HoloBlade: An Open-Hardware Spatial Light Modulator Driver Platform for Holographic Displays

ANDREW KADIS^{1,*}, YOUCHAO WANG¹, DAOMING DONG¹, PETER CHRISTOPHER¹, RALF MOUTHAAAN¹ AND TIMOTHY D. WILKINSON¹

¹Center for Molecular Materials, Photonics and Electronics, Department of Engineering, University of Cambridge

*Corresponding author: ack49@cam.ac.uk

Abstract: Spatial Light Modulators (SLMs) are a key research tool in several contemporary applied optics research domains. In this paper, we present the argument that an open platform for interacting with SLMs would dramatically increase their accessibility to researchers. We introduce *HoloBlade*, an open-hardware implementation of an SLM driver-stack and provide a detailed exposition of *HoloBlade*'s architecture, key components and detailed design. An optical verification rig is constructed to demonstrate that *HoloBlade* is capable of providing Fourier imaging capability in a $4f$ system. Finally, we discuss *HoloBlade*'s future development roadmap and the opportunities that it presents as a research tool for applied optics.

1. Introduction

Spatial light modulators (SLMs) are an important tool for applied optics research across a range of diverse fields such as holographic displays [1], telecommunications [2], astronomy [3], microscopy [4] and optical computing [5]. However, contemporary SLMs are a high-end scientific tool and are not readily accessible to the wider research community. Individual devices are expensive, pose integration challenges and typically run on proprietary data interfaces.

Fig. 1. *HoloBlade*, an open-hardware SLM driver-stack

Cost is a major barrier to the uptake of SLMs, however previous work on low-cost SLMs, from both from the scientific-user [6] and the mass-production perspectives [7], have not lead to significant cost reductions. In the opinion of the authors, a significant cost to device manufacturers is the engineering infrastructure (hardware development, software development and continuous support) to enable their end-users to interface with a SLM. SLMs are specialist, low-volume, industrial-scientific devices where the lack of economies-of-scale translate to a higher proportion of the unit cost resulting from research and development. Hence, a motivation behind *HoloBlade* is to release freely an implementation of this engineering infrastructure as open-hardware to significantly reduce the cost of SLMs across the field.

Beyond cost, SLM driver interfaces also pose an accessibility problem for researchers and end-users. SLM manufacturers often employ proprietary interfaces which results in researcher's systems and software being locked into a single vendor. Moreover, SLMs typically employ consumer video interfaces such as HDMI [8], making it difficult to incorporate research innovations such as dedicated computer-generated holography (CGH) acceleration hardware [9] and data compression [10]. Hence, there is a strong case for a set of common interfacing standards to drive SLMs.

It is desirable for the implementation to be open. Compared with the traditional, commercial-off-the-shelf (COTS) approach of hardware being a proprietary black-box, open-hardware is published and open-hardware developers actively seek peer-review. The open-hardware approach has been shown to produce designs which are better suited for their given application with fewer bugs [11]. For scientific equipment in particular, a strong case has been established for open-hardware reducing costs and providing higher value research for funding bodies [12].

Thus, the authors propose *HoloBlade*, an open-hardware driver-stack for SLMs. The primary application for *HoloBlade* is holographic display applications, but the technologies are also applicable to adjacent research fields utilising SLMs. A functional *HoloBlade* implementation is presented here along with the architecture, key components and detailed design. An optical test system is then used to verify functionality and demonstrate that the system is performing appropriately. Finally, we discuss *HoloBlade's* future development roadmap. By adopting an open-hardware approach, it is hoped to democratise SLMs and increase their accessibility to researchers.

2. Implementation

In this section the *HoloBlade* driver-stack is described in-depth. The relevant design files are hosted on the *HoloBlade's* GitHub repository, available [here](#). *HoloBlade* is released under the CERN Open Hardware Licence's permissive variant, [CERN-OHL-P v2](#).

2.1. Driver Architecture

Interfacing a high-speed SLM to a PC is a complex task. In order to manage the flow of data in a controlled manner, several technologies have to co-ordinate to produce the desired outcome. This collection of technologies is referred to as a driver-stack, as shown in Figure 2. The fact that several technologies have to work in conjunction is one of the key challenges with driving SLMs.

At the top of the *HoloBlade* driver-stack, shown in Figure 2, an end-user writes an application-layer piece of software to prescribe their desired SLM functionality. This piece of software calls the appropriate PC drivers to transfer data between the PC and the physical electronics driving the SLM. A custom electronics driver printed circuit board (PCB) mounts the SLM and provides appropriate connectivity to the PC. Here, a Field-Programmable Gate Array (FPGA) provides interfacing logic between the PC-specific calls and the format expected by the SLM. Additionally, the SLM needs appropriate optomechanical mounting. The mounting needs to accommodate the optical requirements of the end-user, support the connector interface between the driver board and the SLM daughter board, and alleviate electromechanical issues such as strain.

Fig. 2. The *HoloBlade* driver-stack

2.2. Key Components

The detailed design of the system was driven by the choice of key components. USB 3.0 was used for the PC data interface, a Field-Programmable Gate Array (FPGA) to interface handle interfacing logic and a Ferroelectric liquid crystal (FLC) Binary-Phase SLM as the display device.

Fig. 3. Binary-phase FLC SLM used in *HoloBlade* implementation

2.2.1. Data Interface

USB 3.0 provides the physical-layer interface between the PC and the dedicated driver electronics. The use of a general-purpose data interface allows the *HoloBlade* driver calls to be tailored to our specific use-case of displaying SLM data. USB 3.0 offers considerable bandwidth at 5.0 Gbps [13] and offers a future migration path with the USB standards committee already committed to faster iterations in USB 3.1, USB 3.2 and USB 4.0 [14]. USB 3.0 ports are readily available on contemporary PCs and the interface uses differential signalling hence is highly noise-resilient and reliable for scientific environments. The specific chip used is a FTDI FT601 interface chip and is supplied with existing software drivers for multiple Operating Systems [15].

USB 3.0 was intentionally selected over consumer electronics audio-visual interfaces (such as HDMI). Interfaces such as HDMI are designed to transmit 8-bit Red, Green Blue Video data at approximately 60 Hz. Whilst some SLMs are used at similar speeds and bit-depths (although in monochrome), these data transmission specifications are not universally appropriate for all SLMs such as the 2400 Hz binary-phase SLM employed here. The advantage of using USB 3.0 is that a dedicated SLM-specific transmission protocol can be built on top of it which is better suited to the application.

2.2.2. Interfacing Logic

An FPGA was used to manage the data marshalling between the USB 3.0 chip and the SLM. FPGAs are highly-configurable hardware that can be thought of as a sea of logic gates capable of being programmed to implement a given logic design. Hence they are distinct from traditional sequential execution CPUs and are considered to be a form of hardware rather than a form of software [16].

FPGAs are used extensively for high-performance video and imaging applications [17]. FPGA designs are specified in portable register-transfer level (RTL) logic languages, allowing a given design to be deployed across different FPGA families from different vendors as well as being incorporated into dedicated silicon [18]. This scalability is important for *HoloBlade* to see significant uptake. Moreover, using a FPGA allows dedicated CGH hardware accelerators to be implemented directly in the FPGA fabric, supporting the current trends within the field [19]; performing the complex CGH routines in dedicated hardware increases performance and reduces the software complexity for researchers.

The particularly FPGA selected here is a low-cost Lattice iCE40 FPGA. It is available in a quad-flat package (QFP) which is significant as it enables the fabrication and assembly of PCBs at a significantly lower cost-point than the majority of Ball-Grid Array (BGA), FPGA packages [20].

2.2.3. Display Device

The goal of *HoloBlade* is to eventually support multiple SLMs from different manufacturers but a single SLM was chosen for the implementation presented here. The selected SLM is a binary-phase FLC with a 1280x1280 display resolution and a 5.6 μm pixel-pitch. This device incorporates 2 internal display buffers for tear-free high-speed operation and is capable of being updated at frequencies up to 2.4 kHz. The advantage of using a FLC SLM is their ability to operate at high frame rates and their low-cost due to their binary nature. *HoloBlade* is primarily intended for display applications, and this is an appropriate device for this task as high-speed, binary phase SLMs have been shown to elicit a response well-matched to the human psycho-visual perception system [21]. Regarding cost, SLMs are typically sold at high price points but binary-phase FLC devices are able to scale to lower unit costs through economies of scale [22].

2.3. Detailed Design

2.3.1. FPGA Hardware Implementation

The FPGA interfacing logic has three key requirements:

1. Extract data from the USB FT601 chip.
2. Restructure the data into the specific format expected by the SLM.
3. Handle appropriate control and configuration of the SLM; ensure clock speeds are correct, the display is correctly DC-balanced and the device is operating in the correct state.

The logical design to implement these requirements is shown in Figure 4. It incorporates several key design features. The design spans two clock domains. Hence, to traverse across clock domains, data is sent through a dedicated dual-clock first-in-first-out (FIFO) data-pipe in keeping with best-practice for multi-clock domain designs [23]. The bulk of the logic is implemented in two complex state machines, a USB 3.0 Interface Controller state machine and a Display Data Controller state machine. A third state machine controls the timing of the SLM display updates, manages buffer switches and ensures that the SLM is DC-balanced. Finally, in addition to the main datapath, the design also incorporates an additional configuration datapath to configure

Fig. 4. Block diagram of the logical components in the synthesised FPGA Design. The majority of the design is common across all target SLMs, with only the Display Data Controller and Configuration Datapath (outlined in blue) bespoke to the specific SLM.

Fig. 5. Display Data Controller state machine logic

the SLM state and support a dedicated test mode where the data loaded into the SLM's internal buffers can be interrogated. The test mode was used extensively during development.

Data is transferred to the SLM via a custom parallel data interface and thus requires bespoke logic code. The Display Data Controller state machine, shown in Figure 5, implements this functionality. The state machine sits in IDLE state until a full line of 1280 pixels is available in the dual-clock FIFO. It then clocks out the data in the appropriate format expected by the SLM before proceeding to wait for the next line, continuing to clock out all lines of data before waiting for the next frame. This implementation ensures the SLM's internal data buffers are not updated whilst displaying data to avoiding screen-tearing effects, and ensuring that the timing constraints required by the SLM are met.

Fig. 6. USB 3.0 Interface Controller state machine logic

Interfacing with the USB 3.0 chip also required a dedicated state machine, shown in fig 6. Following each display update (or buffer switch), the state machine shall check if data is available and, if available, it shall begin clocking it into the dual-clock FIFO. If no data is available it shall halt until the next display update. There are several timing requirements in interfacing with the the chip and these are handled by dedicated single-cycle states. The state machine clocks out an entire frame of data before waiting for the next buffer switch. It is also possible that all the data has been read out of the USB 3.0 chip and dedicated recovery states handle this edge case.

Individual logic modules were developed using myHDL, a framework that allows register-transfer level (RTL) logical designs to be synthesised from python files [24]. The generated verilog files were then collated into a hierarchical logical design for the FPGA. Finally the Lattice toolchain was used to generate the appropriate bitstream hex file to program the FPGA.

As it is not possible to debug FPGA logic code [25], simulation was used extensively in the development of individual logic modules. To support this, simulation objects for the FT601 USB 3.0 chip and the dual-clock FIFO (an intellectual property (IP) block supplied by Lattice) were

constructed in myHDL alongside the logical design; these permitted the entire design to be tested and verified in simulation. Beyond simulation, GPIO tests point from the FPGA were used to debug interfacing issues between the FPGA, the FT601 and the SLM; the simulated blocks were then updated accordingly based on observed behaviour.

Finally, the SLM's test mode was employed to read back the data loaded in the SLM's internal display buffers. This capability allowed system functionality to be verified during development of the FPGA logic, as the SLM pixels are too small to individually inspect.

2.3.2. Electronics Design

The Electronics architecture developed in-house supports two key functions. The FT601 USB 3.0 chip interfaces with the input side of the iCE40 FPGA and the output side of the iCE40 FPGA interfaces with the FLC SLM's custom data-interface. The remainder of electronics design is supporting circuitry to accomplish this.

The data lines between the FT601 chip and the iCE40 FPGA are high-speed lines. The chip converts the USB data lines to a 32-bit wide synchronous parallel data bus. Due to the speed, the 32 data lines and clock line are implemented as matched-length traces to ensure high-fidelity signal integrity and timing, shown in Figure 8c. The lines were also routed through external resistor packages to provide series termination to the transmission lines before they are sampled as inputs to the FPGA.

The lines between the FPGA and the SLM are also matched-length traces. Connection between the SLM and the PCB is realised via an external flexible-PCB ribbon cable which has been developed bespoke for the SLM, see Figure 8b. Care is needed around this connector, as it is mechanically fragile and the design of the mechanical mounting has to support this.

With regards to supporting circuitry, the design uses 2 dedicated oscillators. One is required to drive the USB chip's internal timing and the other is used to drive the FPGA and consequently the SLM. Dedicated power circuitry provides well-conditioned power supplies to the components and programming circuitry is used to write the RTL implementation to the FPGA. Finally, general purpose input output (GPIO) lines provide debugging capability to support the development of the FPGA firmware.

The bespoke PCB is a 4-layer design and the layer stack-up is shown in Figure 8d. From a design-for-manufacture perspective, the design has been explicitly engineered to ensure that the board can be fabricated and assembled for a low-cost manufacturing process. Only leaded components were used, which results in significantly lower board fabrication, manufacturing and quality inspection costs [26]. This was a non-trivial exercise as it significantly curtailed the available parts which could be used for certain components on the board. This is in accordance with the desire for *HoloBlade* to be as accessible as possible.

2.3.3. Driver-Level Software

The driver-level software is layered upon the USB 3.0 drivers supplied by FT601. This software-layer allows an end-user to author C++/C# software which allows frame data and SLM configuration data to be sent to the SLM. Additionally, the SLM configuration, current state and loaded frame data can be read back to the PC.

The FT601 drivers are employed in asynchronous, non-blocking mode. This allows multiple writes of frame data to be queued up internally within the FT601 drivers while the FT601 is waiting for the PC's USB Host Controller to release access to the bus. This mode is recommended by FTDI to maximise data throughput, a requirement given the high data rates of the SLM [27].

2.3.4. Application-Level GUI Software

Although not formally part of the *HoloBlade* driver-stack, a Graphical User Interface (GUI) application is disseminated alongside the driver-stack. The intention is that the end-user shall

Fig. 7. *HoloBlade* bespoke electronics with circuitry shaded according to functionality.

author their own application-level software to employ the *HoloBlade* with the desired functionality. This tool was used extensively during development of the driver-stack and its inclusion acts as an interface for simpler use-cases (such as the test patterns shown in Figure 12) as well as an exemplar of how to deploy *HoloBlade* for more advanced use-cases.

2.3.5. Mechanical Design

To support optomechanical integration of the SLM with the end-users optical system, a bespoke optical mount was 3D printed in polylactide (PLA) thermoplastic using a low-cost fused deposition modelling (FDM) printer. The design readily integrates with a standard cage-mount optical

(a) Bare PCB

(b) Bespoke flexible-PCB ribbon cable

(c) Matched-length data traces

(d) Board stack-up

Fig. 8. Design features of the bespoke *HoloBlade* electronics.

Fig. 9. Optomechanical mounting

Fig. 10. GUI application screenshot

system, with the SLM mounted perpendicular to the optical axis. Additionally, a heat-set insert is mounted directly underneath the reflective face of the SLM providing a standard optical mount female thread. This allows the SLM to be further secured and permits off-axis mounting if required experimentally.

2.4. Limitations

A driver-stack is a complex series of interacting technologies, thus the desired functionality has been achieved but there are several limitations to the implementation highlighted here for transparency. The root-cause of individual issues are well understood and shall be addressed in a future PCB iteration.

Firstly, there is an issue with the PCB design currently hindering the full-bandwidth operation at USB 3.0 speeds. The solution has been to run the device at USB 2.0 speeds. Although significantly slower than the maximum achievable update speeds, this maintains a display update rate of 100 Hz. Secondly, it is possible to corrupt certain pixel's data values when they enter the SLM's frame buffer. The issue is rare, only occurring for approximately 100 pixels for each 1.2 MegaPixel frame, but it does lead to some level of data corruption in the displayed frame. The FT601 chip supplies additional control lines which are currently not used and incorporating these into a future PCB revision shall resolve the data corruption issue.

3. Results

A $4f$ optical system was built up to verify that the driver stack was functioning as intended. The optical configuration and the corresponding results are presented here.

3.1. Optical System

A $4f$ optical system, as shown in Figure 11, is used to test the SLM. The FLC SLM operates in reflective mode, hence a non-polarising beamsplitter is used to image the replay field generated by the SLM. Linear polarisers placed on the illumination and imaging arms are independently rotated to remove the zeroth-order of the replay field.

On the illumination arm, coherent light is supplied with a 658nm pig-tailed single mode laser diode. Collimation is achieved using a Thorlabs 23mm lens selected such that the Gaussian beam diameter corresponds to the SLM's active area.

For the imaging arm, a monochrome Basler acA1920-150um machine vision camera placed at the imaging plane allows the replay field to be imaged. The image is focused by a second discrete lens mounted along the optical axis of the imaging arm.

Fig. 11. $4f$ optical system used to verify SLM functionality.

3.2. Test Patterns

Fig. 12. Outputs from the *HoloBlade* driver-stack used in an experimental optical system. The expected replay fields are formed for three distinct test patterns. The left column are the binary-phase masks loaded onto the SLM, with a polariser film on top for contrast, and the right column shows the images acquired at the replay field plane imaged by the Basler USB 3.0 camera.

In order to ascertain that the *HoloBlade* driver-stack was functioning as expected, several test patterns were loaded onto the system to demonstrate system functionality. These include a vertical grating, a horizontal grating and a checkerboard pattern, as shown in Figure 12.

Clear diffraction fringes are seen along the vertical and horizontal axes for the horizontal and vertical gratings respectively. The diffraction patterns exhibit a sinc envelope behaviour with the intensity trailing off at the higher orders towards the edges of the image. For the checkerboard, diffraction fringes are seen across the replay field. Intensity is highest towards the centre along

the horizontal and vertical axes with additional fringes visible towards the corners.

Note that there are some artifacts of the SLM’s construction. The dead space between the SLM pixels manifests itself as discrete spots residing just outside the central order. These can be seen in all of the replay field results. Additionally, the majority of the zeroth-order in the centre has been removed by the polarisers, but there is still some light remaining at the centre, this is most likely due to the dead-space between pixels, tolerance in the SLM’s construction or errors in the polarising optics.

To quantify the diffraction efficiency of the experimental system, the relative intensities for different regions of the replay field were determined experimentally and have been collated in Table 1. The corresponding SLM test pattern was the vertical diffraction grating test pattern shown in (Figure 12c). Approximately 33% of the intensity was concentrated in the replay field’s first-order, the rest of the light being distributed across the other diffraction-spot orders and the aforementioned artifacts present in the remainder of the replay field.

Table 1. Relative intensity [%] of distinct replay field regions for a horizontal grating.

	-5 th Order	-3 rd Order	-1 st Order	0 th Order	+1 st Order	+3 rd Order	+5 th Order	Remainder of Replay Field
Relative Intensity [%]	0.5	2.5	32.9	16.3	33.5	2.7	0.7	10.9

In summary, the results shown demonstrate that the *HoloBlade* driver-stack is functioning as intended. The replay field formed at the imaging plane shows the expected results as predicated by Fourier optics [28].

4. Development Roadmap

The work presented here is an initial implementation for *HoloBlade*. Further improvements are desirable for it to one day grow to an ecosystem of holographic and SLM support technologies.

In the medium term, we foresee two key developments to accelerate *HoloBlade*’s uptake. The first is to extend the current set of C++/C# drivers to support Matlab and Python run-times. This would allow end-users to write to a SLM directly from their Matlab and Python scripts, permitting rapid testing and development of holographic display algorithms and would dramatically increase SLM accessibility for research applications.

The second is expansibility. Presented here is an initial implementation for a single SLM; we envision future *HoloBlade* designs expanding upon the open-hardware implementation presented here to support multiple individual SLMs. The eventual goal is for end-users to be able to write a piece of *HoloBlade* compatible software once and this can be deployed onto multiple SLMs from different manufacturers. The majority of the design is portable across different target SLMs, as shown in Figure 4. However it is recognised that certain SLMs are only available with manufacturer-supported driver interfaces which require different interfaces such as HDMI. For such cases, the intent is to develop a *HoloBlade* USB 3.0 to HDMI adaptor board. This shall permit a given *HoloBlade* user to develop their software using the appropriate *HoloBlade* functions and output to both USB 3.0 and HDMI SLM driver boards.

Beyond this, the eventual goal is for *HoloBlade* to act as a catalyst for the development of holographic display systems [29]. As an emerging technology, holographic displays will have to overcome challenges such as high data-bandwidth requiring data compression [30]. As open-hardware, *HoloBlade* is well-positioned to act as a bedrock technology to address scaling issues as the technology matures.

5. Conclusion

We have introduced *HoloBlade*, to the best of the author’s knowledge, the first implementation of an open-source SLM driver-stack. We have discussed the limitations of existing SLM interfaces

and presented the case for an open-source SLM driver-stack. The design goals for *HoloBlade* have been discussed, accompanied by a detailed discussion of architecture, key components and detailed design in this initial implementation. System functionality has been demonstrated on a 4f optical test rig. Finally, we have presented the future development roadmap for *HoloBlade*. The *HoloBlade* implementation detailed here represents an initial implementation, it is envisioned that future refinements will incorporate advanced features to grow the platform into an enabling tool for the wider research community.

Acknowledgments

The authors would like to thank Daniel McGraw for his work on the SLM flexible-PCB connector.

Portions of this work were presented at the OSA Imaging and Applied Optics Congress / OSA Optical Sensors and Sensing Congress in 2020, as presentation 3394464, *HoloBlade: An Open Platform for Holography*.

Disclosures

The authors declare no conflicts of interest.

References

1. A. Maimone, A. Georgiou, and J. S. Kollin, "Holographic near-eye displays for virtual and augmented reality," *ACM Transactions on Graph.* **36**, 1–16 (2017).
2. W. A. Crossland, T. D. Wilkinson, I. G. Manolis, M. M. Redmond, and A. B. Davey, "Telecommunications applications of LCOS devices," *Mol. Cryst. Liq. Cryst. Sci. Technol. Sect. A: Mol. Cryst. Liq. Cryst.* **375**, 1–13 (2002).
3. R. Davies and M. Kasper, "Adaptive optics for astronomy," *Annu. Rev. Astron. Astrophys.* **50**, 305–351 (2012).
4. M. Hasler, T. Haist, and W. Osten, "SLM-based microscopy," *Opt. Micro- Nanometrology IV* **8430**, 84300V (2012).
5. P. Ambs, "Optical computing: A 60-year adventure," *Adv. Opt. Technol.* (2010).
6. D. Huang, H. Timmers, A. Roberts, N. Shivaram, and A. S. Sandhu, "A low-cost spatial light modulator for use in undergraduate and graduate optics labs," *Am. J. Phys.* **80**, 211–215 (2012).
7. W. A. Crossland, T. D. Wilkinson, T. M. Coker, T. C. B. Yu, and M. Stanley, "The fast bitplane SLM: A new ferroelectric liquid crystal on silicon spatial light modulator designed for high yield and low cost manufacturability," *OSA TOPS* **14**, 102–106 (1997).
8. C. Wang, Y.-C. Hsu, and S.-H. Chan, "SLM-based educational kit for wave optics," *Educ. Train. Opt. Photonics: ETOP 2015* **9793**, 979315 (2015).
9. T. A. A. Kamatsu, R. Y. H. Irayama, H. Irotaka, N. Akayama, T. A. K. Akue, T. O. S. Himobaba, and T. O. I. To, "Special-purpose computer HORN-8 for phase-type electro-holography," *Opt. Express* **26**, 26722–26733 (2018).
10. F. Dufaux, Y. Xing, B. Pesquet-Popescu, and P. Schelkens, "Compression of digital holographic data: an overview," *Appl. Digit. Image Process. XXXVIII* **9599**, 95990I (2015).
11. E. Van Der Bij, J. Serrano, T. Wlostowski, M. Cattin, E. Gousiou, P. Alvarez Sanchez, A. Boccardi, N. Voumard, and G. Penacoba, "Open Hardware for CERN's accelerator control systems," *J. Instrumentation* **7** (2012).
12. A. Maia Chagas, "Haves and have nots must find a better way: The case for open scientific hardware," *PLoS biology* **16**, e3000014 (2018).
13. B. Dunstan, "USB 3.0 Architecture Overview," Tech. rep., USB 3.0 Technical Working Group (2009).
14. A. Lohse and J. M. Silva, "Advantages of Active Optical Cables for Industrial Applications," *PhotonicsViews* **17**, 50–53 (2020).
15. Future Technology Devices International Limited, "FT600Q-FT601Q IC Datasheet (USB 3.0 to FIFO Bridge)," Tech. rep., Future Technology Devices International Limited (2017).
16. S. Kesturt, J. D. Davis, and O. Williams, "BLAS comparison on FPGA, CPU and GPU," *Proc. - IEEE Annu. Symp. on VLSI, ISVLSI 2010* pp. 288–293 (2010).
17. K. Benkrid, D. Crookes, J. Smith, and A. Benkrid, "High Level Programming for FPGA Based Image and Video Processing Using Hardware Skeletons," *Proc. - 9th Annu. IEEE Symp. on Field-Programmable Cust. Comput. Mach. FCCM 2001* pp. 219–226 (2001).
18. I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *IEEE Transactions on Comput. Des. Integr. Circuits Syst.* **26**, 203–215 (2007).
19. Y. Wang, D. Dong, P. J. Christopher, A. Kadis, R. Mouthaan, F. Yang, and T. D. Wilkinson, "Hardware implementations of computer-generated holography: a review," *Opt. Eng.* **59**, 1 (2020).
20. A. Drumea and M. Pantazica, "Aspects of using low layer count PCBs for embedded systems with FPGA devices in BGA packages," *2016 IEEE 22nd Int. Symp. for Des. Technol. Electron. Packag. SIITME 2016* pp. 74–77 (2016).

21. A. J. Cable, E. Buckley, P. Mash, N. A. Lawrence, T. D. Wilkinson, and W. A. Crossland, "53.1: Real-time Binary Hologram Generation for High-quality Video Projection Applications," in *SID Symposium Digest of Technical Papers*, vol. 35 (2004), p. 1431.
22. T. D. Wilkinson, W. A. Crossland, T. Coker, A. B. Davey, and T. C. Yu, "Ferroelectric liquid crystal on silicon spatial light modulator designed for high yield and low cost fabrication: The fast bitplane SLM," *Ferroelectrics* **213**, 219–223 (1998).
23. Y. Li, B. Nelson, and M. Wirthlin, "Synchronization techniques for crossing multiple clock domains in FPGA-based TMR circuits," *IEEE Transactions on Nucl. Sci.* **57**, 3506–3514 (2010).
24. K. Jaic and M. C. Smith, "Enhancing hardware design flows with MyHDL," *FPGA 2015 - 2015 ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays* pp. 28–31 (2015).
25. J. Goeders and S. J. Wilton, "Effective FPGA debug for high-level synthesis generated circuits," *Conf. Dig. - 24th Int. Conf. on Field Program. Log. Appl. FPL 2014* (2014).
26. C. Liu, J. Wang, A. Zhang, and H. Ding, "Research on the fault diagnosis technology of intermittent connection failure belonging to FPGA solder-joints in BGA package," *Optik* **125**, 737–740 (2014).
27. Future Technology Devices International Limited, "Application Note 386: FT600 Maximize Performance," *Tech. rep.*, Future Technology Devices International Limited (2015).
28. J. W. Goodman, *Introduction to Fourier optics* (Roberts & Co. Publishers, 2005), 3rd ed.
29. Z. He, X. Sui, G. Jin, and L. Cao, "Progress in virtual reality and augmented reality based on holographic display," *Appl. Opt.* **58**, A74 (2019).
30. X. Xu, Y. Pan, P. P. M. Y. Lwin, and X. Liang, "3D holographic display and its data transmission requirement," *2011 Int. Conf. on Inf. Photonics Opt. Commun. IPOC 2011* pp. 3–6 (2011).