# FPGA Platform for Realtime 3D Reconstruction of Digital Holograms

[1]Chien-Ting Chen, [1]Tzu-Hsin Chuang, [1]Jheng-Chi Lin, [1*]Wen-Jyi Hwang, and [2]Chau-Jern Cheng

[1] *Department of Computer Science and Information Engineering*
[2] *Institute of Electro-Optical Science and Technology, National Taiwan Normal University, Taipei, 117, Taiwan*
**Author to whom correspondence should be addressed:*
whwang@csie.ntnu.edu.tw

*Abstract*—**This paper present a novel FPGA-based hardware platform for realtime 3D reconstruction of digital holograms. The platform can be viewed as a hardware implementation of Fresnel transform for diffraction computation. The circuit employs a novel 2D FFT processor operating in fully pipelined fashion for accelerating the computation. Experimental results reveal that the proposed architecture has the advantages of high throughput, high accuracy and low power consumption for the 3D rendering and display.**

***Keywords*—Holography; 3D Display; FPGA**

## I. INTRODUCTION

Digital holography (DH) is an important 3D rendering technique. An advantage of DH is that the holograms can be stored and transmitted digitally, which allows 3D rendering in remote sites. Therefore, DH is gaining importance in various fields such as metrology, biology, industrial inspection, and consumer electronics [2], [3], [4], [5]. With widespread use of wireless network for data delivery and mobile/embedded devices for display, remote 3D reconstruction of holograms on portable end devices may further extend popularity of the DH-based applications.

A challenging issue for remote 3D reconstruction of holograms is the high computational complexities. Different techniques can be used for for diffraction computation [6] on holograms, including Fresnel transform method, convolution method and angular spectrum method. These methods have a common drawback that they are computationally intensive. Although the fast Fourier transform (FFT) can be used to accelerate the computation, realtime 3D rendering may still be difficult for the end devices with limited computation capacities.

One way to accelerate the diffraction computation is to enhance the computation capacities in the end devices by the employment of the general purpose graphic computation units (GPUs). The GPU-based implementations for Fresnel transform are proposed in [7], [8]. The convolution approach for diffraction computation by GPU is also implemented by [**?**], [9], [10]. These implementations exploit the parallel many-core capability of the GPUs to offer a significant speedup over general purpose multicore CPUs. Although the GPU-based implementations are able to enhance the throughput for the calculation of Fresnel transform, their power consumption may be high.

An alternative to the GPUs is to implement diffraction computation by field programmable gate arrays (FPGAs) [11]. As compared with the GPUs, the FPGAs consume less power. The resulting design can then be used as a co-processor or an accelerator to the CPU in the low power embedded devices for 3D realtime rendering. A number of FPGA-based implementtations have been proposed [12], [13]. These architectures, termed FFT-HORN systems, are designed based on the convolution approach, involving the cascaded operations of Fourier transforms, frequency domain multiplications, and inverse Fourier transform. By the employment of hardware FFT cores, a substantial acceleration can be observed. However, because multiple FFT operations are still required, the computational load may still be high, limiting the throughput of the implementations.

This paper aims to implement a hardware architecture for fast 3D DH reconstruction. The circuit is based on Fresnel transform in its basic form, which contains only a single FFT operation. Although calculations before and after the FFT operation are required, many of these alculations can be simplified by table lookup processes to reduce the computational load. The circuit contains three units: the pre-transform unit, the FFT unit, and the posttransform unit. Each unit is fully pipelined for expediting the computation. Experimental results reveal that the proposed circuit attains higher throughput as compared with existing GPU-based and FPGA-based implementations. The proposed architecture therefore is an effective alternative for DH-based 3D rendering applications where both realtime calculation and low power consumption are important concerns.

## II. THE PROPOSED ARCHITECTURE

### A. Fresnel Transform

The proposed architecture is able to perform diffraction computation for a phase-shifting DH system, where an optical

setup with lasers [14] are used to record light interference

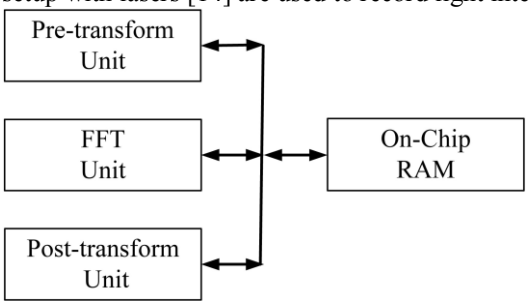



Fig. 1. The proposed architecture for Fresnel transform.

between the object and reference waves. The resulting hologram, denoted by $\eta$, can be captured by CCDs and stored in digital computer. Given the hologram $\eta$, an object image $B$ in a plane parallel to the hologram plane at distance $z$ can be reconstructed by Fresnel transform as follows:

$$B(\mathrm{r},\mathrm{s}) = \frac{-j}{\lambda z} e^{j\frac{2\pi}{\lambda}z} e^{j\frac{\pi}{\lambda z}(r^2+s^2)} \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \eta(p,q) e^{j\frac{\pi}{\lambda z}(p^2+q^2)} e^{-j\frac{2\pi}{\lambda z}(pr+qs)}\, dpdq, \tag{1}$$

where λ is the wavelength of light source, and $(p,q)$ and $(r,s)$are the coordinates on the hologram and image planes, respectively.

Since the hologram $\eta$ is discretized in a CCD, the discrete representations of Fresnel transform is necessary for DH. Suppose the digital recording/sampling operations produce $N \times N$ samples for $\eta$ with sampling interval $\Delta f$ in both the $x$ and $y$ directions. Direct discretization of the Fresnel integral gives the following:

$$B_{u,v} = \frac{-j}{\lambda z} e^{j\frac{2\pi}{\lambda}z} e^{j\frac{\pi}{\lambda z}\Delta_g^2(u^2+v^2)} \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \left[\eta_{x,y} e^{j\frac{\pi}{\lambda z}\Delta_f^2(x^2+y^2)}\right] e^{-j2\pi(\frac{xu}{N}+\frac{yv}{N})}, \tag{2}$$

where$\{B_{u,v}, 0 \le u,v \le N-1\}$ is the object image in digital form,

$$\eta_{x,y} = \eta(x\Delta f, y\Delta f),$$

is the $(x,y)$-th sample of the discretized hologram $\eta$, $0 \le x,y \le N-1$, and

$$\Delta_g = \frac{\lambda z}{N\Delta_f},$$

is the inverse of $\Delta_f$ scaled by $\frac{\lambda z}{N}$.

*B. Architecture Overview*

The proposed architecture aims to compute eq.(2) by FPGA. As shown in Figure 1,

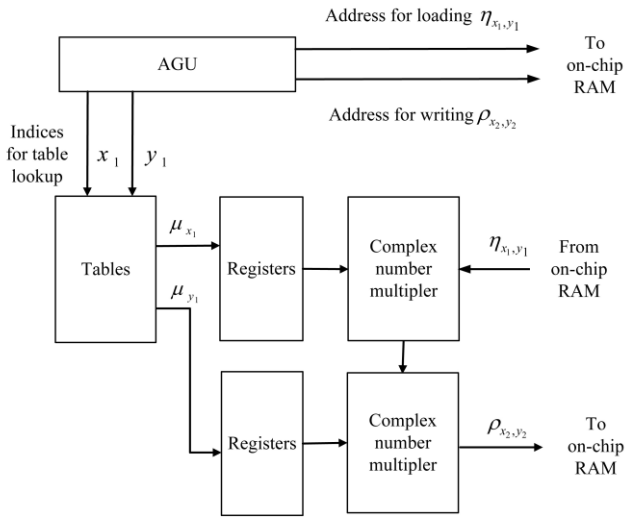



Fig. 2. The architecture of pre-transform unit.

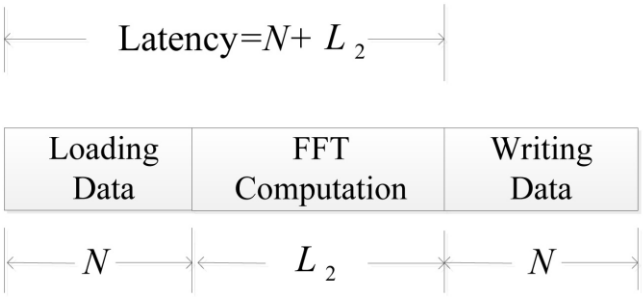



Fig. 3. The operations of the 1D FFT module.

there are three units in the proposed architecture: pre-transform unit, FFT unit, and post-transform unit. The circuit also includes on-chip RAM for reducing memory access time.

The goal of pre-transform unit is to compute

$$\rho_{x,y} = \eta_{x,y} \times \mu_x \times \mu_y, \tag{3}$$

where

$$\mu_x = e^{j\frac{\pi}{\lambda z}\Delta_f^2 x^2}, \quad \mu_y = e^{j\frac{\pi}{\lambda z}\Delta_f^2 y^2}. \tag{4}$$

The FFT unit then takes Fourier transform on $\rho_{x,y}$. The result produced by FFT unit, termed $\tau_{u,v}$, is given by

$$\tau_{u,v} = \sum_{x=0}^{N-1}\sum_{y=0}^{N-1} \rho_{x,y} e^{-j2\pi(\frac{xu}{N}+\frac{yv}{N})}. \tag{5}$$

Define

$$\alpha = \frac{-j}{\lambda z} e^{j\frac{2\pi}{\lambda}z}, \quad \omega_u = e^{j\frac{\pi}{\lambda z}\Delta_g^2 e^2}, \quad \omega_v = e^{j\frac{\pi}{\lambda z}\Delta_g^2 v^2}. \tag{6}$$

By substituting eqs.(5)(6) into eq.(2), it follows that

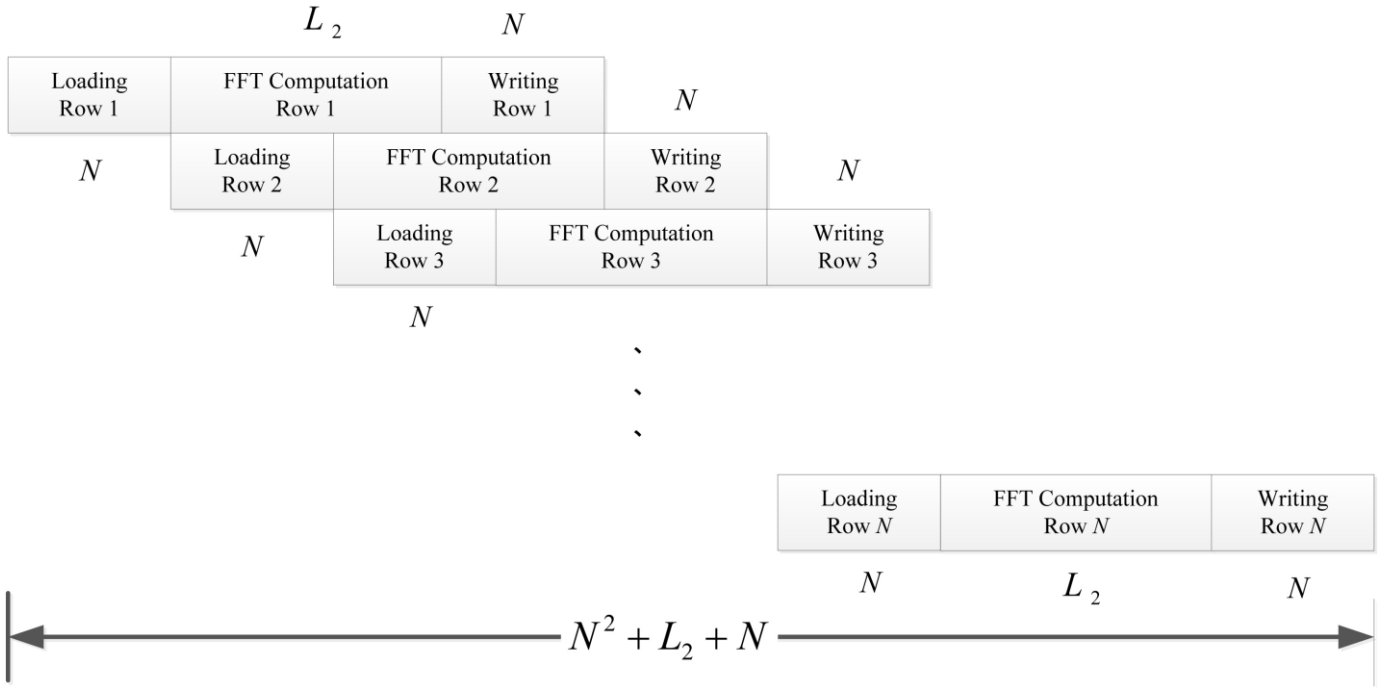



Fig. 4. The row operations of 2D FFT using the 1D FFT module.

$$B_{u,v} = \alpha \cdot \omega_u \cdot \omega_v \cdot \tau_{u,v} \quad (7)$$

Therefore, when $\tau_{u,v}$ is available, the post transform unit computes $\alpha \cdot \omega_u \cdot \omega_v \cdot \tau_{u,v}$ to find $B_{u,v}$. In addition, $\phi_{u,v}$ the phase of $B_{u,v}$, is also computed in the unit for 3D reconstruction.

### C. Pre-transform Unit

The operations of pre-transform unit is based on eq.(3). Therefore, the unit involves the computation of $\mu_x, \mu_y$ and multiplications. To accelerate the computation, the values of $\mu_x$ and $\mu_y$ can be pre-computed, and stored in tables. Because $0 \leq x, y \leq N-1$, $\mu_x$ and $\mu_y$ only take $N$ different values when $\lambda$, $z$, and $\Delta_f$ are known. Therefore, each table for the computation of $\mu_x$and $\mu_y$ contains $N$ entries.

Figure 2 shows the architecture of the pre-transform unit, which contains an address generation unit (AGU), tables, complex number multipliers, and registers. The AGU is responsible for the generation of indices and addresses. The indices are the values of $x$ and $y$, which are used as the inputs to the tables for loading $\mu_x$ and $\mu_y$values. The addresses are sent to the on-chip RAM for loading the hologram $\eta_{x,y}$. The multipliers in the circuit are then used to compute $\rho_{x,y}$, which is then sent back to on-chip RAM for subsequent FFT operations.

Because the multipliers in the architecture are for complex numbers with floating point format, it may be difficult for the multiplications to be completed in a single clock cycle. In our design, all the multipliers perform multiple clock cycles multiplications. To enhance the throughput, they are all fully pipelined. Let $L_1$ be the latency of the pre-transform circuit, defined as the number of clocks required to compute output $\rho_{x,y}$ from input $\eta_{x,y}$. Therefore, as shown in Figure 2, suppose $\eta_{x_1,y_1}$ and $\rho_{x_2,y_2}$ are the input and output to the pre-transform unit, respectively. For the raster scanning order, it then follows that $x_1 \geq x_2$, and $(x_1 N + y_1) - (x_2 N + y_2) = L_1$. The total computation time for the entire array $\{\eta_{x,y}, 0 \leq x, y \leq N-1\}$ is then $N^2 + L_1$.

### D. FFT Unit

The goal of FFT unit is to compute $\tau_{x,y}$ given by eq.(5). The FFT unit consists of an AGU and a one-dimensional FFT (1D-FFT) module. To perform two-dimensional FFT (2D-FFT) using the 1D-FFT module, rows of the array $\{\rho_{x,y}, 0 \leq x, y \leq N-1\}$ are loaded from on-chip RAM and operated *one at a time*. The FFT unit then writes the computational results directly back to the same row in the on-chip memory. After the row operations are completed, the column operations will proceed in the same manner. After the completion of all the column operations, the array stored in the on-chip RAM is $\{\tau_{x,y}, 0 \leq u, v \leq N-1\}$, the 2D-FFT of $\{\rho_{x,y}, 0 \leq x, y \leq N-1\}$.

We use Altera FFT MegaCore function [15] to implement the 1D-FFT module. Because one row or one column is operated at a time, the transform length of the FFT is $N$. The 1D-FFT module has single data input and single data output. The module is fully pipelined. In addition, the input/output dataflow of the module is able to operate in streaming mode, allowing the continuous process of input data stream, as well as producing the continuous output data stream.

Figure 3 shows the operations of the 1D FFT module. There are 3 steps for the computation of 1D FFT using the module. The goal of the first step is to load data to the module. In this step, *N* inputs to FFT computation are loaded from on-chip RAM one at a time. Therefore, there are *N* clock cycles in this step. The second step performs FFT computation. This step will be activated when *N* inputs are available in the module. The number of clock cycles required for 1D FFT computation is denoted by $L_2$. The third step is for write back, where the *N* outputs produced from FFT computation is sent back to on-chip RAM one at a time. The latency of 1D FFT module is defined as the number of clock cycles between each input and its corresponding output. Because the write back operations are in order, it suffices only to consider the time between the first input and the first output as the latency, which equals to $N + L_2$, as shown in Figure 3.

To perform 2D FFT using the 1D FFT module, the employment of AGU is necessary. The AGU in the FFT unit generates addresses for loading the source data from on-chip RAM and writing the results produced by 1D-FFT module to the on-chip RAM. Because the 1D-FFT has single data input and single data output, two addresses are generated in each clock cycle: one for loading data, and another for writing result. In addition, because the 1D FFT module is fully pipelined, and is able to operate in streaming mode, consecutive rows (or columns) can be loaded to the module in a seamless way, as shown in Figure 4. We then see from Figure 4 that the total computation time for row operations is *N2+L2+N* clock cycles. Likewise, the total computation time for column operations is *N2+L2+N* clock cycles. The total computation time for the 2D FFT is then $2(N2 + L2 + N)$ clock cycles.

Let the array $\{\beta_{x,v}, 0 \leq \mathrm{x}, v \leq N-1\}$ be the results obtained from the row operations over $\{\rho_{x,y}, 0 \leq x, y \leq N-1\}$. Figure 5.(a) shows the FFT unit in the mode of row operations, where the input to the 1D FFT module is $\rho_{x_1,y_1}$ while the output produced by the module is $\beta_{x_2,y_2}$. Because the latency of 1D FFT module (observed from Figure 3) is $N + L2$, it follows that $(x_1N + y_1) - (x_2N + v_2) = N + L_2$. The FFT unit in the mode of column operations is depicted in Figure 5.(b). In this mode, the input to the 1D FFT module is $\beta_{x_1,v_1}$ while the output produced by the module is $\tau_{u_2,v_2}$, where $(x_1 + v_1N) - (u_2 + v_2N) = N + L_2$.

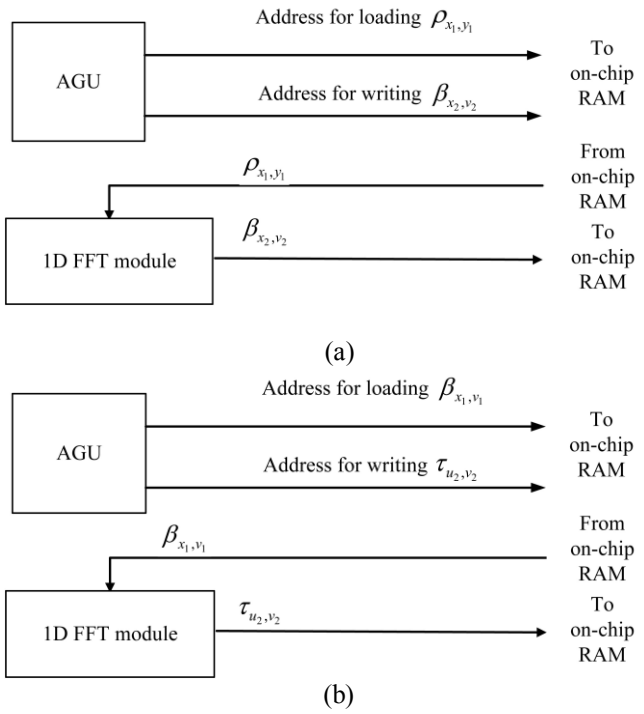



Fig. 5. The architecture of FFT unit: (a) in the mode of row operations, (b) in the mode of column operations.

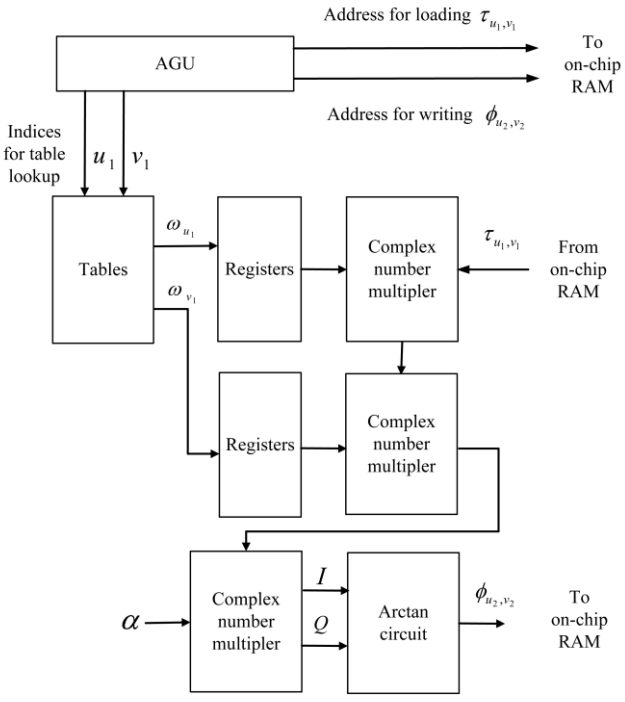



Fig. 6. The architecture of post-transform unit.

## *E. Post-transform Unit*

The post-transform unit is responsible for reconstructing the object image $B_{u,v}$ using eq.(7). As depicted in Figure 6, the architecture of the post-transform unit is similar to that of the pre-transform unit, comprising of an AGU, tables, multipliers, and delay units. The only difference is that the post-transform circuit contains an additional arctan circuit for phase computation.

In the post-transform unit, the tables are used to store the pre-computed values of $\omega_u$ and $\omega_v$. Similar to the cases for $\mu_x$ and $\mu_y$, because $0 \leq u, v \leq N - 1$, each table for the computation of $\omega_u$ and $\omega_v$ contains *N* entries. The AGU in the post-transform unit operates in the similar fashion to that of the pre-transform unit. The AGU produces indices (i.e., $u$ and $v$ values) for loading $\omega_u$ and $\omega_v$ values from the tables. It also

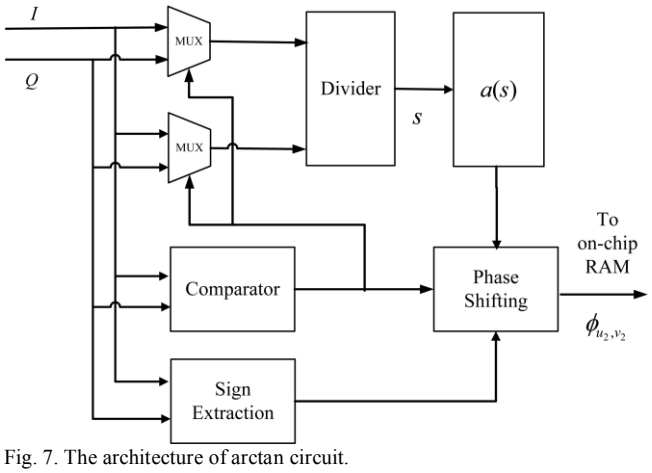



Fig. 7. The architecture of arctan circuit.

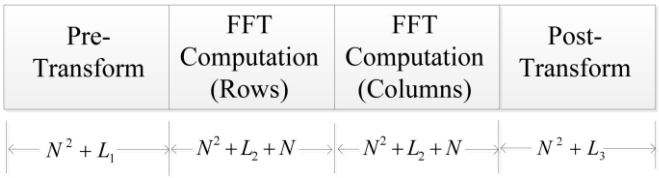



Fig. 8. The operations of the proposed circuit.

generate addresses to the on-chip RAM for loading $\tau_{u,v}$. The result of multiplication, $B_{u,v}$, is delivered to the arctan circuit for computing the phase $\phi_{u,v}$. The results of phase computation are then stored back to on-chip RAM. The design of arctan circuit is based on the approximation presented in [16].

Similar to the pre-transform unit and FFT unit, all the components in the post-transform unit are fully pipelined. Let $L3$ be the latency of the post-transform unit, defined as the number of clocks required to compute output $\phi_{u,v}$ from input $\tau_{u,v}$. Let $\tau_{u_1,v_1}$ and $\phi_{u_2,v_2}$ be the input and output to the post-transform unit in Figure 6, respectively. For the raster scanning order, it then follows that $u_1 \geq u_2$, and $(u_1 N + v_1) - (u_2 N + v_2) = L_3$. The total computation time for the post transform unit is $N^2 + L_3$.

### *F. Architecture Operations*

In the proposed architecture, the pre-transform unit, FFT unit and post-transform unit operate in sequence. That is, the pre-transform unit start the execution first, and produce $\rho_{x,y}$. The FFT unit will start the execution only after all the elements in the array $\{\rho_{x,y}, 0 \leq x, y \leq N-1\}$ have been stored in the on-chip RAM. Similarly, the post-transform unit will start the execution only after all the elements in the array $\{\tau_{u,v}, 0 \leq u, v \leq N-1\}$ produced by FFT unit are available in the on-chip RAM. Figure 8 shows the operations of the circuit. It can be observed that the total computation time is the sum of the computation time of the individual units.

## III. EXPERIMENTAL RESULTS

This section presents some experimental results of the proposed architecture. We first consider the area complexities of the proposed architecture. Because adders, multipliers, dividers and registers are the basic building blocks of the proposed architecture, the area complexities are separated into four categories: the number of adders, the number of multipliers, number of dividers and the number of registers. Table 1 shows the area complexities of the proposed architecture. It can be observed from the table that all the arithmetic operators do not grow with the image size. Only the number of registers is dependent on the image size.

Next we consider the physical implementation of the proposed architecture. The design platform is Altera Quartus II with SOPC Builder and NIOS II IDE. The hardware resources consumption of each unit in the proposed architecture is revealed in Table 2.

The images resolutions considered in the table are $128 \times 128$ (i.e., $N = 128$), $256 \times 256$ (i.e., $N = 256$) and $512 \times 512$ (i.e., $N = 512$). The target FPGA device is Altera Stratix III EP3SL. There are three types of area costs considered in the experiment: adaptive logic modules (ALMs), embedded memory bits, embedded multipliers. The ALMs are used for the implementations of arithmetic operators and registers. The embedded memory bits are used mainly for the design of on-chip RAM. The multipliers are used only for arithmetic operators such as multipliers in the FFT unit.

Table 3 reveals the actual computation time and throughput of the proposed architecture for various image sizes. The clock rate is 500 MHz (i.e., the period is 2 ns) for the experiments. In our experiments, the throughput is defined as the number of pixels of an object image can be reconstructed per second. It can be observed from Table 3 that the proposed architecture has fast computation time and high throughput. In particular, when the image size is $512 \times 512$, the throughput is 124.73 Mpixels/sec. Therefore, the maximum frame rate of the proposed circuit is 475 frames per second for frames with size $512 \times 512$.

Table 4 lists the throughput of various existing diffraction calculation implementations. The direction comparisons of these implementations may be difficult because these implementations are based on different algorithms with different image sizes. In addition, these implementations are realized by different platforms. Nevertheless, we can still see from Table 4 that the proposed circuit in parallel mode has highest throughput. Therefore, it is an effective alternative for diffraction calculation when high speed computation is an important concern.

Figures 9 and 10 show the reconstruction results of the proposed architecture. The images considered in the experiments are produced by the digital holographic microscopies (DHMs) in the IOP lab at the Institute of Electro-Optical Science and Technology, National Taiwan Normal University.

TABLE I
THE AREA COMPLEXITIES OF THE PROPOSED ARCHITECTURE.

| | Pre-transform Unit | FFT Unit | Post-transform Unit | On-Chip RAM | Total |
|---|---|---|---|---|---|
| Adders | $O(1)$ | $O(1)$ | $O(1)$ | 0 | $O(1)$ |
| Multipliers | $O(1)$ | $O(1)$ | $O(1)$ | 0 | $O(1)$ |
| Dividers | 0 | 0 | $O(1)$ | 0 | $O(1)$ |
| Registers | $O(N)$ | $O(N)$ | $O(N)$ | $O(N^2)$ | $O(N^2+N)$ |

TABLE II
THE CONSUMPTION OF HARDWARE RESOURCES OF EACH UNIT IN THE PROPOSED ARCHITECTURE FOR VARIOUS IMAGE SIZE.

| Sizes | Hardware Resources | Pre-transform Unit | FFT Unit | Post-transform Unit | On-Chip RAM |
|---|---|---|---|---|---|
| $128 \times 128$ | ALMs | 3642 | 7557 | 7544 | 201 |
| | Memory bits | 0 | 24567 | 9216 | 1053600 |
| | Multipliers | 32 | 24 | 88 | 0 |
| $256 \times 256$ | ALMs | 4123 | 7490 | 7995 | 339 |
| | Memory bits | 0 | 58368 | 9216 | 4210720 |
| | Multipliers | 32 | 24 | 88 | 0 |
| $512 \times 512$ | ALMs | 4977 | 7910 | 8347 | 796 |
| | Memory bits | 0 | 116736 | 9216 | 16810016 |
| | Multipliers | 32 | 24 | 88 | 0 |

TABLE III
THE SPEED OF THE PROPOSED ARCHITECTURE FOR VARIOUS IMAGE SIZE

| | $128 \times 128$ | $256 \times 256$ | $512 \times 512$ |
|---|---|---|---|
| Time (ms) | 0.1325 | 0.5266 | 2.1017 |
| Throughput (Mpixels/sec) | 123.65 | 124.45 | 124.73 |

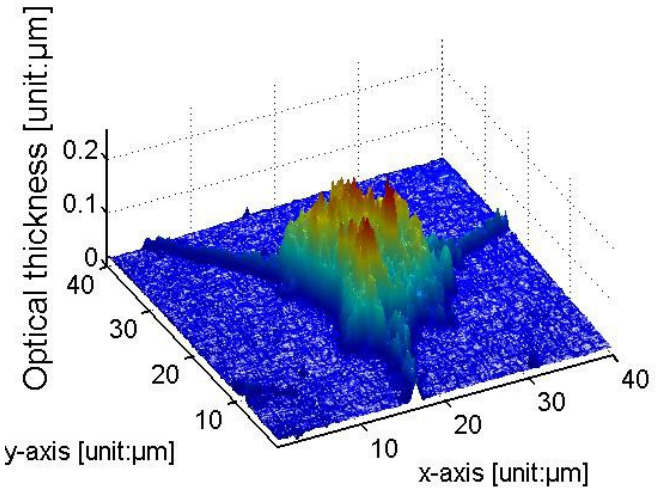
The reconstructed images shown in Figure 10 are the 3D images of microlens array and neural cells. The size of the images is $256 \times 256$. The reconstructed images shown in Figure 11 is the 3D image of a single microlens with size $512 \times 512$. Table 5 shows the mean squared distance (MSD) between the reconstructed images produced by the proposed architecture and its software counterpart. The MSD is defined as $\frac{1}{N^2}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}(\phi_{u,v}-\bar{\phi}_{u,v})^2$, where $\bar{\phi}_{u,v}$ is the phase produced by software. The software Fresnel transform and phase computation are implemented by Matlab. We can observe from Figures 10 and 11 that the reconstructed images have high visual quality. This is because single precision floating number format is adopted in the proposed implementation. As shown in Table 5, the MSD between the hardware design and its software counterpart is very small. While having high speed computation, the architecture is also able to achieve high accuracy for 3D reconstruction.

TABLE IV
THE THROUGHPUT OF VARIOUS IMPLEMENTATIONS FOR DIFFRACTION COMPUTATION

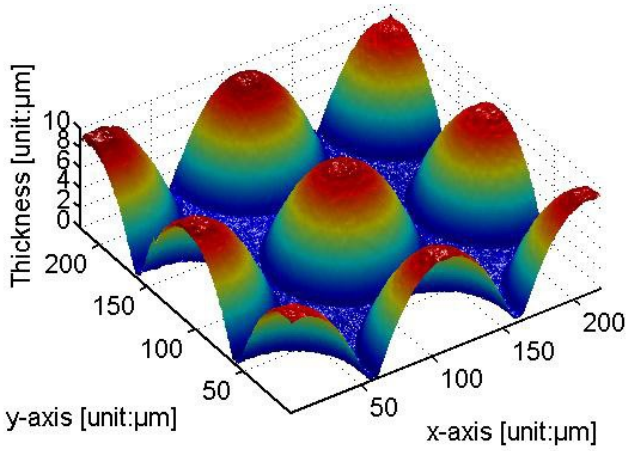
| Implementations | Throughput | Image Size | Platforms |
|---|---|---|---|
| Proposed Architecture | 124.73 Mpixels/sec | $512 \times 512$ | FPGA Altera Stratix III EP3SL |
| Pandey et al. [7] | 15.9 Mpixels/sec | $2000 \times 2000$ | GPU NVIDIA Geforce 8800 GTX |
| Zhu et al. [8] | 1.27 Mpixels/sec | $2200 \times 2900$ | GPU NVIDIA Quadro NVS 135M |
| Shimobaba et al. [9] | 6.53 Mpixels/sec | $512 \times 512$ | GPU NVIDIA Geforce 8800 GTX |
| Nishitsuji et al. [10] | 44.4 Mpixels/sec | $512 \times 512$ | GPU AMD Cypress HD 5850 |
| Masuda et al. [12] | 25.2 Mpixels/sec | $256 \times 256$ | FPGA Xilinx Virtex II pro XC2VP70 |
| Abe et al. [13] | 31.78 Mpixels/sec | $1024 \times 1024$ | FPGA Xilinx Virtex II pro XC2VP70 |

TABLE V
THE MSD BETWEEN THE RECONSTRUCTED IMAGES PRODUCED BY THE PROPOSED ARCHITECTURE AND ITS SOFTWARE COUNTERPART

| | Neural Cell | Microlens Array | Single Microlens |
|---|---|---|---|
| Size | $256 \times 256$ | $256 \times 256$ | $512 \times 512$ |
| MSD | $9.2832 \times 10^{-9}$ | $9.4158 \times 10^{-9}$ | $3.6689 \times 10^{-9}$ |




(a)




(b)

Fig. 9. The 3D reconstruction of object images by the proposed architecture.( a) a neural cell with image size $256 \times 256$ , (b) a microlens array with image size $256 \times 256$ .

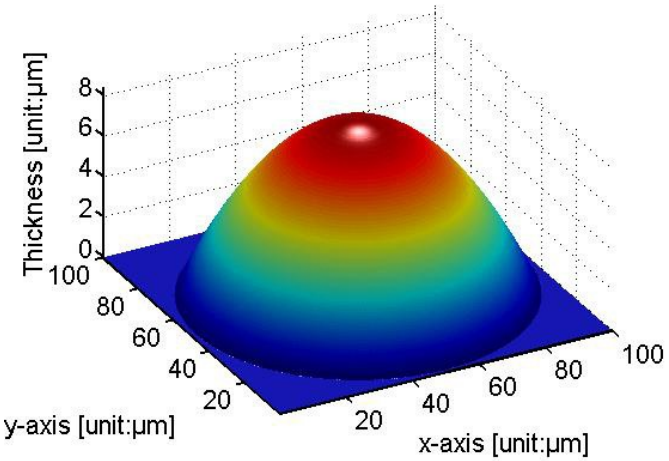



Fig. 10. The 3D reconstruction of a single microlens by the proposed architecture with image size $512 \times 512$.

## IV. CONCLUDING REMARDS

Experimental results reveal that the proposed architecture has the advantages of high computation speed, low power consumption, and high accuracy.When operating at 500 MHz, the proposed architecture is able to achieve throughput of 124.73 Mpixels/sec. The corresponding frame rate is 475 frames per second for images with size $512 \times 512$. The architecture is also able to produce 3D reconstruction results with quality comparable to its software counterpart implemented by Matlab. The proposed architecture therefore is beneficial for the high quality 3D rendering of holograms for the resource-limited end devices.

## REFERENCES

[1] U. Schnars and W.P. Jueptner, Digital Holography, Springer-Verlag, 2005.

[2] J. Mundt and T. Kreis, Digital holographic recording and reconstruction of large scale objects for metrology and display, *Optical Eng.*, Vol. 49, 2010.

[3] B. Kemper and G. von Bally, Digital holographic microscopy for live cell applications and technical inspection, *Applied Optics*, Vol. 47, pp. A52-A61, 2008.

[4] Y. Emery, E. Cuche, F. Marquet, N. Aspert, P. Marquet, J. Kuhn, M. Botkine, T. Colomb, F. Montfort, F. Charriere, C. Depeursinge, Digital Holography Microscopy (DHM): Fast and robust systems for industrial inspection with interferometer resolution, *Proc. SPIE*, Vol. 5856, 2005.

[5] T. Kreis, Digital Holography Methods in 3D-TV, *Proc. IEEE 3DTV Conference*, 2007.

[6] M. Kim, L. Yu and C. Mann, Interference techniques in digital holography, *J. Opt. A: Pure Appl. Opt.*, Vol. 8, pp. S518S523, 2006.

[7] N. Pandey, D. P. Kellya, T. J. Naughtona and B. M. Hennellya, Speed up of Fresnel transforms for digital holography using precomputed chirp and GPU processing, *Proc. SPIE*, Vol. 7442, 2009.

[8] Z. Zhu, M. Sun, H. Ding, S. Feng and S. Nie, Fast numerical reconstruction of digital holography based on graphic processing unit, *Proc. IEEE Pacific Rim Conference on Lasers and Electro-Optics*, 2009.

[9] T. Shimobaba, Y. Sato, J. Miura, M. Takenouchi and T. Ito, Real-time digital holographic microscopy using the graphic processing unit, *Opt. Express*, Vol. 16, pp.11776-11781, 2008.

[10] T. Nishitsuji, T. Shimobaba, T. Sakurai, N. Takada, N. Masuda, and T. Ito, Fast calculation of Fresnel diffraction calculation using AMD GPU and OpenCL, OSA Technical Digest, 2011.

[11] S. Hauck and A. Dehon, Reconfigurable Computing: The Theory and Practice of FPGA-Based Computing, Morgan Kaufmann: San Fransisco, CA, USA, 2008.

[12] N. Masuda, T. Ito, K. Kayama, H. Kono, S. Satake, T. Kunugi and K. Sato, Special purpose computer for digital holographic particle tracking velocimetry, *Opt. Express*, Vol. 14, pp.587-592, 2006.

[13] Y. Abe, N. Masuda, H. Wakabayashi, Y. Kazo, T. Ito, S. Satake, T. Kunugi, and K. Sato, Special purpose computer system for flow visualization using holography technology, *Opt. Express*, Vol. 16, pp.7686-7692, 2008.

[14] Y. L. Lee, Y. C. Lin, H. Y. Tu, and C. J. Cheng, Phase measurement accuracy in digital holographic microscopy using a wavelength-stabilized laser diode, *Journal of Optics*, 025403, 2013.

[15] Altera Corporation, FFT megaCore function user guide, 2012.

[16] S. Rajan, S. Wang, R. Inkol, A. Joyal, Efficient Approximations for the Arctangent Function, *IEEE Signal Processing Magazine*, Vol. 23, pp.108- 111, 2006.

[17] Altera Corporation. Quartus II Handbook Ver 13.0, 2013; Volume 3. Available online: http://www.altera.com/literature/lit-qts.jsp (accessed on 9 September 2013).

[18] S. Collange, D. Defour, and A. Tisserand, Power Consumption of GPUs from a Software Perspective, *Lecture Notes in Computer Science*, Vol. 5544, pp.914-923, 2009.

[19] T. Matsumoto, S. Yamaguchi, and T. Sakai, A Study on Improving Power-Consumption Performance Ratio in GPGPU Computing, *Proc. IEEE International Conference on Networking and Computing*, pp.288- 290, 2011.