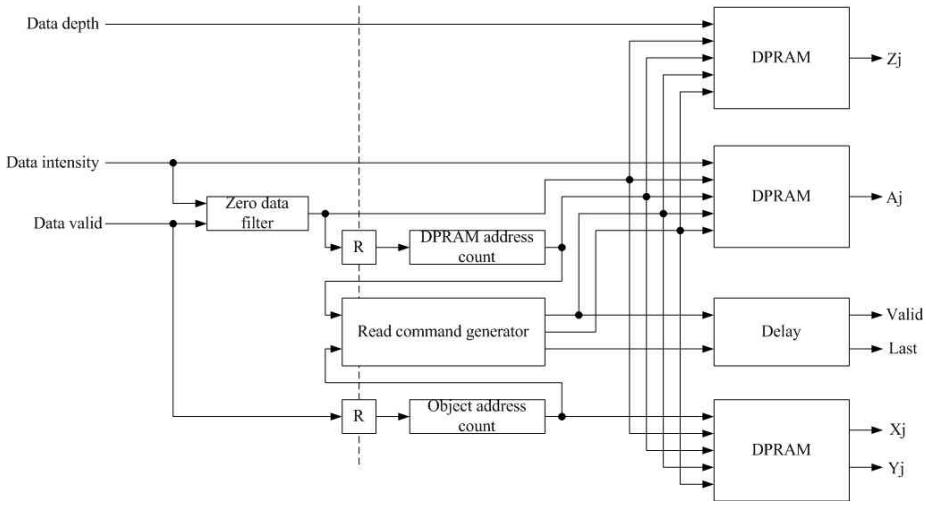


그림 3. 입력 인터페이스 셀의 구조
Fig. 3. The architectur of input interface

화 할 수 있는 규격화기 이렇게 3단계로 구성된다.

그림 3은 그림 2의 입력 인터페이스 블록을 상세하게 나타낸 것이다. 입력 인터페이스 블록의 경우 입력받은 데이터 중에서 밝기 값이 0일 경우 식(1)과 같이 어떠한 코사인 값이 나와도 결과 값이 0이 되어서 누적 덧셈에 영향을 주지 않으므로 저장을 하지 않아도 된다. 하지만 값이 0이라 해도 해당하는 객체의 좌표는 카운트를 해야 함으로 다음과 같은 구조로 구성하였다. 따라서 입력되는 밝기 값이 0이 아닐 경우 DPRAM의 주소를 카운팅 하여서 3개의 DPRAM에 화소에 해당하는 밝기, 깊이정보, 좌표를 저장을 하게 된다. 한 프레임에 관하여 저장이 완료 되면 모든 홀로그램 평면에 대하여 연산이 완료 될 때까지 읽기 명령을 내리게 된다.

하드웨어를 구현하기 위해 [15]에서 CGH를 새롭게 정의한 식을 이용한다. 이 식들을 식 (18)부터 식 (20)까지 나타냈다. 이전 연구 [15]에서 식 (18)은 홀로그램 평면의 세로줄을 연산할 시에 초기 항으로 쓰이는 값을 연산하는 부분과 업데이트 항을 구분한다. 초기 항에 해당하는 부분에 Δ를 위 식(17)을 이용하여 Δ'로 변화 하고 식 (19)와 같이 공통항(i_{common})으로 정의 하고 업데이트항(i_{update})은 (20)로 정의한다[14][15].

$$I_\alpha = \sum_j^N A_j \cos(2\pi(\theta_Z + \theta_{XY,d=0} + \Gamma_d)) = \sum_j^N A_j \cos(2\pi(\theta_Z + \theta_{XY,d=0} + \frac{d}{2}(\Gamma_1 + (d-1)\Delta))) \quad (18)$$

$$i_{common}(x_{\alpha j}, y_{\alpha j}, z_j) = \{\theta_Z + \theta_{XY,d=0}, \Gamma_1, \Delta'\} \quad (19)$$

$$i_{update}(i_{common}(), d) = \{\theta_H = \theta_Z + \theta_{XY,d=0} + \Gamma_d\} \quad (20)$$

식 (19)으로 부터 객체 내의 하나의 광원에 의한 홀로그램 내의 한 세로줄은 식 (20)로 정의된다. 식 (19)의 공통항은 객체내의 하나의 광원에 대하여 홀로그램의 한 세로줄에 처음 한 화소에 대해서만 연산되고 그 이후에는 다음 광원에 대한 처음 한 화소만 연산된다. 식 (20)의 업데이트 항은 공통항에서 연산된 값을 이용하여 나머지 세로줄에 대한 화소를 연산한다. 따라서 식 (19)은 식 (20)와 다른 동작 특성을 갖는다.

그림 4는 CGH를 연산하는 블록은 식(19)와 식(20)의 단계로 구성된다. 각 단계는 공통항 연산기와 업데이트 연산기로 명명하였다. 그림 5처럼 동작을 순서대로 수행할 수 있는 구조를 구성 하였다. 공통항 연산기는 홀로그램의 한 세로줄의 가장 처음 위치($d=0$)에 해당하는 θ_{XY}와 다음 위치($d=1$)에서의 보정 값을 연산한다. 업데이트 연산기는 공통항에서 연산된 데이터를 가지고 ($d>0$)인 세로줄에 대하여 연산한다.

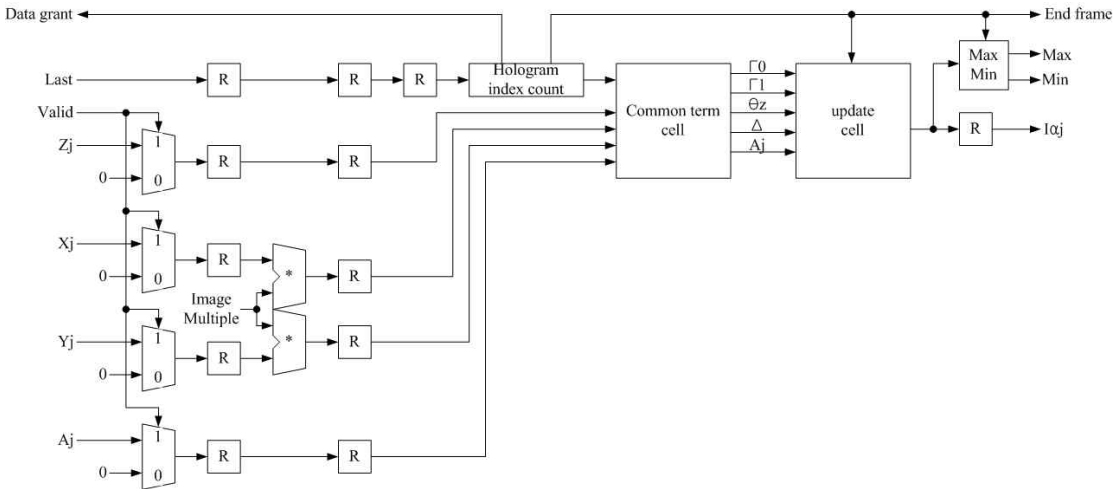


그림 4. CGH연산기의 구조
Fig. 4. The architecture of CGH calculator

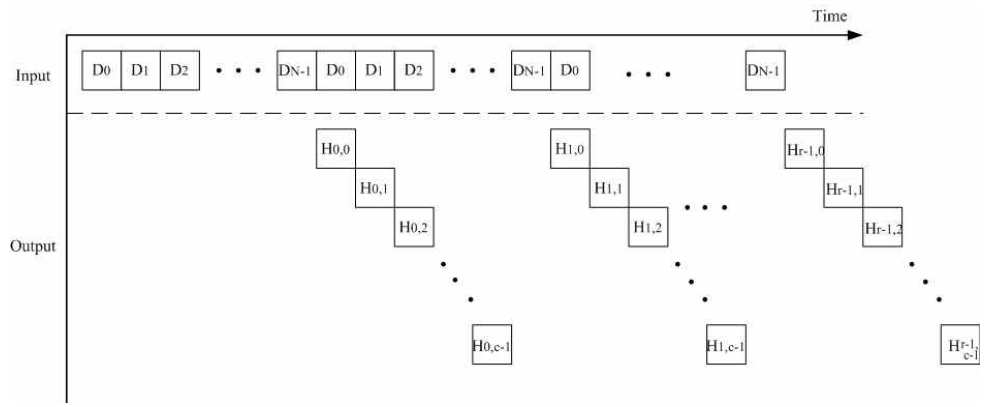


그림 5. CGH연산기의 타이밍도
Fig. 5. The timing diagram of CGH calculator

하드웨어 구현을 고려한다면 식 (19)에서 θ_Z와 Δ'를 LUT로 만드는 것이 바람직하다. 두 개의 항 모두 z_j를 변수로 갖는 연산이다. z_j는 거리를 256단계로 나누는 깊이 정보이다. 따라서 θ_Z와 Δ'은 256개의 주소를 가지는 LUT1으로 통합하고 하드웨어에 내장시킨다. θ_{XY}에서 $y_{\alpha j}$의 y_α처음 좌표 값이므로 $-y_j$같게 되는데 제곱 항이 필요하므로 $y_{\alpha j}^2$은 y_j^2와 같다. 또한 Γ_1항의 경우 홀로그램의 두 번째 좌표 값이므로 $y_{\alpha j}$는 $(1-y_j)$와 같다. 그림 6은 그림 7과 같은 동작을 수행하는 공통항 연산기를 구성하였다. Δ'의 경우 업데이트 항에서 우선적으로 연산해야 하므로 다른 파라미터보다 하나의 스텝 먼저 출력이 되므로 레지스터를 줄일 수 있다. 또한 식 (20)에서도 하드웨어 구현을 고려한다면 코사인 함수 연산은 간단한 룩업테이블로 처리할 수 있다. 따라서 누적되는 항은 LUT2의 결과를 A_j로 곱하는 것으로 매우 단순한 과정이다. 그림 8은 그림 9와같이 동작을 수행하는 업데이트 연산기를 구성하였다. 업데이트 연산기는 HD급(1920×1080)의 한 세로줄의 화소 수

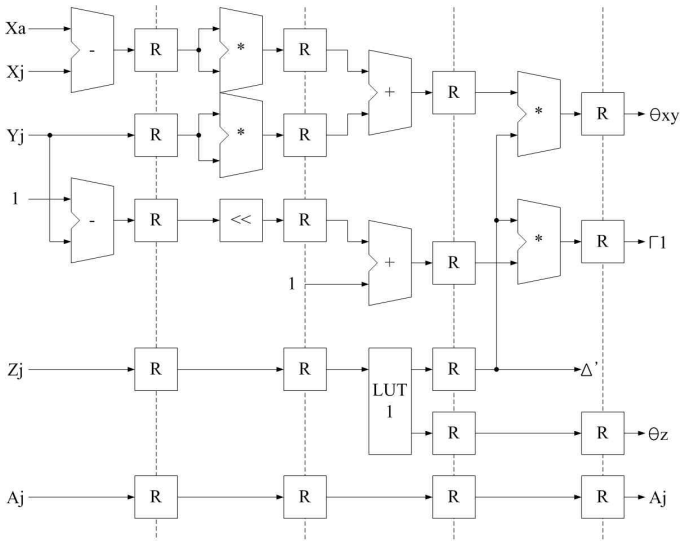


그림 6. 공통항 연산기의 세부 구조
Fig. 6. Detail architecture of common term calculator.

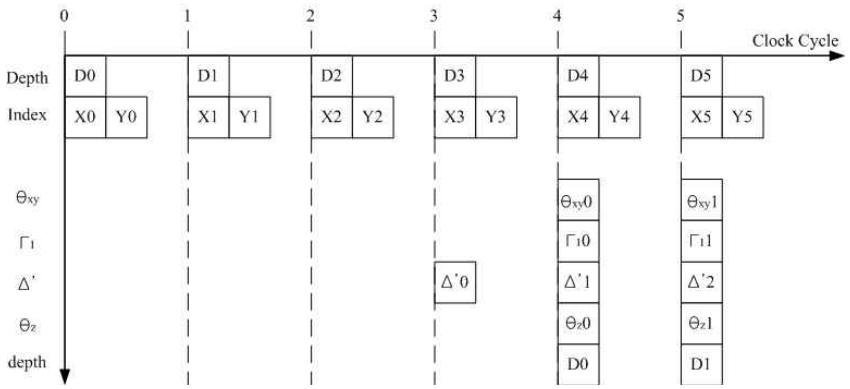


그림 7. 초기 파라미터 연산기의 타이밍도
Fig. 7. Timing diagram of initial parameter calculator

의 배수만큼 구성하므로 d는 각 업데이트 셀마다 일정한 값을 가지게 되므로 카운터 하나가 감소한다. 카운터 하나의 감소는 중요하지 않을 수 있다고 생각할지도 모르지만 한 세로줄의 화소 수(1080개)만큼 감소한다면 중요한 요소로 고려될 수 있다.

CGH 셀의 내부 단위 연산시간을 바탕으로 그림 10과 같이 CGH 셀을 파이프라인화하였다. 총 10 단계의 파이프라인 단계를 가지므로 10 클록의 대기지연시간 이후에 객체의 유효한 화소 수만큼의 주기당 하나의 최종 결과를 출력할 수 있다. 그림 10에서 R이 파이프라인 레지스터에 해

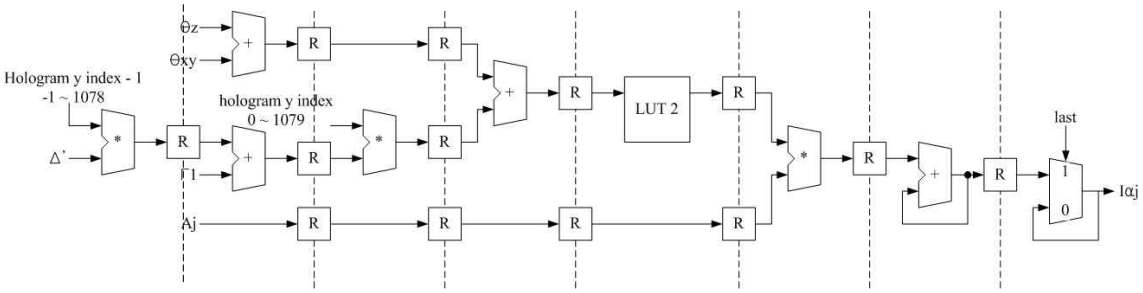


그림 8. 업데이트 연산기의 세부구조
Fig. 8. Detail architecture of initial parameter calculator

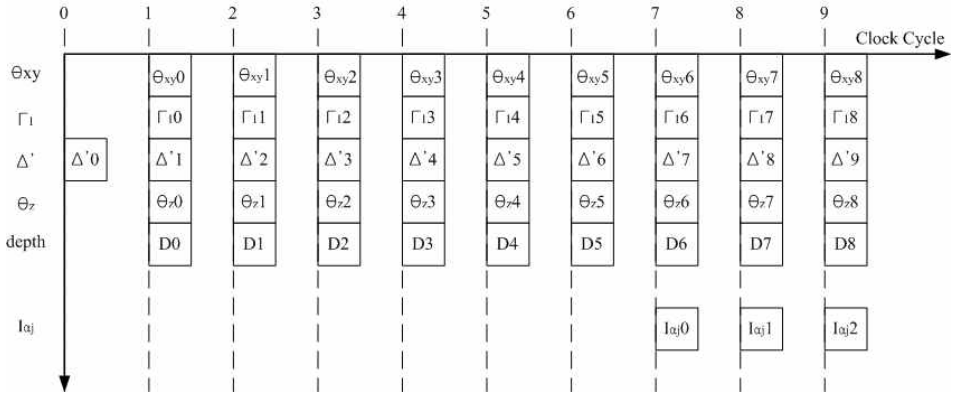


그림 9. 업데이트 연산기의 타이밍도
Fig. 9. Timing diagram of initial parameter calculator

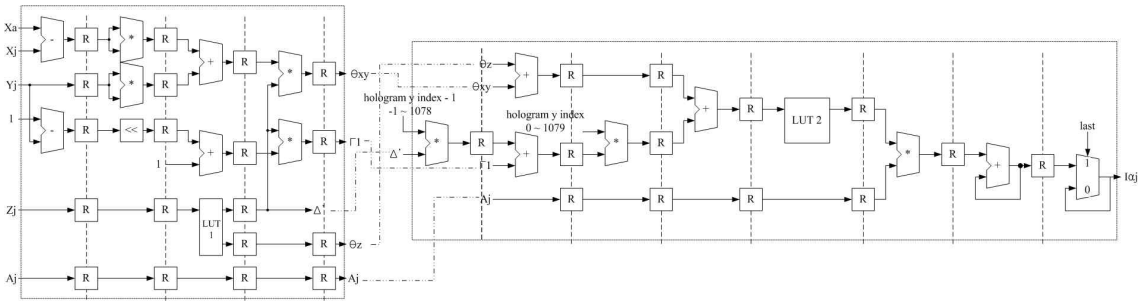


그림 10. 파이프라이닝 초기 파라미터-업데이트 연산기의 구조
Fig. 10. The pipelined architecture of the initial parameter-update calculator

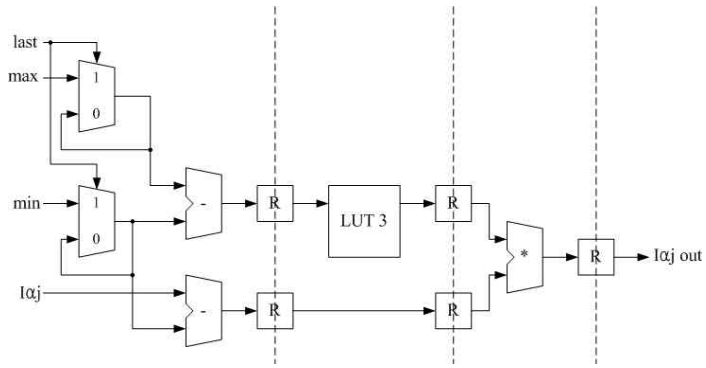


그림 11. 정상화기의 세부구조
Fig 11. Detailed architecture of the nomalization

당한다.

앞서 설명한 것과 같이 식 (15)의 특성은 한 화소에 대한 공통항이 연산된 이후에 하나의 세로줄에 모든 화소들을 병렬적으로 연산을 할 수 있다는 것이다.

CGH연산을 하게 되면 누적 덧셈으로 인하여 유효한 광원의 개수에 따라 홀로그램 한 화소에 대한 값은 증가하게 된다. 하지만 디스플레이 하기 위해서는 한 장의 홀로그램에 모든 값이 256개의 단계로 양자화 되어야 한다. 외부 메모리에 저장되어 있던 이전 프레임의 홀로그램 데이터들과 이전 프레임에서 미리 연산되었던 최대 최소값을 이용하여 현재 프레임이 연산되고 있을 때 규격화를 하는 구조를 구성하였다. 규격화하기 위해서는 식(21)처럼 최대값과 최소값의 차이를 구해서 홀로그램의 값의 크기를 구할 수 있다. 식(22)를 통해서 총 크기에서 외부 메모리에서 읽어오는 홀로그램 값이 전체에서 어느 부분에 해당하는지 연산한다. 식(23)으로 큰 홀로그램의 크기를 256단계로 나누었을 때 그에 해당하는 값을 구할 수 있다. 하지만 총 크기의 범위는 매우 크기 때문에 LUT3에 미리 저장되는 값은 많아지게 되어 비효율적이다. 그래서 홀로그램의 크기를 단계별로 나누어서 LUT3를 구성하였다.

$$magnitude = \max - (-\min) \tag{21}$$

$$order = I_{\alpha j} - (-\min) \tag{22}$$

$$Pixel\,Value = order \times \frac{255}{magnitude} \tag{23}$$

Ⅳ. 구현된 하드웨어

제안한 하드웨어는 Verilog HDL을 이용하고 Altera사의 FPGA 환경을 이용하였다. VHDL의 설계는 Quartus II 10.0을 이용하였고, 시뮬레이션은 Modelsim 6.5e를 사용하였다. 이전 연구[15]와 새롭게 제안한 하드웨어의 메모리 사용 사용량과 30장과 60장의 홀로그램 생성시간을 표1로 비교 하였다. 제안한 하드웨어는 기존의 하드웨어 보다 더 작은 메모리 자원을 사용하면서도 병렬 처리 후 연산된 데이터 처리에 있어서 병목 현상을 제거했기 때문에 여러 장의 프레임을 병렬 적으로 연산이 가능한 장점을 갖는다. N장의 프레임의 홀로그램을 계산하는데 걸리는 시간은 식(24)로 정의할 수 있다. 제안한 하드웨어는 홀로그램의 수직 해상도만큼 CGH 셀 수를 갖고 데이터 입력부 부터 규격화까지 100% 파이프라인 구조로 되어 있다. 따라서 N개의 프레임을 연산할 시에 한 장의 객체 프레임을 입력 받는 시간과 한 장의 홀로그램 프레임을 생성하는 시간이 N번 만큼 반복되고 규격화 하는 시간이 지나면 N장의 홀로그램 프레임을 생성된다.

$$\begin{aligned} &Nframe\,CGH \\ &= ((Input\ object \times 1\ CGH) \times N + \ Normalization) \\ &\quad \times Clock\ Period \end{aligned} \tag{24}$$

식 (24)에 따라서 10K개의 광원을 갖는 객체에 대해서 1920×1080(HD)크기의 홀로그램 30장을 생성하는 데에는

3.47s가 소요된다. 세로 방향으로 3줄씩 홀로그램을 병렬로 연산하는 구조를 적용한다면 1.16s가 소요되고, 5줄에 대해서는 0.7s가 소요된다. 또한 1K개의 광원을 갖는 객체에 대해서 같은 크기의 홀로그램을 30장 생성하는데 0.35s가 소요된다. 즉, 초당 약 87장의 홀로그램을 생성할 수 있다. 표 1에는 구현결과를 정리하였고, 이전 연구들과 비교하였다. 표 1에서 [14]보다 [15]가 더 좋은 결과를 나타내므로 [14]와 제안한 방식의 성능을 비교할 수 있다. 먼저 [15]에 비해서 제안한 하드웨어는 메모리에 병목 현상을 제거하였기 때문에 여러 장의 홀로그램을 생성할 시에 더 적은 메모리 사용량에도 불구하고 고속으로 홀로그램을 생성할 수 있다. 제안한 하드웨어가 더 적은 메모리 사용량과 CGH연산 셀 임에도 여러 장의 홀로그램을 더 빠른 시간에 생성할 수 있다.

표 1. 구현 결과의 비교
Table 1. Comparison of implementing results

Item	FPGA					
	[14]		[15]		Proposed	
Total Frame	30	60	30	60	30	60
Hologram Size	1,408x1,050		1,920x1,080		1,920×1,080	
Frequency (MHz)	166MHz		166MHz		166MHz	
$\frac{Time(s)}{CGH\ Frame}$	2.3	4.8	1.02	2.32	0.76	1.38
Memory Resource	1,478,400	1,478,400	2,073,600	2,073,600	11,080	11,080
Parallel Unit Cells	1,408	1,408	5760	5760	5400	5400

V. 결 론

본 논문에서는 기존의 CGH수식을 이용하여 병렬화된 고성능의 CGH 생성 시 발생하는 다량의 데이터를 효율적으로 처리하기 위해 병목현상을 제거하는 하드웨어 구조를 제안하였다. 하나의 세로줄에 해당하는 공통항을 연산한 후 이 값으로 임의의 세로줄에 대한 홀로그램 화소 값을 구하며 공통항과 세로줄에 해당하는 모든 업데이트 항은 객체의 모든 광원에 대하여 계산을 한다. 이 구조는 최종 홀로그램이 한 세로줄에 연산이 끝날 때 마다 생성되기 때문에 이전 연구[15]에서 최종 홀로그램이 한 번에 생성되는 구조에 비하여 후처리(규격화)를 하는데 있어서 병목 현상을 줄여서 동영상 홀로그램 생성에 효율적이다. 또한 이전 연구[15]에서는 누적 덧셈을 위하여 홀로그램 전체의 해당하는 메모리를 사용하는데 비하여 제안한 하드웨어 구조는 객체의 광원 수와 홀로그램 세로줄에 해당하는 화소 수만큼만 메모리를 사용하기 때문에 비용절감에도 효율적이다. 본 논문은 홀로그램 생성 시 사용되는 메모리, 셀의 수 등 실질적인 비용절감에 대하여 초점을 맞추었다. 기존 연구와 비교할 때 더욱 작은 하드웨어 자원을 사용하면서도 더 높은 성능을 보였고, 세로 방향의 병렬화가 가능하여 성능을 선형적으로 증가시킬 수 있는 장점도 갖는다. 본 연구는 비용 절감으로 인하여 범용적으로 CGH 하드웨어에 사용될 것으로 예상된다.

참 고 문 헌

[1] T. Motoki, H. Isono, and I. Yuyama, “Present Status of Three-Dimensional Television Research,” Proc. IEEE 83(7): 1009-1021(July 1995).

[2] J. K. Chung and M. H. Tsai, Three-Dimensional Holographic Imaging, John Wiley & Sons, Inc., 2002.

[3] P. Hariharan, Basics of Holography, Cambridge University Press, May 2002.

[4] Mark Lucente, "Interfactive Computation of Holograms Using a Look-up Table", Journal of Electronic Imaging, vol. 2, #1, pp. 28-34, Jan. 1993.

[5] H. Yoshikawa, S. Iwase, and T. Oneda, "Fast Computation of Fresnel Holograms employing Differences", Proceeding of SPIE, vol. 3956, 2000.

[6] T. Shimobaba, T. Ito, "An efficient computational method suitable for hardware of computer-generated hologram with phase computation by addition", Computer Physics Communications, vol. 138, pp. 44-52, 2001.

[7] N. Masuda, T. Ito, T. Tanaka, A. Shiraki, and T. Sugie, “Computer generated holography using a graphics processing unit,” Optics Express, Vol. 14, No. 2, January 2006.

[8] L. Ahrenberg, P. Benzie, M. Magnor, and J. Watson, “Computer generated holography using parallel commodity graphics hardware,” Optics Express, Vol. 14, No. 17, August 2006.

[9] Y. Pan, X. Xu, S. Solanki, X. Liang, R. Bin, A. Tanjung, C. Tan,

and T.-C. Chong, "Fast CGH computation using S-LUT on GPU", Optics Express, vol. 17, No. 21, pp. 18543-18555, Oct. 2009.
[10] Y.-Z. Liu, J.-W. Dong, Y.-Y. Pu, B.-C. Chen, H.-X. He, and H.-Z. Wang, "High-speed full analytical holographic computations for true-life scenes", Optics Express, vol. 18, no. 4, pp. 3345-3351, Feb. 2010.
[11] T. Shimobaba, T. Ito, N, Masuda, Y, Ichihashi, and N. Takada, "Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL", Optics Express, vol. 18, no. 10, pp. 9955-9960, May. 2010.
[12] T. Ito, N. Masuda, K. Yoshimura, A. Shiraki, T. Shimobaba, and T. Sugie, “Special-Purpose computer HORN-5 for a real-time electroholography,” Optics Express, Vol. 13, No. 6, March 2005.
[13] Y. Ichihashi, H. Nakayama, T. Ito, N, Masuda, T. Shimobaba, A, Shiraki, and T. Sugie, "HORN-6 special-purpose clustered computing system for electroholography", Optics Express, vol. 17, no. 16, pp. 13895-13903, Aug, 2009
[14] Y.-H. Seo, H.-J. Choi, J.-S. Yoo, and D.-W. Kim, "An architecture of a high-speed digital hologram generator based on FPGA", Journal of Systems Architecture, Vol. 56. pp. 27-37, Dec. 2009.
[15] Y.-H. Seo, H.-J. Choi, J.-S. Yoo, and D.-W. Kim, "A New Parallelizing Algorithm and Cell-based Hardware Architecture for High-speed Generation of Digital Hologram", Journal of Systems Architecture, Vol. 16. pp. 54-63, Jan. 2011.

저 자 소 개

이 윤 혁

- 2006년 3월 ~ 현재 : 광운대학교 전자재료공학과 학사과정
- 주관심분야 : 디지털 홀로그램, SoC 설계

서 영 호

- 1999년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2001년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2004년 8월 : 광운대학교 일반대학원 졸업(공학박사)
- 2003년 6월 ~ 2004년 6월 : 한국전기연구원 연구원
- 2005년 9월 ~ 2008년 2월 : 한성대학교 조교수
- 2008년 3월 ~ 현재 : 광운대학교 교양학부 부교수
- 주관심분야 : 2D/3D 영상 및 비디오 처리, 디지털 홀로그램, SoC 설계, 워터마킹/암호화

최 현 준

- 2003년 2월 : 광운대학교 전자재료공학과 졸업(공학사)
- 2005년 2월 : 광운대학교 일반대학원 졸업(공학석사)
- 2009년 2월 : 광운대학교 일반대학원 졸업(공학박사)
- 2009년 3월~2010년 2월 : 광운대학교 연구교수
- 2010년 3월~현재 : 안양대학교 정보통신공학과 조교수
- 주관심분야 : 영상압축, 워터마킹, 암호학, FPGA/ASIC 설계, Design Methodology

저 자 소 개

김 동 욱

- 1983년 2월 : 한양대학교 전자공학과(공학사)
- 1985년 2월 : 한양대학교 공학석사
- 1991년 9월 : Georgia공과대학 전기공학과 (공학박사)
- 1992년 3월 ~ 현재 : 광운대학교 전자재료공학과 정교수 신기술 연구소 연구원
- 2000년 3월 ~ 2001년 12월 : 인티스닷컴(주) 연구원
- 2009년 3월 ~ 현재 : 광운대학교 실감미디어연구소 연구소장
- 2006년 3월 ~ 현재 : (사)실감미디어산업협회 이사
- 주관심분야 : 3D 영상처리, 디지털 홀로그램, 디지털 VLSI Testability, VLSI CAD, DSP 설계, Wireless Communication