Hardware Implementations of Computer Generated Holography: A Review

Youchao Wang[a, b], **Daoming Dong**[a, b], **Peter J. Christopher**[a], **Andrew Kadis**[a], **Ralf Mouthaan**[a], **Fan Yang**[a], **Timothy D. Wilkinson**[a,*]

[a]University of Cambridge, Centre for Molecular Materials, Photonics and Electronics, Department of Engineering, 9 JJ Thomson Avenue, Cambridge, UK, CB3 0FA

[b]Both author contributed equally.

Abstract. Computer generated holography (CGH) is a technique to generate holographic interference patterns. One of the major issues related to computer hologram generation is the massive computational power required. Hardware accelerators are used to accelerate this process. Previous publications targeting hardware platforms lack performance comparisons between different architectures and do not provide enough information for the evaluation of the suitability of recent hardware platforms for CGH algorithms. We aim to address these limitations and present a comprehensive review of CGH-related hardware implementations.

Keywords: Computer generated holography (CGH), Central proccesing unit (CPU), Graphics processing unit (GPU), Fied-programmable gate array (FPGA), Digital signal processor (DSP), Hardware accelerator, Holography, System-on-Chip (SoC).

* Corresponding author: Timothy D. Wilkinson, tdw13@cam.ac.uk

1 Introduction

Holography is a technique used to record and reconstruct the entirety of an optical field.[1] This approach was pioneered by Dennis Gabor in 1948 as a two-step, lensless imaging process for improving the quality of electron microscopy.[2]

In the early days, holograms were primarily single-use as the only recording media available resembled photographic film. It was not until the mid-1960s when computer generated holography (CGH),[3] together with the noticeable improvements in technology, revolutionized the field and drew a significant amount of interest.

The late 1980s saw a further shift in holography from analogue to digital with the emergence of digital imaging sensors as well as increases in computational powers and electronic display devices such as digital micromirror devices (DMDs) and liquid crystal spatial light modulators

(LC SLMs). Holograms could, for the first time, be digitally captured, processed and displayed. Over time, holography has become regarded as a serious display technology for far-field and 3D applications.[4]

CGH is the field of algorithmically generating holographic interference patterns using digital computers, with target applications including but not limited to display technologies[5], wavelength selective switch (WSS)[6], optical tweezers[7] and telecommunications.[8] Generating computer holograms in real-time is one of the key goals of research, with algorithms for CGH traditionally running on central processing units (CPUs). Despite recent increases in the processing power of CPUs, it remains insufficient for real-time photographic applications. Accelerated hardware platforms, including graphics processing units (GPUs), field programmable gate arrays (FPGAs), digital signal processors (DSPs), co-processors as well as application-specific integrated circuits (ASICs), are able to bring high fidelity holographic imagery to real-time applications.

Figure 1 shows a typical system setup for a CGH. The creation of the computer holograms can be divided into three parts:[1]

1. **Calculate**: to allow the computer to digitally, instead of optically, calculate the interference fringes for a target object;
2. **Encode**: to determine the method to represent or encode the computation results;
3. **Display**: to display the encoded fringes on a suitable medium.

CGH algorithms, regardless of them being point-source-based, polygon-based, layer-based, etc., would typically require a very high level of computational power. Hence, when designing any new holographic systems, the selection of a suitable hardware platform is the primary decision to be made.

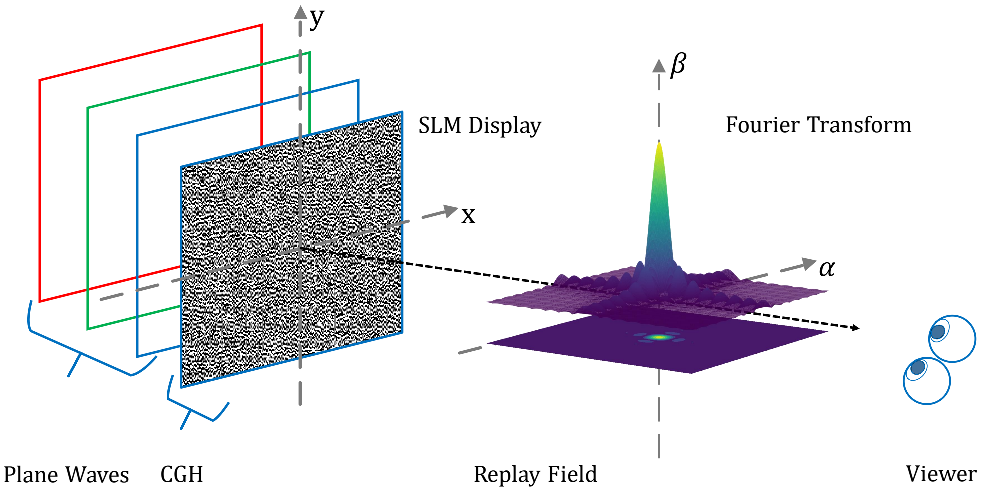

Fig 1 A typical system for computer generated holography consisting of three main components: a light source, a computer or hardware platform for interference pattern calculation and a device to display the hologram.[4]

To the best of our knowledge, there is no modern review paper that specifically targets the hardware used for the generation and processing of computer holography. Previously published survey papers[9–14] provide analyses and conceptual reviews of fast hologram generation algorithms. Additionally, Shimobaba *et al.* in 2016[15] and 2019[16] provided overviews in terms of CGH-related hardware implementations. However, all of the above reviews suffer from a lack of the following:

1. A comparison between different hardware platforms;
2. A dedicated discussion with respect to hardware implementations;
3. An assessment of the trade-offs between different development factors for a given hardware platform;
4. An up-to-date review with respect to the most recent developments in modern hardware.

We aim, therefore, to provide review by comparing different hardware platforms and discussing each platform's advantages and disadvantages. This review paper considers CPUs, GPUs, FPGAs and other hardware accelerators in dedicated sections. For each platform, we provide a literature survey on the applications utilizing these specific hardware platforms. This is followed by a discussion of device properties, available development toolchains, the ease of development, and their advantages and disadvantages. We also present cross-platform comparisons to gain insights regarding the use of different types of accelerators. Generally, we provide a thorough examination of the current state-of-the-art hardware implementations along with a review of their applications over the previous decade (2008-2020).

This literature survey is outlined as follows. Section 1 first introduces the field holography alongside key concepts and a discussion of the CGH challenges. CPU, GPU, FPGA and other platform implementations are discussed in Sections 2, 3, 4 and 5, respectively. Section 6 reports the comparison between different hardware platforms and provides in-depth discussion to guide hardware selections. Finally, the paper is concluded after presenting future work directions in Section 7.

1.1 The hologram and the replay field

In a classical imaging system, Figure 2, focusing optics are used to focus light scattered from a point of an object onto a corresponding point on a sensor (the recording device). In such a system, any different origin point on the object leads to a corresponding change in the position on the recording plane of the sensor. The loss of a portion of the sensor data will result in a corresponding loss in the image.

In a holographic imaging system, Figure 3, scattered light is collected without the use of a focusing optics, instead interfering the scattered light with a reference beam. Replicating recording conditions allows for replication of the light field and the resulting image as depicted in Figure 4. The image is stored across all parts of the recording device leading and loss of a portion of the

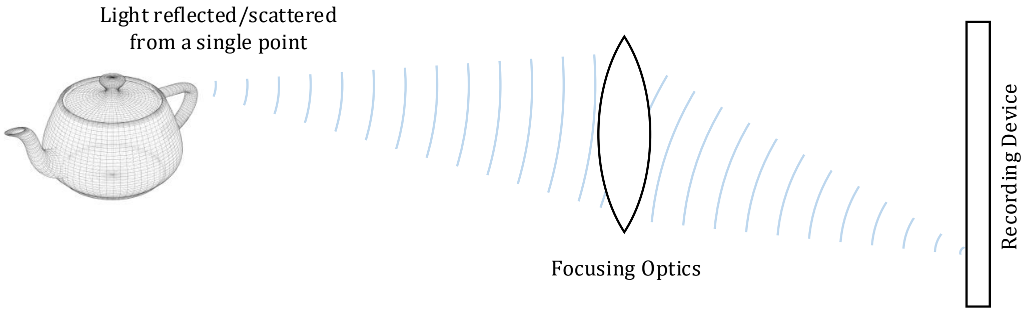

Fig 2 A classical optical imaging system.

recording only causes a loss in the quality of the image. The entire image can still be reproduced.

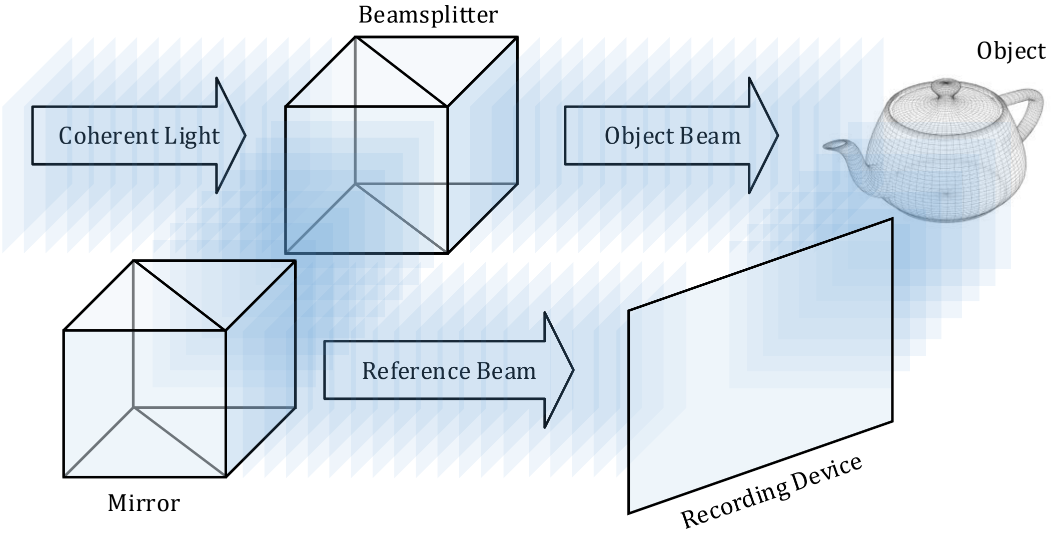

Fig 3 A holographic imaging system for hologram recording.

Traditional analogue holography follows two steps known as recording and reconstruction:[1,2,17]

1. **Recording** - Figure 3 - A coherent, collimated light source is split into *object* and *reference* beams. The object beam is directed onto a physical object and the resulting scattered light interfered with the reference beam. The interference fringes are recorded on a photosensitive film to produce the hologram.

2. **Reconstruction** - Figure 4 - A similar system is used to reproduce the hologram. An identical light source is directed onto the film leading to a visible output equivalent to viewing the

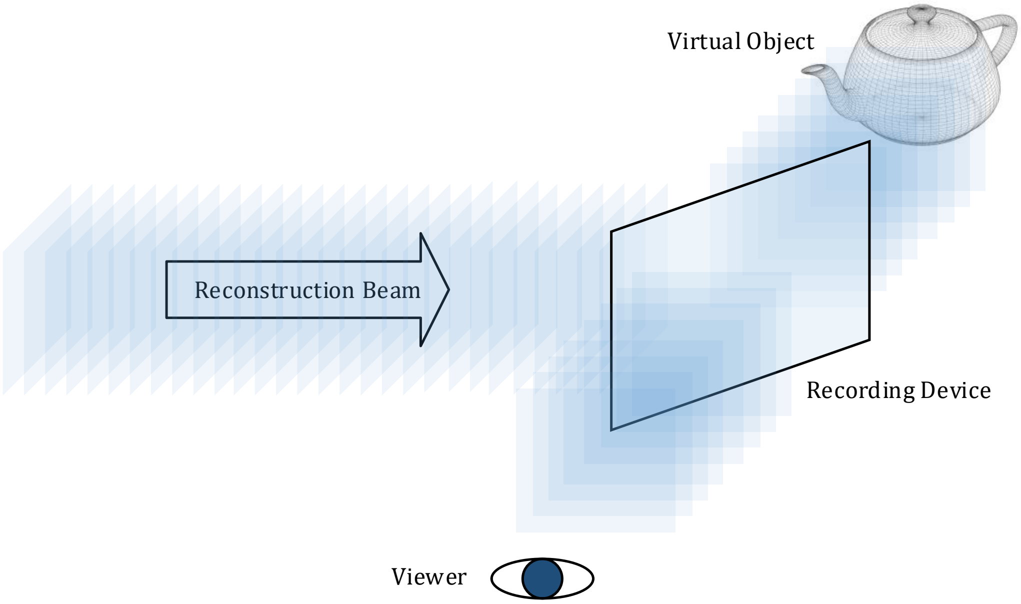

Fig 4 A holographic projection system for hologram reconstruction.

object directly.

Computer generated holography goes further than this by using a known target or scene to generate the reconstruction image, and thus eliminating the requirement of a recording step.

1.2 Limitations of Computer Generated Holography (CGH)

Computer generated holography promises a great deal; however, in practice there exist a number of key limitations:

1. **Hologram representation.** Complex modulation schemes are achievable by several means and methods[18,19], despite the fact that display medium, such as SLMs, are still facing technological limitations to perform true arbitrary complex modulations. However, these methods often require non-trivial modifications and device setups, consequently limiting the repre-

sentations of holograms. Moreover, the hardware manufacturing constraints limit the size and quality of the reconstructed holographic images as well as the viewing angle.[16]

2. **High computational power demand.** The hologram and object field is correlated by *Fourier transforms* for any given pixel display. For a single image frame with $N \times N$ points or pixels, the computation complexity can be as high as $O(N^4)$. By utilizing the power of fast Fourier transforms (FFTs), we are able to reduce this complexity to $O(N^2 \log(N))$. Unfortunately, this is still computationally expensive, before even considering the inclusion of other operations for any given algorithms to produce high quality images and videos where the incorporation of visual effects such as shading[20], occlusion effects[21,22], directional scattering[23] are of great essence. Such high quality image demand is one of the key limitations.

3. **Downgrade of the replay field image quality.** The quality of the holographic reconstructed image would be affected by factors such as speckle noise[13], ringing artifacts[24], the opto-mechanical properties of SLMs[25], etc. Moreover, in real-world display devices are incapable of modulating light continuously; being limited to a number of discrete levels results in quantization artifacts which have an adverse effect on image quality.[26] During this process, the information stored in the interference pattern will be reduced, leading to a degradation in image quality.

A suitable hardware platform for CGH algorithm implementations needs to be selected in order to speed up the generation of computationally heavy holograms while ideally also improve replay field quality. In this paper, we aim to address this problem by providing a selection guideline for researchers and developers to choose the most suitable hardware platforms for computer hologram applications. While we stay focused on the hardware choice, it should be pointed out that such choices are also intimately related to the algorithm selected as some would require more dedicated and specialized hardware resources as compared to others. A further discussion of such is outlined in Section 6.

We divide the current state-of-the-art hardware platforms into two categories: conventional processors where we refer to CPUs; and hardware accelerators such as GPUs, FPGAs, DSPs and co-processors. Traditionally, basic arithmetic calculations were done in CPUs. However, for certain computationally expensive applications, there is a need for specialized architecture where the design is optimized for the application to accelerate performance.

Hardware accelerators were designed to tackle this issue by exploiting properties such as parallelism and application-specific dedicated hardware accelerations. These devices are usually based on different architectures and inherently make use of different development tools and utilities. The code and algorithm migrations between these hardware platforms are not often straightforward. They require a good understanding of the specific hardware architectures as well as microarchitectures in order to carry out code implementations and optimizations properly.

2 Central Processing Units (CPUs)

Since its invention in the early 1970s, CPUs have become the core of this ever-developing digital world. Von-Neumann, Harvard architectures and their architectural variants will continue to dominate the market in the foreseeable future. The fundamental operations and underlying theories remained largely unchanged throughout the years. These CPUs are designed to complete computational tasks that are as general as possible. Unfortunately, it is this very generality which prevents CPUs from executing high-performance computational operations since they lack sufficient amount of parallelism within their architectures.[27]

It was not until 2005 when Intel introduced the Pentium D series—the first desktop-class dual-core processor—that exploited parallel processing for individual consumer computers running multi-core processors. A typical contemporary computer with a multi-core processor can run tens and hundreds of tasks at any given time. Running multiple programmes simultaneously utilises concurrency by switching and jumping between different threads, or instruction streams, under real-time.[28] The job-switching operations take up and waste CPU cycles and would hence prevent the platform to run at optimal efficiency when performing multi-tasking and exploiting parallel processing.

For this paper, we will only evaluate Intel and AMD chip families as they are the two vendors to produce x86/64 architecture, the dominant high-performance CPU architecture at the time of writing, design.

2.1 A platform for preliminary verification of algorithms

Most hologram generation algorithms were developed on conventional computers, utilizing the power of the latest CPU chip families. Software-based algorithms run on CPUs to efficiently minimize the development time and reduce the computational burden by exploiting advanced computation libraries, software packages and other utilities.

Reported work based purely on CPUs form the preliminary analysis of various proposed computer hologram generation algorithms. Researchers tend to focus more on theoretical development rather than code optimization since conventional CPUs are not used for acceleration purposes.

Due to their commonality and the ease-of-use, the majority of work that incorporate CPUs often use them as the comparison baseline for algorithm implementations on other hardware platforms that utilize dedicated accelerators.

2.2 *Available tools and utilities for CPUs*

Since CPUs are the core components within a modern personal computer (PC) and workstation, the vast majority of software packages and development suites are readily available. Code and programmes can be written in many high-level languages, while low-level application programming interfaces (APIs) and frameworks, such as OpenMP and OpenCV, are also widely available. As an API for shared memory multiprocessing, OpenMP is dedicated to high-level parallelism in Fortran and C/C++ programs.[29] Compiler directives, library routines as well as environment variables can be used to optimize for multiprocessing by, for example, distributing workloads among the available threads and physical-cores.

We endeavour to conclude the tools and utilities that have been reported in previously published papers since 2008, as shown in Table 1. The most commonly used software application is Matlab, due to its simplicity and numerous package supports. No strict understandings in terms of hardware architectures and memory management are necessary when developing algorithms over Matlab, as compared to other realization methods. C/C++ tend to be the most popular programming language used for algorithm implementations. C/C++ programming libraries and functions, such as FFTW, cvDFT from OpenCV and custom library CWO++,[30] offer strong support for improved hologram generation performances.

2.3 *The advantages and disadvantages of using CPUs*

The most significant advantage of using CPUs for algorithm implementation is the short development time and sophisticated software toolchain support. Nearly all the software packages that can be found on other hardware accelerator platforms have the same or equivalent toolkits which are available on CPU-based PCs. These ranges from programming languages, such as C/C++ and

Table 1 Tools and utilities employed for CPU implementations since 2008

Name	Category	Appearance	Year
Matlab	Software	Novel LUT algorithm,[31] Fast computation,[32] Compressed LUT algorithm,[33] Binary detour phase holograms,[34] Specific solutions for Gerchberg-Saxton (GS) algorithm,[35] Highly efficient calculation,[36] Rotational transformation of wavefields[37]	2008, 2009, 2013, 2014, 2019
CWO++ library (CWO: Computational Wave Optics)	Customized C++ library	CWO++ library (CWO for CPU),[30] Wavelet ShrinkAge-Based superpositIon (WASABI) using CWO++[38–40]	2012, 2017, 2018
FFTW library	FFT library	Wavefront recording plane (WRP) GPU comparison,[41] CWO++[30]	2009, 2012
Intel Math Kernel Library (MKL)	Math and FFT library	Polygon-based extremely high-definition projection[42]	2009
cvDFT (OpenCV)	FFT function from OpenCV	Wavefront recording plane[43]	2018
OpenMP	Multi-processing API	Baseline for multi-GPU cluster comparison[44]	2012
C	Programming language	Simulated annealing (SA) GPU comparison,[45] Multi-GPU cluster comparison[44]	2010, 2012
C++	Programming language	Polygon-based extremely high-definition projection,[42] WRP GPU comparison,[41] SA GPU comparison,[45] CWO++ and WASABI,[30,38–40] Full colour and colour space conversion using WASABI[46]	2010, 2012, 2017, 2018, 2019
Python	Programming language	Compressive-sensing GS[47]	2019

Python, compile-time and run-time libraries, to software packages.

Other merits of using CPUs are as follows:

1. **Comparatively high clock frequency**: Contemporary CPUs run in GHz domain as compared to the frequencies in other hardware that are usually between hundreds of MHz to above 1 GHz. Higher clock rates provide shorter clock cycles, consequently speeding up sequential processes.

2. **Floating point precision**: CPUs tend to have better support for double-precision floating point arithmetic from the tools that are available, although the use of full-precision components can downgrade run-time execution speed.

The disadvantages are also apparent. CPUs are optimized for sequential operations and consequently full parallelism cannot be achieved. Although state-of-the-art CPUs at the time of writing feature a higher level of parallelism than older CPUs, with tens of cores being available in a single package, this pales in comparison to the massively parallel architectures of GPUs and FPGAs which feature thousands of parallel execution units. Moreover the software libraries and APIs to support parallelism, such as OpenMP which help shorten the developing time needed for multi-thread and multiprocessing applications, exist but require an advanced level of skills to utilize effectively.

2.4 *Reported work using CPUs*

Most of the reported work covering CPU-based applications are for either algorithm developments or, more commonly, for establishing baselines for cross-platform performance comparisons.

The most common method to optimize the performance of hologram generation using a computer is to combine both CPUs and GPUs together.

Shimobaba *et al.* reported on the development of a C++ library CWO++, which is used for diffraction calculations.[30] This library has been developed to run on both CPU (CWO class) and GPU (GWO class, GPU-based wave optics), and has been used in various algorithm developments.[24,38–40,44,48–50]

We aim not to thoroughly review the work that reports on CPU-based platform performance, as in the majority of cases the CPU results are used to provide a baseline performance reference. However, the baselines are subsequently encountered in several throughout the survey.

2.5 *Summary of CPUs*

Contemporary CPUs offer insufficient performance for *real-time* CGH and hence it is not recommended to build a real-time holographic system based solely on them. Moreover,the readily availability of hardware accelerators, such as GPUs and FPGAs, provide further rationale for hologram generation algorithms to not be implemented purely on a CPU-only platform.

Algorithm developments in the initial phase, however, are one exception for purely CPU simulations and implementations, e.g., with MATLAB and Simulink, in order to significantly cut down the development time and improve the efficiency of research outputs. Moreoever, this approach also encourages collaboration, lowering the skills and knowledge barriers for other research groups to replicate and improve the corresponding algorithms.

3 Graphics Processing Units (GPUs)

In both academia and industry, GPUs, being the dedicated graphics accelerators, have gained much attention since their introduction in the late 1990s.[51] Through the use of parallel operations, these accelerators maximise the performance of image- and video-related applications.

Benefiting from economics of scale, GPU products are cost-effective and readily available. High-end products with a large count of processing units that perform parallel half (16-bit), single (32-bit) and double (64-bit) precision floating point operations in parallel are eminently suitable for image and video processing applications. The introduction of compute unified device architecture (CUDA)[52] by NVIDIA in 2007 further extends the ease of development and shortens the implementation as well as transplantation time. Due to their strong parallel performance and well-supported development environment, GPUs are one of the most effective hardware accelerators available on the market.

Traditionally, GPUs have been dedicated to graphics rendering. However, throughout years of development which have brought forth increases in computational power, contemporary GPUs are encroaching upon application domains that formerly belonged to high-end high-power CPUs.

These GPUs are regarded as general-purpose graphics processing units (GPGPUs). Non-specialized calculations, such as machine learning computations, scientific computations, heavy image/video editing, encryption/decryption, have been taken over by the use of GPGPUs based on their merits of having massive parallelism and large processing core counts as opposed to the traditional CPUs.

Two vendors, NVIDIA and AMD are major players in the graphics processing industry. Intel, with the recent development of its own GPU hardware, makes it another major producer of GPUs. However, based on the past lines of work, we will mainly focus the NVIDIA GPU families, since they are the most popular hardware platform used in the CGH and image processing community.[53]

3.1 The parallelism in GPUs

Architecturally, a GPU is significantly different from a CPU. The major difference being that GPUs exploit massive parallelism at the hardware level. A single mainstream contemporary GPU incorporates thousands of dedicated processor cores, whereas even the highest-end CPUs typically contain less than 24 cores.[54] It is this inherent parallelism that provides high-performance computation capability for highly parallel problem spaces.

3.2 GPU performance trends

Over the years, NVIDIA brought out a range of core microarchitectures in their GPU series, targeting both the professional high-performance uses as well as individual consumer level applications. The last decade has seen a large increase in the performance capability of GPUs, as summarized in Table 2[55,56]. While the rated power consumption has remained relatively constant (at around 200-300 Watts), we have seen a significant increase in processing power. Ever since Fermi being regarded as '*the first complete GPU computing architecture*'[57], ten years has seen NVIDIA

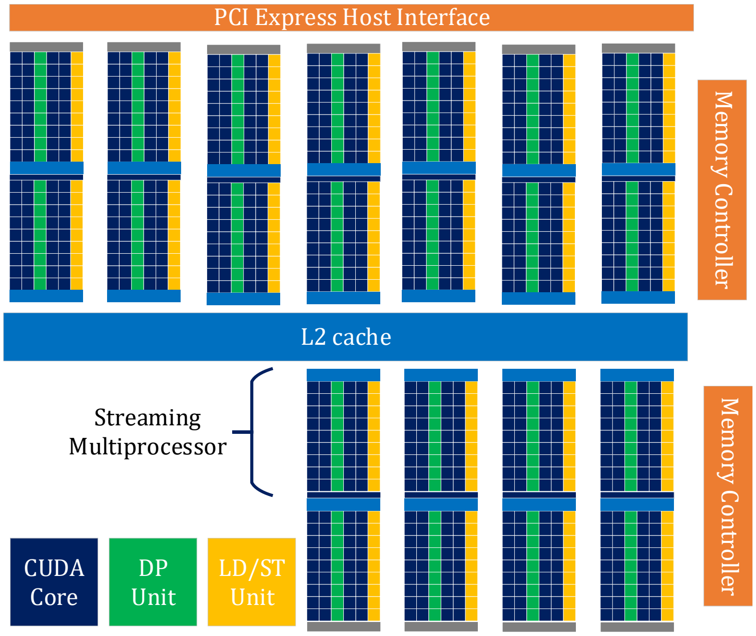

Fig 5 A typical parallel pipeline overview of an NVIDIA GPU consisting of streaming multiprocessors each containing a number of cores and functional units.

working on incorporating advanced shaders, hardware ray tracing and many high performance functionalities, with larger and faster processing capability and speed for not only graphics rendering but more general purpose usages[58] Consider the floating point operations per second (FLOPS) performance of a top-end GPU. Between 2008 and 2018, the performance has increased from 432 GFLOPS to 16312 GFLOPS, an improvement of more than 37×.

GPU designs vary between different microarchitectures and production families. Figure 5 shows a typical structural overview of an NVIDIA GPU with numerous streaming multiprocessors (SM) consisting of shared memories, L1/L2 caches, CUDA cores, arithmetic units (e.g. double precision unit), load/store (LD/ST) units, etc. This architectural setup reveals the high level of inherent hardware parallelism within modern GPU devices.

Table 2 Microarchitectures since 2008 and their representative flagship GPU products (SP: single precision floating point)[56]

Model	Year of launch	Microarchitecture	Transistors (million)	Fab (nm)	GFLOPS	TDP (watt)
9800 GTX	2008	Tesla	754	65/55	432	140
GTX 295	2009	Tesla	2× 1400	55	1192.3	289
GTX 480	2010	Fermi	3000	40	1344.96 (SP)	250
GTX 590	2011	Fermi	2× 3000	40	2488.3 (SP)	365
GTX 690	2012	Kepler	2× 3540	28	2× 2810.88 (SP)	300
GTX TITAN	2013	Kepler	7080	28	4499.7 (SP)	230
GTX TITAN Z	2014	Kepler	2× 7080	28	8121.6 (SP)	375
GTX TITAN X	2015	Maxwell	8000	28	6604.8 (SP)	250
GTX TITAN X 2	2016	Pascal	12000	16	10974.2 (SP)	250
GTX TITAN V	2017	Volta	21100	12	14899.2 (SP)	250
GTX TITAN RTX	2018	Turing	18600	12	16312.32 (SP)	280

3.3 Available tools and utilities for GPUs

Two utilities are widely used: Compute Unified Device Architecture (CUDA) platform and Open Computing Language (OpenCL) framework.

In 2007, a parallel computing platform and application programming interface (API) model, CUDA was released by NVIDIA. Prior to the introduction of CUDA, graphics and GPU programming skills for use in tools such as Direct3D, DirectX and OpenGL, with a good understanding in High Level Shader Language (HLSL) were required in order to take advantage of the high computational performance of graphics cards.[59] CUDA, however, only required standard C/C++ or Fortran programming language skills as the bare minimum.

As of the writing of this review, CUDA has iterated to its tenth generation (10.1) and comes with both compile-time and run-time libraries.[52] In particular, the CUDA fast Fourier transform (cuFFT) library enables high-performance FFT and IFFT computations similar to the FFTW library.[60] Other useful libraries provided includes but not limited to a basic linear algebra subroutine library, cuBLAS, useful for linear algebraic operations; a random number generation library, cuRAND, useful for random phase generations; and a parallel algorithms and data structures library, Thrust, to accelerate operations such as sum and average as well as boundary (maximum and minimum) search algorithms in parallel.

Developing programmes over CUDA is straightforward with the support of a modified C programming language dedicated to the CUDA framework.

Additionally, vendors such as NVIDIA and AMD have all provided full support and have released the implementations of OpenCL for their GPUs. OpenCL is a framework with low-level APIs for cross-platform computing. Developers can use the provided APIs from OpenCL to write programmes that run across CPUs, GPUs, etc., with C programming language. However, it is worth noting that a study (not related to holography) conducted by Memeti *et al.* in 2017 suggested that CUDA outperforms OpenCL in terms of productivity, requiring two times less programming effort for a specific benchmark suite.[61]

Matlab, in the meantime, provides a parallel computing toolbox for GPU computing. Despite some limitations, it is argued that combining both CUDA kernels and Matlab support (using the Parallel Computing Toolbox) can further improve and smooth the programming process.[34,62] No knowledge in CUDA is needed while exploiting the parallel computing capabilities for CGH-related computation speed-ups.

3.4 The advantages and disadvantages of using GPUs

GPUs are used for accelerating the processing of images and videos at birth. The hardware architectures are specially designed for this purpose by highly optimizing the parallel characteristics in both hardware and software.

The key advantage of GPGPUs is that they can be programmed using high-level programming languages such as C/C++, making code development and corresponding debug processes faster and easier than in other platforms such as FPGAs.

As shown in Table 2, one of the major disadvantages of using GPUs for algorithm implementation is their high power consumption. The thermal design power (TDP), which is the maximum amount of heat generated by the chip during operation and which serves as a basic indicator of power consumption, is typically around 200-300 watts.

A good understanding of GPU microarchitectures and, in particular, memory management is required for speed optimization, although dedicated utilities tend to offer modest support for the managing of the memory.

More importantly, most of the GPUs cannot work as a standalone platform. A system incorporating CPUs and other essential hardware devices tend to create limits on data throughput during read, fetch and write operations and would increase the overall power consumption. Additionally, this level of integration introduces a data transfer bottleneck, which downgrades the overall performance of the platform. The speed for data transfers between the host PC and the GPU or GPU cluster would even slow down when the implementation has not been properly optimized.

3.5 The development time using GPUs

The use of CUDA makes GPU implementations simpler. The majority of the development time will be spent on software coding using C/C++ programming language.

The major difficulty in the development of GPU hologram generation application is to optimize the codes for the potentially high-throughput and heavy computational requirements. This requires a good understanding of the GPU architectures, as well as hardware and software optimization techniques. However, since most of the fast algorithms implemented, such as in work,[45,66,73,74] require less sophisticated operations, the optimization can be based purely on increasing the data throughput and improving the computational power with parallel processing.

Table 3 A summary of CGH implemented on GPUs since 2008

Project and application (year)	Implemented algorithm	Hardware model (GFLOPS SP based on[56])	Performance
Holographic optical tweezers and 4π-microscopy (2008)[63]	Gerchberg-Saxton algorithm	Geforce 8800 GTX (345.6) and 8800 GTS (416)	One GS loop at 512×512 in 16.5 msec
Data-parallel computing for point cloud (2009)[64]	Nonuniform sampling, Common visibility group (CVG) approximation	Geforce 9800 GX2 (2×384)	Non-uniform 7592 points in 10.3 sec, CVG in 5.07 sec
Depth buffer rasterization for 3D display (2009)[21]	Ray tracing algorithm with precomputed look-up tables	Geforce 8800 GT (336)	266 sampling rays with 12 quads in 1.37 sec
Colour reconstruction system with GPU (2009)[65]	1000-point based	Geforce GTX280 (622)	1400×1050 in 31 msec
Fast CGH using S-LUT (2009)[66]	Split look-up tables	GTX 285 (708.48)	700× faster than LUT on Intel Core i7 965 for object point larger than 40k
Ray-tracing (as the baseline reference) using GWO library[41]	Ray-tracing algorithm	GTX 260 (approx. 550)	48277 points 3D object in 1380 msec
Real-time CGH using multiple GPUs (2010)[67]	1000-point based	3×GTX 285 (708.48 per GPU)	1000 points per colour at 22 FPS
CGH with AMD (2010)[68]	1024-point based	AMD RV870 (unknown) comparing NVIDIA GTX 260 (approx. 550)	1920×1024 in 31 msec
GPU acceleration using SA (2010)[45]	Simulated annealing	NVIDIA GTX 260 (approx. 550)	Performance improvement of more an order of magnitude compared to using CPU only
Holographic optical tweezers algorithm implementation (2010)[69]	Superposition algorithm, weighted Gerchberg-Saxon algorithm	GTX 260 (approx. 550)	350× (SR) and 45× (GSW) faster than Intel Pentium D
Interpolated wavefront-recording plane (2011)[70]	Interpolated wavefront-recording plane (IWRP) approach	GTX 580 (1581.1)	2048×2048 in 25 msec
CWO++ library performance benchmark (2012)[30]	Gerchberg-Saxton algorithm	GTX 460M (518.4), GTX 295 (1 chip, approx 600), GTX 580 (1581.1)	2048×2048 Two magnitudes faster than an Intel Core i7 740QM
GPU cluster for divided CGH (2012)[44]	Optimized 2048-point based	12×GTX 480 (1344.96 per GPU)	6400×3072 in 55 msec
GPU cluster for distributed hologram computation (2013)[71]	Split look-up table	9×GTX 590 (2488.3 per GPU) and 14×Quadro 5000 (722.3 per GPU)	A computation cluster with 32.5 TFLOPS computing power
Binary detour-phase holograms (2013)[34]	Binary detour-phase method	NVIDIA TESLA C2050 (1030.4)	35×-53× speedup compated to AMD Phenom 9850 CPU
Localized error diffusion and redistribution (2014)[72]	Localized error diffusion and redistribution (LERDR) algorithm	GTX 590 (2488.3)	2048×2048 in 6 msec
3D binary CGH (2014, 2015)[73,74]	Precalculated triangular mesh	GTX 770 (3213.3)	Performance is better than point-based methods but slower than triangle-based algorithm
3D object tracking mask-based novel-look-up-table (2015)[75]	OTM-NLUT	3× GTX TITAN (4499.7)	31.1 FPS of Fresnel CGH patterns
Fourier hologram benchmarking (2015)[62]	Kinoform, Detour Phase, Lee and Burckhardt methods	NVIDIA TESLA C2050 (1030.4)	Speed-up of up to 68× compared to AMD Phenom 9850 CPU
Fast occlusion processing (2016)[76]	Point-source and wave-field hybrid	GTX 780Ti (5045.7)	1024 layers with 6.7 million points 21.28 msec
GPU for block-based parallel processing (2018)[77]	10K-point based	GTX 1080Ti (11339.7)	1024×1024 in 18.7 msec
Photorealistic CGH benchmark (2018)[43]	Backward ray-tracing and wavefront-recording planes (WRPs)	Quadro M5000 (4300.8)	1920×1080 in 20 msec
Real-time colour holographic reconstruction (2020)[78]	Point cloud based	13×GTX TITAN X (8000)	1920×1080 RGB + alpha coloured at 38.31 FPS

3.6 Reported work using GPUs

In 1995, Lucente and Galyean demonstrated the first published result of CGH generation using a computer graphics workstation.[79] The achieved performance was calculated using eight 128×60 pixels full-colour images, which lead to a replay field of a 3D object with different viewing angles. At that time, the calculation time of 2 seconds over the graphics workstation was 100 times faster than a conventional computer.

Later in 2003, Petz and Magnor used an NVIDIA Geforce 4600Ti to generate the interference fringes for holograms.[80] In their work, the authors assessed both the GPU performance and the computational time dependency based on the resolution of holograms. For an object that contains 1024 light source points, it takes 0.96s and 3.86s to calculate the corresponding holograms of the resolutions of 512×512 and 1024×1024, respectively.[80]

Before the introduction of CUDA, the graphical API OpenGL was used to compute holograms, as was reported in 2006[81] and 2009[21], however, the performance was not promising. Additionally, a real-time reconstruction system for an 800×600 64-point based 3D object CGH was reported in 2006 using HLSL and DirectX API, achieving a calculation speed 47× faster than a Pentium 4 CPU.[59]

The use of OpenCL for parallel computing to generate holograms with an AMD HD5000 was reported by Shimobaba *et al.* in 2010.[68]

Since the release of CUDA, there has been a surging interest in the generation of computer holograms utilizing the full credibility and computational power of GPUs.

The GPU microarchitectures have changed remarkably throughout the past decade, and the increased computational power produced an improvement of at least ten times. This performance improvement can also be seen in the reported literature.

Shiraki *et al.*[65] in 2009 reported a 1000 point light source (3D object) real-time holographic video generation system using an NVIDIA GTX 280 utilizing Tesla microarchitecture. The generated hologram resolution was 1400×1050 pixels. The performance was later surpassed by the introduction of GTX 1080Ti with Pascal microarchitecture in 2018.[77] The system reported by Kim *et al.* can produce real-time high definition (HD) holographic generation and projection using 10,000 points of light, a ten-times increment in terms of object-point counts than that of the system reported in.[65]

Table 3 provides a summary of some of the hardware implementations using different hologram generation algorithms in recent literature.

3.7 Summary of GPUs

Traditionally, GPU vendors design their line of products in order to carry out single precision floating point operations effectively.[82] Throughout the years, these vendors have worked to redesign their products to allow for the use of half-precision numbers and fixed points.

GPUs are by design powerful single and double precision floating point hardware accelerators, recent trends have led to the use of half-precision and fixed-point arithmetic, which further enhanced the computational speed while making a trade-off in terms of precision.

Due to the hardware and manufacturing constraints, the number of streaming (CUDA) cores that can be embedded within a single GPU is limited. Therefore, in order to speed up the hologram generation process, one practical solution is to form a GPU cluster using multiple GPUs. This can be done either in a single stand-alone system[67] or over a dedicated network.[44]

In general, GPU offers a strong candidate for CGH systems.

4 Field Programmable Gate Arrays (FPGAs)

Field programmable gate arrays (FPGAs) are highly-configurable integrated circuits capable of being reprogrammed by designers after manufacture. This degree of flexibility enables designers users to implement logical hardware designs during their product's development stage and to assess performance before the fabrication of expensive application specific integrated circuits (ASICs).

The three traditional vendors in this field have been Intel, Xilinx and Lattice. However, the growth of the market has seen additional vendors arise such as GOWIN Semiconductors. The cost for a single FPGA chip ranges from several dollars at the low-end to tens of thousands of dollars depending on the performance and hardware requirements as well as the market capability.

As shown in Figure 6, a typical FPGA architecture consists of the following five fundamental elements:[82,83]

1. **Functional unit**: A fundamental programmable cell that implements both combinational and sequential circuits. Depending on the vendors, these logic cells have been given different names, e.g., Intel names these cells as logic array blocks (LABs), whereas Xilinx calls them configuration logic blocks (CLBs).

2. **Interconnect fabric**: A mesh of programmable wires to establish the signal connections between functional units and inputs/outputs (I/Os).

3. **Configuration memory blocks**: A portion of on-board memory which stores the synthesized bitstream contents for the use of programming the functional units and fabrics.

4. **I/O interfaces**: General purpose inputs and outputs connect the signal from the integrated circuit to physical peripherals and I/O pins.

5. **Digital signal processing blocks**: Recent FPGAs incorporate dedicated 'hard' digital signal processing blocks that support various precisions, either fixed-point or floating-point, accumulations and multiplications to further boost the performance of FPGA-based implementations.

The implementation of a functional unit is vendor specific. The units in Xilinx and Intel FPGA products are SRAM-based, whereas those from the Lattice Semiconductors are based on EEP-

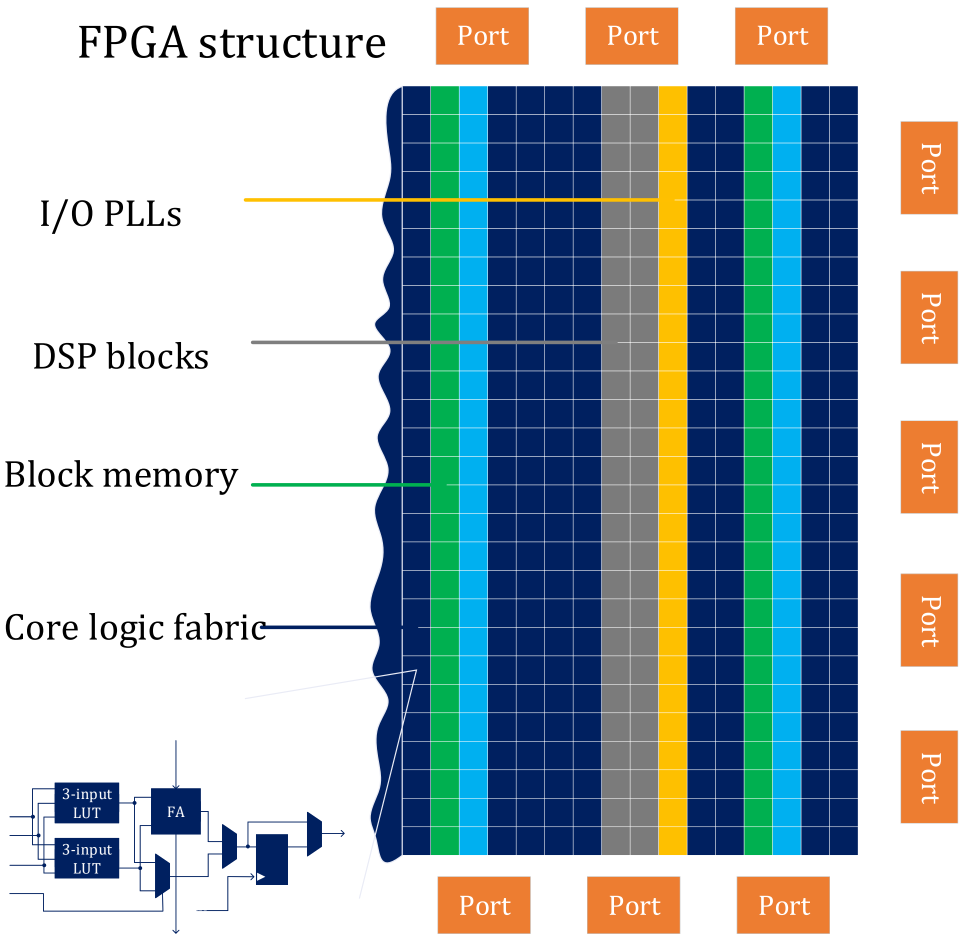

Fig 6 A typical architecture of FPGA, which consists of logic cells, I/O ports, DSP blocks, block memory, etc.

ROMs. Note that this can make it challenging to compare two FPGAs from different vendors; a fact that should be kept in mind when assessing FPGA performance.

Given their dominance of high-performance FPGA product families, we will mainly focus on FPGA products from Intel and Xilinx, and provide a comprehensive review based on their product families.

Both Intel and Xilinx provide intellectual property cores (IP-cores) that are programmable-hardware implementations of application specific peripherals and algorithms. These are optimized for a given product line and should be used where possible to expedite development time and boost performance.

4.1 The highly configurable hardware platform

The key strength of FPGAs is their highly configurable and hardware-programmable nature. The applications can be developed using computer-based hardware description languages (HDLs) such as Verilog/SystemVerilog, VHDL, etc.[83] These language-based designs are portable and usually independent of technology, with the exception of applying intellectual property (IP) cores and other chip-specific configurations. The designers are able to repeatedly programme and reconfigure a given chip to affect changes at the hardware level and reuse designs across different FPGA chips that are normally from the same vendor.

The ability for FPGAs to support HDLs provides a significant benefit in that almost all on-chip cells are highly configurable and can be used to synthesize any possible hardware implementations as long as the designs can potentially be fitted into the available logic cells and hardware units.

4.2 Implementations based on fixed and floating point

Since FPGA platforms are highly reconfigurable, the use of either fixed point or floating point arithmetic becomes one of the most important design considerations. According to a report produced by Xilinx,[82] FPGA applications will benefit from the conversion from floating point to fixed point arithmetic for certain applications requiring less power but higher speed.

Floating point precision, typically includes IEEE 754 half-precision (16-bit), single-precision (32-bit) and double-precision (64-bit) configurations, whereas fixed points are more flexible and usually range from several bits to 32-bit in width.

Devices such as GPUs, which are used in computationally heavy applications, have traditionally been designed architecturally so that they are more efficient when supporting floating point operations. When implemented on FPGAs at the hardware level, however, conventional floating point operations, e.g. based on GPUs or CPUs, are slower than fixed point alternatives. This is due to the need and difficulty which arises when controlling the mantissa and exponents of IEEE 754 floating points during the calculations.[84]

4.3 Available tools and utilities for FPGAs

FPGA vendors typically provide their own proprietary tools. Intel's Quartus Prime is widely used among the community to facilitate development for Intel-based FPGAs. As for those devices offered by Xilinx, there are development tools such as the Vivado design suite, which has replaced the Xilinx integrated synthesis environment (ISE).

ModelSim is a functional simulation software package from Mentor Graphics. It can be used independently to simulate hardware based on HDL descriptions, as well as being compatible with Intel Quartus Prime, Xilinx ISE and Vivado.

Table 4 Tools and utilities reported for FPGA implementations in recent years

Name	Category	Appearance	Year
Intel			
Quartus	Design tools	[85]	2001
		[86]	2002
		[87]	2011
		[88]	2012
		[89]	2015
Max+Plus II	Legacy design tool	[86]	2002
Xilinx			
Vivado design suite	Design tool	[90]	2019
ISE	Design tool	[91]	2011
DisplayPort IP	IP-core for DisplayPort	[90]	2019
MIG IP	IP-core for memory interface	[90]	2019
AXI interconnect IP	IP-core for AXI4	[90]	2019
Others			
ModelSim	Simulation tool	[87]	2011
		[88]	2012
		[89]	2015
Verilog	HDL language	[91]	2011
		[88]	2012
		[89]	2015
VHDL	HDL language	[92]	2010
		[87]	2011

In addition, many hardware implementations use intellectual-property (IP) cores provided by the vendors to perform certain operations on FPGAs.

Due to their unique nature, the FPGA development process is very distinct from traditional CPUs and GPUs. A simplified overview is summarized as follows:[83]

1. **Design specification and partition**: These two initial steps set up the entry point for the design.

2. **Simulation and functional verification**: This verification step tests the functionality of a compiled design using a user-specified testbench file.

3. **Design integration and verification**: This step integrates all partitioned modules into one large system.

4. **Pre-synthesis sign-off**: At this stage, all the known functional errors should have been eliminated.

5. **Synthesis and implementation**: Translates the hardware description language syntax and contents to an optimal Boolean description that maps the selected FPGA chip. The language synthesis tool will also remove redundant logic from the design if optimization is selected.

6. **Configuration bitstream download**: The development tool will map the synthesized HDL to the selected chip and configure the logic unit blocks.

7. **Prototype functional testing and verification**: At this stage, the design is tested on hardware to prove its functionality.

8. **Final sign-off**: All constraints should at this stage be satisfied and errors eliminated via hardware and simulation debugging before the final chip production.

4.4 The development time using FPGAs

Generally, depending on the level of hardware complexity and the use of IP-cores, the development time might vary. For example, reported by Takada *et al.*,[93] the group took over 3 years to develop and implement their algorithms into a custom-made bespoke FPGA platform consisting of 8 high-end FPGA chips.

The average development time for a project based on FPGA hardware implementations will typically be significantly longer than an equivalent CPU or GPU project. Although not being reported for CGH applications, a study conducted in 2012 estimated that developing algorithms on a GPU-based hardware platform for dense optical flow, stereo and local image extraction features takes approximately 2 months for one full-time post-doctoral employee whereas developing the same algorithms and functionalities over an FPGA platform will likely take 12 months for two post-doctoral employees[94]. Overall, the development time for FPGA-based applications are likely to take longer than the equivalent for an algorithm to be implemented on a GPU platform.

4.5 The advantages and disadvantages of using FPGAs

One of the merits of FPGA implementation is the potential to migrate a given FPGA register-transfer level (RTL) design into ASICs. ASICs are dedicated chipsets specifically designed for a certain application. They are inflexible and require significant one-off tooling costs, but once designed represent an optimal combination of performance, power and cost for a given hardware accelerator. The performance can be optimized for the generation of computer holograms with the use of ASIC technology. A recent work in 2017[95] demonstrated that an FPGA-based implementation can be migrated into a very-large-scale integration without the need for vast modifications.

The potential for high performance at moderate power consumption along with the ability to migrate a given design to an ASIC provides a strong argument for the use of FPGAs in CGH applications.

As pointed out in Section 4.4, the most significant drawback for FPGA implementation is the relatively long development time. FPGA-based hologram generation projects often require years of work by a group of researchers. Moreover, the required knowledge in terms of understanding of hardware architecture and FPGA technology for the developers sets up a high entry barrier.

Table 5 A summary of CGH implemented on FPGAs since 2008

Project and application (year)	Implemented algorithm	Hardware model	Pixel size and performance
HORN 5 2-dimensional FFT (2008)[96–99]	Phase computation by addition, point-cloud	4×Xilinx XC2VP70 and 1×XC2V1000	3D image with 10,000 points at 30 FPS
HORN 6 (2009)[100]	Phase computation by addition, point-cloud	A 16-board cluster each containing 4×Xilinx XC2VP70 and 1× XC2V1000	67.9 msec per hologram
Realtime hologram generation (2010)[92]	40,000 point light sources	Xilinx XC2VP70	1408×1050 in 9.3 msec
Cell-based hardware architecture (2011)[87]	Point light source	Altera (no specific model no.)	1408×1050 in 15.8 msec
One-step phase-retrieval (2011)[91]	OSPR	Xilinx Virtex-4 SX35	512×512 in 0.9 msec
Pixel-by-pixel hardware simulation(2012)[88]	Pixel by pixel and parallel schemes	Altera simulation	Performance not measured in physical hardware implementation
HORN 7 (2012, 2013)[101,102]	Phase computation by addition, point-cloud	Xilinx Virtex-6 ML605	2 million pixels of 16,000 points in 0.4 sec
Full analytical Fraunhofer CGH (2015)[89]	Polygon based	Altera Cyclone IV EP4CE115	800×600 in 9.6 msec
HORN 8 (2018)[50,93]	Amplitude modulation,[93] Phase modulation[50] point-cloud	7×Xilinx Virtex5 XC5VLX110 and 1×XC5VLX30T	An effective speed equivalent to 0.5 PFLOPS, 1920×1080 65000 points at 8.3 FPS
Clustered HORN 8 (2018)[103]	Spatiotemporal division point-cloud	8×HORN 8 boards	1920×1080 65000 points at 63 FPS
Single-chip video processor (2019)[90]	Layer based	Xilinx XCKU115	1920×1080 RGB at 16 FPS

4.6 Reported work using FPGAs

Table 5 summarizes recent implementations on FPGA platforms. Most of the hardware models used in the lines of work are high-end FPGAs from Xilinx.

HORN (HOlographic ReconstructioN) computers, which have been in active development by Ito *et al.* since 1992,[104] have provided the research community with many insights into the field of CGH hardware implementation, notably the use of FPGAs for real-time hologram generation. So far there are, in total, eight generations of devices being produced by this group, ranging from low-speed devices to high-speed special purpose computers. The first four generations of HORN use DSP or small-scale FPGA chips for real-time computation tasks.[86, 104–106] The later four generations of devices consist of large-scale FPGA chips embedded on delicate custom printed circuit boards (PCBs).[46, 50, 93, 96, 100] The latest product within this line of work is HORN-8, which comprises of seven powerful FPGA chips for calculation and one FPGA chip for communication. As reported in[93] and,[50] the HORN-8 special computer can generate a hologram for a 3D object of 10,000 points within 0.019 seconds with a peak performance of 0.5 tera floating point operations per second (TFLOPS) running at a 0.25 GHz clock cycle. At the time of writing, the team's outlook is to further develop an ASIC design based on the HORN-8 structure[107] .

Seo *et al.* proposed a hardware architecture based on pixel-by-pixel calculation scheme.[87,88,92,95] In this line of work, the authors efficiently reduced the number of memory accesses by utilizing the pixel-by-pixel method, which is different from the conventional light source-by-source calculation method. The authors also demonstrated a very-large-scale integration (VLSI) chip, based on the proposed FPGA architecture.[95] The work reported by Seo *et al.*[95] demonstrated that it is relatively simple to migrate an FPGA system into an ASIC design.

4.7 Summary of FPGAs

Benefiting from its highly configurable architecture, FPGAs are to date the most flexible hardware accelerators for use in hologram generation applications. The required calculations in hologram generation algorithms can take advantage of the high degree of parallelism within a FPGA chip. However, the development time to implement optimized algorithms on FPGAs are typically significant and require an advanced skillset of HDLs, digital logic design and hardware architecture; skills not typically present in traditional optics groups researching holography.

5 Review of other available hardware platforms

In parallel to researches on hardware implementations using hardware accelerators such as GPUs and FPGAs, there have been several attempts to implement holographic generation algorithms within other existing platforms. This section aims to review some of the candidates.

5.1 Digital Signal Processors (DSPs)

Digital signal processors (DSPs) are dedicated hardware platforms for signal processing applications. The microprocessors have architectures that are tuned for analogue and digital signal processing tasks with the ability to support single instruction multiple data (SIMD).

Nishikawa *et al.* reported the use of a DSP to generate holographic images in the late 1990s.[108] A multi-DSP system consisting of 3× 4 i860 DSPs was proposed to generate 3D objects for the application. The 3D object consists of 200 points and is 640 × 480 pixels in size. The multi-DSP system takes 68 seconds to generate the object as opposed to a reference workstation (SPARCstation 10) which generates the object in 291 seconds.

The most recent work was reported by Oi *et al.*[109] Twenty TMS320C6727 DSPs running floating point arithmetic was used to form the DSP block in the proposed system. These DSPs were dedicated to the conversion of integral photography (IP) images to holograms in the Fresnel diffraction domain. With a 1.5× redundancy design, an real-time performance of 50 FPS was achieved.

The current highest-end DSP products are those from Analog Devices and Texas Instruments. A TI TMS320C6678 eight-core floating-point DSP runs at a clock rate of 1 GHz to 1.4 GHz, with a maximum computational performance of 20 GFLOPS per core for single precision floating point operations.[110]

It is unlikely that these DSPs will be capable of performing complicated hologram generation algorithms due to the hardware specifications and limited computational power. However, it is still worthwhile to regard DSP as a valuable candidate to implement less complicated algorithms due to their low power profile and ease of programming. DSPs are typically programmed using

C language and assemblies. The toolchain support is considered mature and time-proven, further minimizing the development time and difficulty.[53]

5.2 Xeon Phi coprocessor and ClearSpeed accelerator board

Xeon Phi is a family of co-processors with x86 manycore architecture designed and produced by Intel.[111] It is to-date one of the few fairly powerful manycore co-processors that are intended for use in hardware acceleration applications.[112] This line of products supports the use of OpenMP.[113] As was introduced in Section 2.2, OpenMP is an API that is optimized for shared memory multi-processing programming and exploits multi-thread parallelism.

Murano *et al.* in 2014 reported on the use of a Xeon Phi coprocessor unit (Xeon Phi 5110P) for computer hologram generation.[114] The authors used the Intel MKL for the calculation of FFTs along with the OpenMP functionalities to make use of the available cores present in the coprocessor. Their results show that in all their test cases, GPU outperforms the Xeon Phi accelerator by a significant margin. However, when using Xeon Phi coprocessor, the amount of existing code that needs to be rewritten in order to port software-based algorithms into the hardware accelerator, as compared to that in the GPU case, can be minimized.

Another hardware acceleration board, ClearSpeed Advance Dual CSX600, was demonstrated in 2009.[115] The authors were able to speed up the point cloud hologram calculation 56× faster than an Intel Xeon CPU performing calculation in single core. Unfortunately, as of the writing of this survey, the production of ClearSpeed accelerator boards is no longer active.

5.3 System-on-Chip (SoC) hybrid CPU and FPGA

There is a growing need for hologram generation systems to become compact and low in power consumption. A trend towards System-on-Chip (SoC) utilizing the heterogeneous system architecture (HSA) has been rapidly growing over the years. The general idea behind SoC is to incorporate different devices and peripherals on a single chip to reduce the overall die area and to minimize the power consumption.[116] One hybrid product is to have both FPGA and microprocessors or CPUs on board one chip. A further discussion is present in Section 6.5.

In one of the most recent studies conducted by Yamamoto *et al.*, the authors developed a compact holographic computer using a Xilinx Zynq UltraScale+ MPSoC consisting of an ARM CPU and an FPGA on one single chip.[117] The reported system was able to reproduce 1920×1080 pixels 3D video at a rate of 15 frames per second.[117] They also compared the result to the performance of a Jetson TX1 platform,[118] the calculation time of 1920×1080 pixels with 6500 points on the SoC platform took 0.066s, whereas the Jetson TX1 took 1.294s.

The development time for these SoC hardware implementations would be even longer than that of pure FPGA developments since the incorporation of both CPU, which requires multi-thread programming, and FPGA, which uses hardware description languages, adds another level of complexity when highly optimized codes and algorithms are needed. However, the power efficiency, die area and package size scale-down can bring about other benefits that mitigate for the increased programming workload.

6 Discussion

As shown in Fig 7, most of the reported work included in this survey implemented algorithms using GPU platforms, totaling 24 papers, as compared to other accelerator platforms between 2008 and

Table 6 General comparison between CPU, GPU, FPGA, DSP and other platforms

Platform	Number of cores	Serial or parallel	Clock frequency	Development time	Power efficiency	Portability
CPU	Low	Mainly serial	High	Short	Average	Straightforward
GPU	High	Parallel	High	Average	Low	Less challenging
FPGA	High	Parallel	Low	Long	High (less power consumption, depending on implementation)	Difficult when vendor-specific IP-cores are used
DSP	Low	Mainly serial	Average to High	Average	Average	Simple (from low- to higher-performance)
Xeon Phi / ClearSpeed	Average	Serial with many-core parallel	Average	Short	Average	Average
Heterogeneous SoC e.g. FPGA + CPU	High	Serial and parallel	Low	Long	High	Platform dependent

2020. FPGA-based systems are popular as well, reaching up to 16 published papers. In particular the line of work exemplified by the HORN group exploits the potential of FPGA parallelism for fast hologram generation.

There are also a number of research papers implementing algorithms with CPUs only, however, as discussed in Section 2.1, most of them tend to focus on the development of novel algorithms and choose a PC platform without hardware accelerators as a means of algorithm evaluation and verification.

It is worth noting that cross-platform comparisons based solely on the calculation speed are not strictly reasonable. This is because different platforms incorporate different architectures and have different toolchain supports. Essentially, the algorithms implemented despite best efforts can still be fundamentally different across multiple platforms. Therefore, it is of great essence that analytical models with key performance metrics, which consider not only FPS and power efficiency but also other factors, be proposed to assess performances over different hardware.

We summarize the hardware specifications for the reviewed hardware and provide a general comparison between these platforms in Table 6. The table shows the difference in terms of the number of cores, serial or parallel architectures, clock frequencies, the estimated development times, power efficiencies and the software portability.

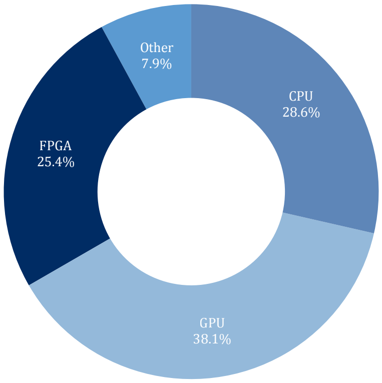

Fig 7 A comparison of CPU, GPU, FPGA and other hardware implementations in recent existing literature (2008-2020).

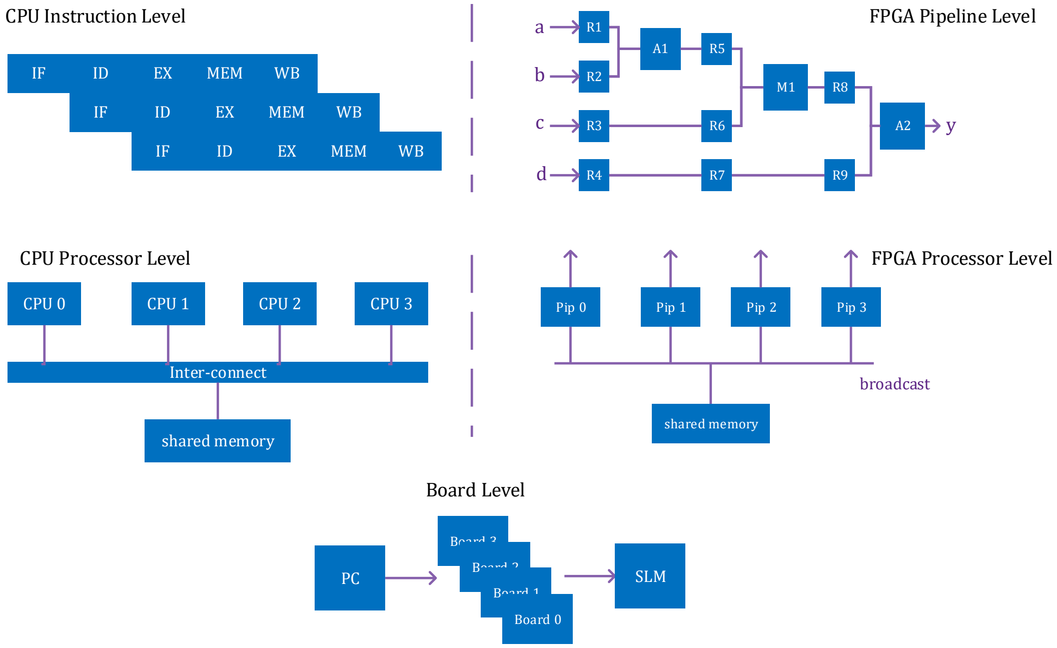

Fig 8 Different levels of parallelism and concurrency on hardware platforms. The left hand side depicts the CPU instruction pipeline and processor-level parallelization, whereas the right hand side shows the FPGA parallelization at the equivalent levels. Hardware clustering at the board level further exploits parallelism.

We conclude the six key considerations when selecting a suitable hardware platform for CGH related implementations:

1. Hardware manufacturing constraints.
2. Toolchain support.
3. Fixed point or floating point arithmetic.
4. Parallel and sequential processing – shown in Figure 8.
5. Development time.
6. Portability of software.

6.1 Toolchain support

One of the most important aspects to consider is the full-cycle development toolchain support. CPUs and GPUs platforms are likely to be less affected by the lack of available software package and library supports, as discussed in the previous sections. However, FPGAs and other accelerators such as DSPs and co-processors might suffer from the lack of active development support and will, in turn, affect the overall development process. In general, the availability of tools and utilities to support the dedicated hardware creates a resource barrier towards the successful implementation.

6.2 Choice of algorithms and parallel/sequential processing

Many algorithms exist for 2D/3D hologram generation. Different algorithms would require different hardware resources in practice, e.g. triangular-mesh based algorithms can take the advantage of being compatible with modern computer graphics technologies utilizing polygon meshes for object computations[13]. Regardless of the algorithm used the size, e.g. hologram resolution size, number of points/polygons, is an important consideration in all cases, and more importantly, increasingly complex holograms demand larger and better hardware.

GPUs and other specialized hardware accelerators are useful to the speed enhancement of hologram calculation by utilizing parallelism and optimizing for sequential processing. For example, in the point-cloud-based calculation, the hologram patterns are calculated using the same mathematical formula, and more importantly, the calculation of these patterns for each object point is independent of other object points.[13] The independent calculation of object points can potentially make use of the parallel processing for hardware platforms.

Moreover, for CGH algorithms involving FFT operations and depending on the hardware utilized, the FFT operations can be parallelized through different cores or pipelines at the processor level, as shown in Figure 8.

It is also of great importance, though being algorithm-dependent, to be aware of the number of sequential processes required and optimize for performance while exploiting concurrency and parallelism within the specified hardware. For example, iterative algorithms such as the Gerchberg-Saxton (GS) algorithm[119] requires sequential processing that cannot or tend to be difficult to parallelize and multi-task. It is then of the developer's responsibility to select a platform that does not only fulfill the need for parallelism but also have the options to optimize for the sequential operations when implementing the desired algorithm.

6.3 Portability of software

It is essential to consider the possibility of transferring the developed software and firmware from one system to another while keeping in mind the trade-offs between portability and performance.

This transfer would likely be required when upgrades toward newer generations of hardware are expected, or performance comparisons between different devices are needed.

The most straightforward transfer comes when the CPU platform, which is usually based on a PC, is used. Upgrading between different operating systems and software platforms are comparatively simple thanks to the abundant software support. In comparison, porting from one NVIDIA GPU to another would sometimes require more work, although CUDA provides a unified development environment. This is mainly due to the upgrades in hardware between different generations of GPU products. As for intra-generation code transplant, it is usually not challenging, as long as the memory and computational power limitations have been taken into account by the developer.

Code transfer among different FPGA platforms, on the contrary, would be slightly difficult, especially when target chip IP-cores are used for the application. With the above noted, transferring from a lower performance FPGA to an FPGA with higher performance can be relatively simple, this will likely be the case when HDL descriptions are used.

6.4 Hologram generation quality assessment

An end-to-end CGH hardware implementation assessment should include fast generation, hardware performance and generated quality assessment of the holograms.

There currently is a lack of available unified criterion to assess the quality of computer holograms generated from different platforms. Kim *et al.*[90] uses a modulation transfer function (MTF) to compare the image quality of different holograms. Structural similarities (SSIM) has also been used in work[49] to evaluate the quality of the generated images. Another widely used metrics are to measure the mean square error (MSE) and peak signal-to-noise ratio (PSNR).[120] Blinder *et al.*[121] provided a more detailed review of the quality assessment for computer generated holograms.

6.5 Heterogeneous computing and its related hardware

There is a growing trend in the embedded systems, image and video processing communities to incorporate the state-of-the-art heterogeneous computing systems into their applications.

Heterogeneous computing systems typically refer to systems that fuse more than one type of processors or cores together,[122] it could also refer to systems that combine a large number of processor cores with the same ISA,[53] e.g. Intel Xeon Phi, or a small number of cores with different execution performances, e.g. ARM big.Little platform.[123] In this section, we focus mainly on the development and trend in heterogeneous hardware accelerators that incorporate different types of instruction set architecture (ISA) devices.

These hardware systems take advantage of conventional multi-core hardware accelerators while in the meantime bypass some of the limitations and disadvantages of using a single hardware accelerator architecture.[116] The aims of having the heterogeneous system architecture (HSA) are to reduce the communication latency between different computing devices and to improve the parallel execution performance.[116]

The level of heterogeneity in a computing system gradually increases, with more and more SoC platforms being produced. Among various of heterogeneous hardware platforms, the combination of CPUs and FPGAs, usually in the form of hard ARM processors embedded in an FPGA, as well as CPUs with DSPs, are potentially good candidates for low-cost low-power hologram generation platforms due to their inherent merits that balance the pros and cons of conventional system architectures.

Another worth mentioning heterogeneous computing platform is the Jetson module. Only the Jetson TX1[118] was evaluated in work.[117] Its upgraded version TX2[124] and the most recent AGX

Table 7 NVIDIA Jetson module products family

Model (year of launch)	GPU	Computational power	Power (watt)
TX1 (2015)[118]	Maxwell	Over 1 Tera-FLOPS	Under 10
TX2 series (2017)[124]	Pascal	1.3 TFLOPS	7.5-20
AGX XAVIER series (2018)[125]	Volta with Tensor Cores	20-32 Tera-operations per second (TOPS)	10-30
Nano (2019)[126]	Maxwell	472 GFLOPS	5-10
Xavier NX (2019/2020)[127]	Volta with Tensor Cores	21 TOPS	10-15

Xavier,[125] both with boosted performance and power efficiency, are also of implementation interest. A low-cost variant of the Jetson family, Jetson NANO, has also become available in the market recently.[126] A list of the Jetson family is shown in Table 7.

6.6 Future trend in embedded systems and high performance hardware platforms

ARM developed the Neon technology for their Cortex-A series and R52 processors as an advanced SIMD architecture extension for image and video as well as general signal processing purposes.[128] No reported work to-date has exploited the possibility of integrating an ARM-based SoC embedded platform for the generation of computer holograms while utilizing technologies such as Neon.

NVIDIA recently announced their plan to bring CUDA acceleration to the ARM ecosystem.[129] This will potentially bring the power and accessibility of CUDA to platforms such as ARM-based SoCs. This is also accompanied by the introduction of CUDA-X high performance computing (HPC) libraries which can potentially further exploit parallelism and provide improved processing performance.[130]

From another perspective, the recently announced Vitis Unified Software Platform from Xilinx provides another degree of flexibility to use high-level synthesis (HLS) FPGA languages in order to help reduce the development overhead of FPGA applications[131]. This unified platform is envisioned to shorten the overall development time with higher level implementations without the need of incorporating fully RTL-level development and to provide a better programmability for the

FPGA hardware.

7 Future work and conclusions

CGH calculations primarily require a high degree of computation parallelism, thus embracing the use of hardware accelerators such as GPUs, FPGAs, etc., for the realization of real-time computer generated holograms.

It is anticipated that there will be two separate research paths that lead towards the future of CGH hardware implementations, including:

1. **High performance hardware platforms for real-time CGH generations and displays.** These systems will usually be of high costs and require a long development cycle. Algorithms for future fast computer hologram generations will likely be developed using these hardware platforms for first-phase verifications and optimizations. A good example is the HORN-8 special purpose computer.[50,93,103] The team has recently announced their future outlook to build ASIC devices to further boost the performance.[107]

2. **Embedded computers and systems for low-power and low-cost applications.** In order for this ultimate display technology to become reachable to ordinary households and individual consumers, a reduction in cost and a significant reduction in volumes and sizes are essential. A large amount of work can potentially be done on SoC platforms, e.g. CPU-FPGA, CPU-DSP devices, as well as on supercomputer-on-a-module embedded computing devices.[132] Examples that demonstrate the implementations for embedded systems are those from Kim *et al.*[90] and Yamamoto *et al.*[117]

In this review paper, we have attempted to provide a useful review on the hardware implementations on CGH, as well as to provide practical information about the current state-of-the-art hardware platforms that can be selected by researchers and developers to implement computer hologram generation algorithms for their specific applications.

A key insight from this review is that the potential for real-time holography exists today without the need for bespoke hardware. A flagship GPU can process an entire holographic frame in 20ms, providing high-quality CGH in real-time. We predict holography transitioning towards mobile and embedded platforms, a trend evidenced by extrapolating from the growth of GPU computational power in Table 7. Bespoke hardware accelerators, such as FPGAs and ASICs, shall continue to advance the field of CGH hardware in this period, pushing the boundaries on what is achievable in terms of computation power, energy consumption and overall system cost.

References

1 J. W. Goodman, "Introduction to Fourier optics," *Introduction to Fourier optics, 3rd ed., by JW Goodman. Englewood, CO: Roberts & Co. Publishers, 2005* **1** (2005).

2 D. Gabor, "A new microcopic principle," *Nature* **161**(4098), 777–778 (1948).

3 B. R. Brown and A. W. Lohmann, "Complex Spatial Filtering with Binary Masks," *Applied Optics* **5**(6), 967–969 (1966).

4 M. Lucente, "The First 20 Years of Holographic Video – and the Next 20," in *SMPTE 2nd Annual International Conference on Stereoscopic 3D for Media and Entertainment*, 1–16, SMPTE (2011).

5 A. Maimone, A. Georgiou, and J. S. Kollin, "Holographic near-eye displays for virtual and augmented reality," *ACM Transactions on Graphics* **36**(4), 1–16 (2017).

6 B. Robertson, H. Yang, M. M. Redmond, *et al.*, “Demonstration of multi-casting in a 1 × 9 LCOS wavelength selective switch,” *Journal of Lightwave Technology* **32**(3), 402–410 (2014).

7 R. W. Bowman, G. M. Gibson, A. Linnenberger, *et al.*, “Red tweezers: Fast, customisable hologram generation for optical tweezers,” *Computer Physics Communications* **185**(1), 268–273 (2014).

8 W. A. Crossland, T. D. Wilkinson, I. G. Manolis, *et al.*, “Telecommunications applications of LCOS devices,” *Molecular Crystals and Liquid Crystals Science and Technology Section A: Molecular Crystals and Liquid Crystals* **375**, 1–13 (2002).

9 F. Yaras, H. Kang, and L. Onural, “State of the art in holographic displays: A survey,” *IEEE/OSA Journal of Display Technology* **6**(10), 443–454 (2010).

10 G. Nehmetallah and P. P. Banerjee, “Applications of digital and analog holography in three-dimensional imaging,” *Advances in Optics and Photonics* **4**(4), 472 (2013).

11 J. Liu, J. Jia, Y. Pan, *et al.*, “Overview of fast algorithm in 3D dynamic holographic display,” *International Symposium on Photoelectronic Detection and Imaging 2013: Optical Storage and Display Technology* **8913**(August 2013), 89130X (2013).

12 T. Nishitsuji, T. Shimobaba, T. Kakue, *et al.*, “Review of Fast Calculation Techniques for Computer-Generated Holograms with the Point-Light-Source-Based Model,” *IEEE Transactions on Industrial Informatics* **13**(5), 2447–2454 (2017).

13 J. H. Park, “Recent progress in computer-generated holography for three-dimensional scenes,” *Journal of Information Display* **18**(1), 1–12 (2017).

14 P. W. M. Tsang, T.-C. Poon, and Y. M. Wu, "Review of fast methods for point-based computer-generated holography [Invited]," *Photonics Research* **6**(9), 837 (2018).

15 T. Shimobaba, T. Kakue, and T. Ito, "Review of Fast Algorithms and Hardware Implementations on Computer Holography," *IEEE Transactions on Industrial Informatics* **12**(4), 1611–1622 (2016).

16 T. Shimobaba and T. Ito, *Computer Holography Acceleration Algorithms And Hardware Implementations*, CRC Press, 1st ed. (2019).

17 D. Gabor, W. E. Kock, and W. S. George, "Holography," *Science* **173**(3991), 11–24 (1971).

18 S. Reichelt, R. Häussler, G. Fütterer, *et al.*, "Full-range, complex spatial light modulator for real-time holography," *Optics Letters* **37**(11), 1955 (2012).

19 A. J. Macfaden and T. D. Wilkinson, "Characterization, design, and optimization of a two-pass twisted nematic liquid crystal spatial light modulator system for arbitrary complex modulation," *Journal of the Optical Society of America A* **34**(2), 161 (2017).

20 T. Kurihara and Y. Takaki, "Shading of a computer-generated hologram by zone plate modulation," *Optics Express* **20**(4), 3529 (2012).

21 R. H.-Y. Chen and T. D. Wilkinson, "Computer generated hologram with geometric occlusion using GPU-accelerated depth buffer rasterization for three-dimensional display," *Applied Optics* **48**(21), 4246 (2009).

22 S. Liu, H. Wei, N. Li, *et al.*, "Occlusion calculation algorithm for computer generated hologram based on ray tracing," *Optics Communications* **443**(March), 76–85 (2019).

23 J. Xiao, J. Liu, Z. Lv, *et al.*, "On-axis near-eye display system based on directional scattering holographic waveguide and curved goggle," *Optics Express* **27**(2), 1683 (2019).

24 Y. Nagahama, T. Shimobaba, T. Kakue, *et al.*, “Image quality improvement of random phase-free holograms by addressing the cause of ringing artifacts,” *Applied Optics* **58**(9), 2146 (2019).

25 G. Li, J. Jeong, D. Lee, *et al.*, “Space bandwidth product enhancement of holographic display using high-order diffraction guided by holographic optical element,” *Optics Express* **23**(26), 33170 (2015).

26 E. Buckley, “Computer-Generated Phase-Only Holograms for Real-Time Image Display,” *Advanced Holography - Metrology and Imaging* (2011).

27 J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th ed. (2011).

28 M. Nemirovsky and D. Tullsen, *Multithreading Architecture*, Synthesis lectures in computer architecture, Morgan & Claypool Publishers (2013).

29 OpenMP Architecture Review Board, “OpenMP FAQ,” (2018).

30 T. Shimobaba, J. Weng, T. Sakurai, *et al.*, “Computational wave optics library for C++: CWO++ library,” *Computer Physics Communications* **183**(5), 1124–1138 (2012).

31 S.-C. Kim and E.-S. Kim, “Effective generation of digital holograms of three-dimensional objects using a novel look-up table method,” *Applied Optics* **47**(19), D55 (2008).

32 S.-C. Kim and E.-S. Kim, “Fast computation of hologram patterns of a 3D object using run-length encoding and novel look-up table methods,” *Applied Optics* **48**(6), 1030 (2009).

33 J. Jia, Y. Wang, J. Liu, *et al.*, “Reducing the memory usage for effectivecomputer-generated hologram calculation using compressed look-up table in full-color holographic display,” *Applied Optics* **52**(7), 1404 (2013).

34 G. Makey, M. S. El-Daher, and K. Al-Shufi, "Accelerating the calculations of binary detour phase method by integrating both CUDA and Matlab programming for GPU's parallel computations," *Optik* **124**(22), 5486–5488 (2013).

35 P. Memmolo, L. Miccio, F. Merola, *et al.*, "Investigation on specific solutions of Gerchberg-Saxton algorithm," *Optics and Lasers in Engineering* **52**(1), 206–211 (2014).

36 Z. Wang, G. Lv, Q. Feng, *et al.*, "Highly efficient calculation method for computer-generated holographic stereogram using a lookup table," *Applied Optics* **58**(5), A41–A47 (2019).

37 A. T. S. Ymeonidou, R. A. M. Uhamad, K. Izhakkumkara, *et al.*, "Efficient holographic video generation based on rotational transformation of wavefields," *Optics Express* **27**(26), 37383–37399 (2019).

38 T. Shimobaba and T. Ito, "Fast generation of computer-generated holograms using wavelet shrinkage," *Optics Express* **25**(1), 77 (2017).

39 T. Shimobaba, K. Matsushima, T. Takahashi, *et al.*, "Fast, large-scale hologram calculation in wavelet domain," *Optics Communications* **412**(August 2017), 80–84 (2018).

40 T. Shimobaba, S. Yamada, T. Kakue, *et al.*, "Fast hologram calculation using wavelet transform," in *Proc. SPIE 10964, Tenth International Conference on Information Optics and Photonics*, **10964**, 116, SPIE (2018).

41 T. Shimobaba, N. Masuda, and T. Ito, "Simple and fast calculation algorithm for computer-generated hologram with wavefront recording plane," *Optics Letters* **34**(20), 3133 (2009).

42 K. Matsushima and S. Nakahara, "Extremely high-definition full-parallax computer-generated hologram created by the polygon-based method," *Applied Optics* **48**(34), H54–H63 (2009).

43 Y. Wang, X. Sang, Z. Chen, *et al.*, “Real-time photorealistic computer-generated holograms based on backward ray tracing and wavefront recording planes,” *Optics Communications* **429**(July), 12–17 (2018).

44 N. Takada, T. Shimobaba, H. Nakayama, *et al.*, “Fast high-resolution computer-generated hologram computation using multiple graphics processing unit cluster system,” *Applied Optics* **51**(30), 7303 (2012).

45 J. Carpenter and T. D. Wilkinson, “Graphics processing unit–accelerated holography by simulated annealing,” *Optical Engineering* **49**(9), 095801 (2010).

46 S. Yamada, T. Shimobaba, T. Kakue, *et al.*, “Full-color computer-generated hologram using wavelet transform and color space conversion,” *Optics Express* **27**(6), 8153 (2019).

47 P. Pozzi, L. Maddalena, N. Ceffa, *et al.*, “Fast Calculation of Computer Generated Holograms for 3D Photostimulation through Compressive-Sensing Gerchberg–Saxton Algorithm,” *Methods and Protocols* **2**(1), 1–11 (2019).

48 Y. Nagahama, T. Shimobaba, T. Kawashima, *et al.*, “Holographic multi-projection using the random phase-free method,” *Applied Optics* **55**(5), 1118 (2016).

49 D. Arai, T. Shimobaba, T. Nishitsuji, *et al.*, “An accelerated hologram calculation using the wavefront recording plane method and wavelet transform,” *Optics Communications* **393**(February), 107–112 (2017).

50 T. A. A. Kamatsu, R. Y. H. Irayama, H. Irotaka, *et al.*, “Special-purpose computer HORN-8 for phase-type electro-holography,” *Optics Express* **26**(20), 26722–26733 (2018).

51 NVIDIA, “NVIDIA Launches the World’s First Graphics Processing Unit: GeForce 256,” (1999).

52 NVIDIA, “CUDA C Programming Guide,” (2019).

53 A. HajiRassouliha, A. J. Taberner, M. P. Nash, *et al.*, “Suitability of recent hardware accelerators (DSPs, FPGAs, and GPUs) for computer vision and image processing algorithms,” *Signal Processing: Image Communication* **68**(July), 101–119 (2018).

54 Intel, “Intel Xeon Processor E7 Family,” (2019).

55 NVIDIA, “NVIDIA Geforce introduction,” (2019).

56 Wikipedia, “List of Nvidia graphics processing units,” (2019).

57 P. N. Glaskowsky, “NVIDIA ’ s Fermi : The First Complete GPU Computing Architecture [white paper],” Tech. Rep. September, NVIDIA (2009).

58 NVIDIA, “NVIDIA Turing GPU [white paper],” tech. rep., NVIDIA (2018).

59 T. Ito, T. Tanaka, T. Sugie, *et al.*, “Computer generated holography using a graphics processing unit,” *Optics Express* **14**(2), 603 (2006).

60 NVIDIA, “CuFFT Library User’s Guide,” Tech. Rep. May, NVIDIA (2019).

61 S. Memeti, L. Li, S. Pllana, *et al.*, “Benchmarking OpenCL, OpenACC, OpenMP, and CUDA,” in *2017 Workshop on Adaptive Resource Management and Scheduling for Cloud Computing*, 1–6, ACM (2017).

62 G. Makey, M. S. El-Daher, and K. Al-Shufi, “Modification of common Fourier computer generated hologram’s representation methods from sequential to parallel computing,” *Optik* **126**(11-12), 1067–1071 (2015).

63 A. Hermerschmidt, S. Krüger, T. Haist, *et al.*, “Holographic optical tweezers with real-time hologram calculation using a phase-only modulating LCOS-based SLM at 1064 nm,” *Complex Light and Optical Forces II* **6905**(January 2008), 690508 (2008).

64 R. H.-Y. Chen and T. D. Wilkinson, “Computer generated hologram from point cloud using graphics processor,” *Applied Optics* **48**(36), 6841 (2009).

65 A. Shiraki, N. Takada, M. Niwa, *et al.*, “Simplified electroholographic color reconstruction system using graphics processing unit and liquid crystal display projector,” *Optics Express* **17**(18), 16038–16045 (2009).

66 Y. Pan, X. Xu, S. Solanki, *et al.*, “Fast CGH computation using S-LUT on GPU,” *Optics Express* **17**(21), 18543 (2009).

67 H. Nakayama, N. Takada, Y. Ichihashi, *et al.*, “Real-time color electroholography using multiple graphics processing units and multiple high-definition liquid-crystal display panels,” *Applied Optics* **49**(31), 5993 (2010).

68 T. Shimobaba, T. Ito, N. Masuda, *et al.*, “Fast calculation of computer-generated-hologram on AMD HD5000 series GPU and OpenCL.” An optional note (2010).

69 S. Bianchi and R. Di Leonardo, “Real-time optical micro-manipulation using optimized holograms generated on the GPU,” *Computer Physics Communications* **181**(8), 1444–1448 (2010).

70 P. Tsang, W.-K. Cheung, T.-C. Poon, *et al.*, “Holographic video at 40 frames per second for 4-million object points,” *Optics Express* **19**(16), 15205 (2011).

71 Y. Pan, X. Xu, and X. Liang, “Fast distributed large-pixel-count hologram computation using a GPU cluster,” *Applied Optics* **52**(26), 6562 (2013).

72 P. W. M. Tsang, A. S. M. Jiao, and T.-C. Poon, “Fast conversion of digital Fresnel hologram to phase-only hologram based on localized error diffusion and redistribution,” *Optics Express* **22**(5), 5060 (2014).

73 F. Yang, A. Kaczorowski, and T. D. Wilkinson, “Fast precalculated triangular mesh algorithm for 3D binary computer-generated holograms,” *Applied Optics* **53**(35), 8261 (2014).

74 F. Yang, A. Kaczorowski, and T. D. Wilkinson, “Enhancing the quality of reconstructed 3D objects by using point clusters,” *Applied Optics* **54**(18), 5726 (2015).

75 M.-W. Kwon, S.-C. Kim, S.-E. Yoon, *et al.*, “Object tracking mask-based NLUT on GPUs for real-time generation of holographic videos of three-dimensional scenes,” *Optics Express* **23**(3), 2101 (2015).

76 A. N. G. Illes, P. A. G. Ioia, R. É. M. I. Cozot, *et al.*, “Hybrid approach for fast occlusion processing in computer-generated hologram calculation,” *Applied Optics* **55**(20), 5459–5470 (2016).

77 D.-W. Kim, Y.-H. Lee, and Y.-H. Seo, “High-speed computer-generated hologram based on resource optimization for block-based parallel processing,” *Applied Optics* **57**(16), 4569 (2018).

78 S. Ikawa, N. Takada, H. Araki, *et al.*, “Real-time color holographic video reconstruction using multiple-graphics processing unit cluster acceleration and three spatial light modulators,” *Chinese Optics Letters* **18**(1), 1–5 (2020).

79 M. Lucente and T. A. Galyean, “Rendering interactive holographic images,” in *SIGGRAPH ’95 Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, 387–394, SIGGRAPH (1995).

80 C. Petz and M. Magnor, “Fast hologram synthesis for 3D geometry models using graphics hardware,” *Practical Holography XVII and Holographic Materials IX* **5005**(June 2003), 266 (2003).

81 L. Ahrenberg and J. Watson, “Computer generated holography using parallel commodity graphics hardware,” *Optics Express* **5664**(17), 603–608 (2006).

82 Xilinx, “Reduce Power and Cost by Converting from Floating Point to Fixed Point Introduction,” *White Paper: Floating vs Fixed Point* , 1–14 (2017).

83 M. D. Ciletti, *Advanced Digital Design with the Verilog HDL*, Prentice Hall Press, Upper Saddle River, NJ, USA, 2nd ed. (2010).

84 R. Solovyev, A. Kustov, V. Rukhlov, *et al.*, “Fixed-Point Convolutional Neural Network for Real- Time Video Processing in FPGA,” in *2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, 1605–1611, IEEE (2019).

85 T. Shimobaba and T. Ito, “An efficient computational method suitable for hardware of computer-generated hologram with phase computation by addition,” *Computer Physics Communications* **138**(1), 44–52 (2001).

86 T. Shimobaba, S. Hishinuma, and T. Ito, “Special-purpose computer for holography HORN-4 with recurrence algorithm,” *Computer Physics Communications* **148**(2), 160–170 (2002).

87 Y.-H. Seo, H.-J. Choi, J.-S. Yoo, *et al.*, “Cell-based hardware architecture for full-parallel generation algorithm of digital holograms,” *Optics Express* **19**(9), 8750 (2011).

88 Y.-H. Seo, Y.-H. Lee, J.-S. Yoo, *et al.*, “Hardware architecture of high-performance digital hologram generator on the basis of a pixel-by-pixel calculation scheme,” *Applied Optics* **51**(18), 4003–4012 (2012).

89 Z.-Y. Pang, Z.-X. Xu, Y. Xiong, *et al.*, “Hardware architecture for full analytical Fraunhofer computer-generated holograms,” *Optical Engineering* **54**(9), 095101 (2015).

90 H. Kim, Y. Kim, H. Ji, *et al.*, “A single-chip FPGA holographic video processor,” *IEEE Transactions on Industrial Electronics* **66**(3), 2066–2073 (2019).

91 E. Buckley, “Real-time error diffusion for signal-to-noise ratio improvement in a holographic projection system,” *IEEE/OSA Journal of Display Technology* **7**(2), 70–76 (2011).

92 Y. H. Seo, H. J. Choi, J. S. Yoo, *et al.*, “An architecture of a high-speed digital hologram generator based on FPGA,” *Journal of Systems Architecture* **56**(1), 27–37 (2010).

93 T. Sugie, T. Akamatsu, T. Nishitsuji, *et al.*, “High-performance parallel computing for next-generation holographic imaging,” *Nature Electronics* **1**(4), 254–259 (2018).

94 K. Pauwels, M. Tomasi, J. Díaz Alonso, *et al.*, “A Comparison of FPGA and GPU for real-time phase-based optical flow, stereo, and local image features,” *IEEE Transactions on Computers* **61**(7), 999–1012 (2012).

95 Y.-H. Seo, Y.-H. Lee, and D.-W. Kim, “ASIC chipset design to generate block-based complex holographic video,” *Applied Optics* **56**(9), D52 (2017).

96 T. Ito, N. Masuda, K. Yoshimura, *et al.*, “Special-purpose computer HORN-5 for a real-time electroholography,” *Optics Express* **13**(6), 1923–1932 (2005).

97 Y. Abe, N. Masuda, H. Wakabayashi, *et al.*, “Special purpose computer system for flow visualization using holography technology,” *Optics Express* **16**(11), 587–592 (2008).

98 S.-i. Satake, Y. Hiroi, Y. Suzuki, *et al.*, “Special-purpose computer for two-dimensional FFT,” *Computer Physics Communications* **179**(6), 404–408 (2008).

99 Y. Ichihashi, N. Masuda, M. Tsuge, *et al.*, “One-unit system to reconstruct a 3-D movie at a video-rate via electroholography,” *Optics Express* **17**(22), 19691 (2009).

100 A. Shiraki, Y. Ichihashi, H. Nakayama, *et al.*, “HORN-6 special-purpose clustered computing system for electroholography,” *Optics Express* **17**(16), 13895 (2009).

101 N. Okada, D. Hirai, Y. Ichihashi, *et al.*, “Special-Purpose Computer HORN-7 with FPGA Technology for Phase Modulation Type Electro-Holography,” in *19th International Display Workshops 2012 (IDW/AD’12)*, 1284–1287, Society for Information Display (2012).

102 N. Masuda, E. Yutaka, K. Takashi, *et al.*, “Special Purpose Computer for Phase Modulation Type Electro-Holography with DVI output [in Japanese],” in *International Conference on 3D Systems and Applications*, 373–374 (2013).

103 Y. Yamamoto, H. Nakayama, N. Takada, *et al.*, “Large-scale electroholography by HORN-8 from a point-cloud model with 400,000 points,” *Optics Express* **26**(26), 34259 (2018).

104 T. Ito, T. Yabe, M. Okazaki, *et al.*, “Special-purpose computer HORN-1 for reconstruction of virtual image in three dimensions,” *Computer Physics Communications* **82**(2-3), 104–110 (1994).

105 T. Ito, H. Eldeib, K. Yoshida, *et al.*, “Special-purpose computer for holography HORN-2,” *Computer Physics Communications* **93**(1), 13–20 (1996).

106 T. Shimobaba, N. Masuda, T. Sugie, *et al.*, “Special-purpose computer for holography HORN-3 with PLD technology,” *Computer Physics Communications* **130**(1), 75–82 (2000).

107 T. Nishitsuji, Y. Yamamoto, T. Sugie, *et al.*, “Dedicated computer for computer holography and its future outlook,” in *Proceedings Volume 10997, Three-Dimensional Imaging, Visualization, and Display 2019*, 16, SPIE (2019).

108 O. Nishikawa, T. Okada, H. Yoshikawa, *et al.*, “High-speed holographic-stereogram cal-

culation method using 2D FFT," in *Diffractive and Holographic Device Technologies and Applications IV*, **3010**, 49–57, SPIE (1997).

109 R. Oi, T. Mishina, and K. Yamamoto, "Real-time IP–hologram conversion hardware based on floating point DSPs," in *Proc. SPIE 7233, Practical Holography XXIII: Materials and Applications, 723305*, **723305**, 723305–1 – 11, SPIE (2009).

110 Texas Instruments, "TMS320C6678 Multicore Fixed and Floating-Point Digital Signal Processor Datasheet," Tech. Rep. November 2010, Texas Instruments (2014).

111 Intel, "Intel Xeon Phi Coprocessor," (2019).

112 J. Fang, H. Sips, L. Zhang, *et al.*, "Test-Driving Intel Xeon Phi," in *ICPE '14 Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, 137–148, ACM (2014).

113 OpenMP Architecture Review Board, "OpenMP Official Website," (2019).

114 K. Murano, T. Shimobaba, A. Sugiyama, *et al.*, "Fast computation of computer-generated hologram using Xeon Phi coprocessor," *Computer Physics Communications* **185**(10), 2742–2757 (2014).

115 N. Tanabe, Y. Ichihashi, H. Nakayama, *et al.*, "Speed-up of hologram generation using ClearSpeed Accelerator board," *Computer Physics Communications* **180**(10), 1870–1873 (2009).

116 G. Kyriazis, "Heterogeneous system architecture: A technical review," tech. rep., AMD (2012).

117 Y. Yamamoto, N. Masuda, R. Hirayama, *et al.*, "Special-purpose computer for electroholography in embedded systems," *OSA Continuum* **2**(4), 1166 (2019).

118 NVidia Corporation, “Jetson TX1 Module.”

119 R. W. Gerchberg and W. O. Saxton, “A practical algorithm for the determination of phase from image and diffraction plane pictures,” *Optik* **35**, 237–246 (1972).

120 P. W. Mash and T. D. Wilkinson, “Realtime hologram generation using iterative methods,” in *Proc. SPIE 6252, Holography 2005: International Conference on Holography, Optical Recording, and Processing of Information*, 62521O–1–62521O–9, SPIE (2006).

121 D. Blinder, A. Ahar, S. Bettens, *et al.*, “Signal processing challenges for digital holographic video display systems,” *Signal Processing: Image Communication* **70**(October 2018), 114–130 (2019).

122 A. Shan, “Heterogeneous processing: A strategy for augmenting moore’s law,” *Linux J.* **2006**, 7– (2006).

123 ARM Ltd., “White Paper: big. LITTLE Technology : The Future of Mobile,” tech. rep., ARM Ltd. (2013).

124 NVidia Corporation, “Harness AI at the Edge with the Jetson TX2 Developer Kit,” (2019).

125 NVidia Corporation, “Jetson AGX Xavier Developer Kit,” (2019).

126 NVIDIA, “Jetson NANO Module,” (2019).

127 NVidia Corporation, “Jetson Xavier NX introduction page,” (2019).

128 ARM Ltd., “Neon Architecture,” (2019).

129 NVIDIA, “NVIDIA Brings CUDA to Arm, Enabling New Path to Exascale Supercomputing,” (2019).

130 NVIDIA, “NVIDIA Announces CUDA-X HPC,” (2019).

131 Xilinx Inc., “UG1393: Vitis Unified Software Platform Documentation: Application Acceleration Development,” tech. rep. (2019).

132 NVIDIA, “Jetson TX2 introduction page,” (2019).

Youchao Wang is currently pursuing the Ph.D. degree in engineering from the University of Cambridge, Cambridge, U.K, where he previously received the M.Phil. degree. He received the B.Eng. (Hons.) degree in electronic engineering from University of Manchester and the B.Eng. degree in electrical and electronic engineering from North China Electric Power University, Beijing, China, under a joint degree program in 2018. His research interests include optical processing, CGH hardware implementations, Internet of Things applications and low cost hardware system design.

Daoming Dong received the B.Eng. degree with first class in electronics from a joint program between University of Liverpool (UoL) and Xi'an Jiaotong Liverpool University (XJTLU) in 2016. He then moved to Imperial College London, where he received the MSc. degree with distinction in Material Science and Engineering in 2017. He is currently a second-year PhD student under the supervision of Prof. Tim Wilkinson in the centre of molecular materials for photonics and electronics (CMMPE) group, Department of Engineering, Cambridge University, Cambridge. His research relates to accelerate and optimize the generation process of computer-generated hologram (CGH) via configurable hardware for the next generation 3D holographic displays.

Peter J. Christopher originally graduated from Bristol University in 2014 with a first class M.Eng. degree in Civil Engineering, before spending two years working as a Software/R&D Engineer for Autodesk's Advanced Manufacturing Group focusing on additive manufacture and 3D Printing. Looking for a new challenge, he joined the CDT in Ultra Precision based out of the Institute for Manufacturing at Cambridge University in 2016 and is currently working with Prof. Tim Wilkinson in the CMMPE group on high-power areal projections systems for additive manufacture.

Andrew Kadis originally graduated from the University of Adelaide, Australia in 2010 with a first class degree in Engineering and a bachelor of Computer Science. Before commencing his PhD studies in 2018, he had considerable experience in industry; working on embedded systems in drones, medical devices and life sciences equipment. He is currently at Cambridge University working with Prof. Tim Wilkinson in the CMMPE group.

Ralf Mouthaan obtained a Physics MSci degree from the University of Nottingham in 2008 before joining the UK's National Physical Laboratory as a microwave metrologist where his research was focused on maintaining and developing the UK's electromagnetic exposure standards. More recently, Ralf has obtained an MRes in Sensor Technologies from the University of Cambridge, where he is now pursuing a PhD investigating holographic mode excitation in optofluidic waveguides.

Fan Yang received the B.Eng. degree with first class from the University of Sydney in 2017 and joined the CMMPE group, Department of Engineering, Cambridge University as an MPhil student in 2018. He is continuing his research in the CMMPE group as a PhD student under the supervision of Prof. Tim Wilkinson to develop a compatible and efficient holographic 3D display system.

Timothy D. Wilkinson received the undergraduate degree from Canterbury University, Riccarton, New Zealand, and the Ph.D. degree from Magdalene College, Cambridge, U.K., in 1994. He is currently a Professor of photonic engineering in the Department of Engineering, Cambridge University, Cambridge, and a Fellow of Jesus College. He has been working in the field of photonics, devices, and systems for more than 20 years. His current research has been into applications of holographic technology. This includes new liquid crystal device structures based on sparse arrays of vertically grown multiwall carbon nanotubes, where the tubes are used as tiny electrodes to great 3-D electric field profiles and graded refractive index structures, which may have applications such as switchable lenslet arrays and 3-D displays.

List of Figures

List of Tables