ARL-TR-9362 ● Dec 2021

DEVCOM ARMY RESEARCH LABORATORY

Army Educational Outreach Program High School Apprenticeship Report: Review of Commercial FPGAs and Benchmarking Using OpenFPGA

by Oluseyi Ayorinde and Derek Favorite

NOTICES

Disclaimers

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer's or trade names does not constitute an official endorsement or approval of the use thereof.

Destroy this report when it is no longer needed. Do not return it to the originator.

ARL-TR-9362 ● DEC 2021

Army Educational Outreach Program High School Apprenticeship Report: Review of Commercial FPGAs and Benchmarking Using OpenFPGA

Oluseyi Ayorinde
Sensors and Electron Devices Directorate, DEVCOM Army Research Laboratory

Derek Favorite
Summer Student, Army Educational Outreach Program Apprenticeship

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED (From - To)
December 2021	Technical Report	21 June–20 August 2021

4. TITLE AND SUBTITLE
Army Educational Outreach Program High School Apprenticeship Report: Review of Commercial FPGAs and Benchmarking Using OpenFPGA

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Oluseyi Ayorinde and Derek Favorite

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
DEVCOM Army Research Laboratory
ATTN: FCDD-RLS-SE
Los Angeles, CA 90045

8. PERFORMING ORGANIZATION REPORT NUMBER
ARL-TR-9362

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release: distribution unlimited.

13. SUPPLEMENTARY NOTES
ORCID ID: Oluseyi Ayorinde, 0000-0002-7028-4024

14. ABSTRACT

This summer's Army Educational Outreach Program apprenticeship consisted of two main focuses. The first was field programmable gate array (FPGA) research. We have examined a multitude of private companies that are developing their own FPGAs to be sold publicly. We collected data and statistics on some of the companies' most powerful devices and placed them in a spreadsheet for comparison. We have also taken a look at lots of research conducted on FPGAs using ACM Digital Library and IEEE Xplore to access conferences and articles. We spent a considerable amount of time scouring through articles gathering information on FPGAs to get a better sense of the current state of FPGAs. The second half of the summer focused on working in OpenFPGA. OpenFPGA is a Linux-based tool that is useful for mapping circuits to FPGAs and generating custom FPGA chips. We worked with the OpenFPGA developers to improve their documentation and experimented in OpenFPGA to see the effects of different design choices. We installed OpenFPGA on a virtual Linux machine, and ran multiple OpenFPGA tasks and got area estimates of FPGAs mapped with different circuits. We also developed a methodology for synthesizing unmapped circuits using Yosys.

15. SUBJECT TERMS
integrated circuits, field programmable gate array, FPGA, AEOP apprenticeship, OpenFPGA, Design Space Exploration

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Oluseyi Ayorinde
Unclassified	Unclassified	Unclassified	UU	23	19b. TELEPHONE NUMBER (Include area code) (310) 448-5394

Contents

List of Figures iv

List of Tables iv

1. Introduction 1

1.1 What is an FPGA? 1

1.2 Research 1

2. OpenFPGA 5

2.1 Setup and Compilation 6

2.2 Work with OpenFPGA Developers 6

2.3 Experiments in OpenFPGA 7

3. Conclusions 9

4. References 10

Appendix. Raw Data for Commercial FPGAs 11

List of Symbols, Abbreviations, and Acronyms 16

Distribution List 17

List of Figures

Fig. 1 Number of LUTs in commercial FPGAs 2
Fig. 2 Technology nodes of FPGAs 3
Fig. 3 Histogram of the amount of I/Os 3
Fig. 4 Histogram of the number of PLLs 4
Fig. 5 Amount of BRAM in FPGAs 4
Fig. 6 Number of multipliers in the FPGAs 5

List of Tables

Table 1 LUT sizes in commercial FPGAs 2
Table 2 DSP description 5

1. Introduction

The first half of the Army Educational Outreach Program summer apprenticeship began with research on field programmable gate arrays (FPGAs) to form an in-depth comparison of some of the industry's top devices.

1.1 What is an FPGA?

FPGAs are "semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing."[1]

FPGAs are uniquely designed to be versatile because of their ability to be reprogrammed after they have been implemented in a device, whereas a non-FPGA chip is set in its function when it is initially programmed.

FPGAs are used for aerospace and defense, application-specific integrated circuit (ASIC) prototyping, automotive, broadcast and professional audio-visual, various consumer electronics, data centers, high-performance computing and data storage, industrial and medical purposes, security, video and image processing, and wired and wireless communications.

1.2 Research

We have examined a multitude of private companies developing their own FPGAs to be sold publicly. These companies include Xilinx, Intel, Lattice, Flex Logix, Microsemi, Efinix, Menta, and Quick Logic. Each company is trying to push the envelope to develop a more efficient and powerful chip.

We collected data and statistics on some of the companies' most powerful devices and placed them in a spreadsheet for comparison. We used information from each company's product briefs, overviews, guides, manuals, and other forms of documentation in order to form our charts. The sum-total of the information gathered from the literature review can be found in the Appendix.

Table 1 highlights the number of inputs in the look-up tables (LUTs) in a histogram. LUTs are a customizable truth table programmed with preloaded values that are specific to the FPGA's purpose. As shown in the table, most of the FPGAs use 4-input LUTs, but a substantial number also use 6-input LUTs.

Table 1 LUT sizes in commercial FPGAs

No. of LUT inputs	Count
4	21
6	6

Figure 1 is a histogram of the number of LUTs themselves within the FPGAs we researched. The total number of LUTs gives a sense of the size of the FPGA. The majority of FPGAs have fewer than 1 million LUTs. However, some FPGAs, like the Intel Stratix, far exceed 1 million LUTs.

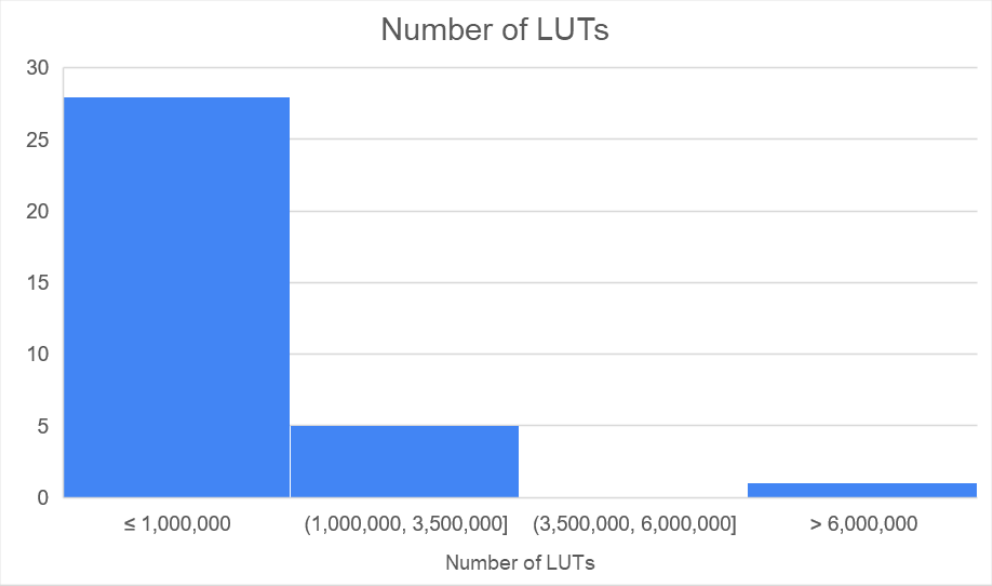

Fig. 1 Number of LUTs in commercial FPGAs

Figure 2 is a histogram that charts the sizes of technology nodes in the FPGAs we researched. Technology nodes refer to the size of the features that make up the chip and are measured in nanometers. The FPGAs researched span technologies from as old as 130 nm, all the way to as advanced as 7 nm. The most common technology nodes are 28 and 40 nm.

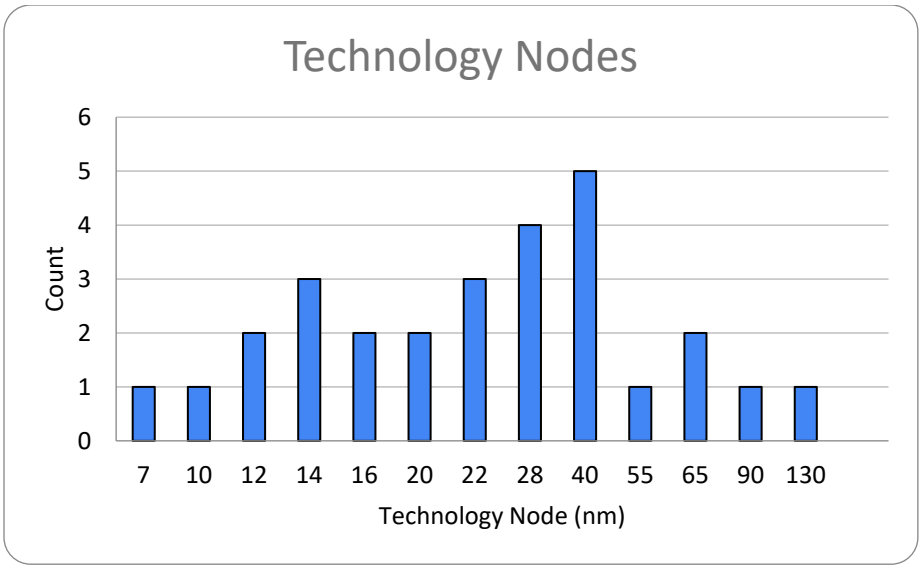

Fig. 2 Technology nodes of FPGAs

Figure 3 is a histogram of the number of inputs and outputs (I/Os) in the FPGAs we researched. I/Os connect the FPGA chip to other circuits and technologies. While most FPGA companies offer a range from 500 to about 1000 I/Os, the largest and most powerful chips offer more than 6000 I/Os.

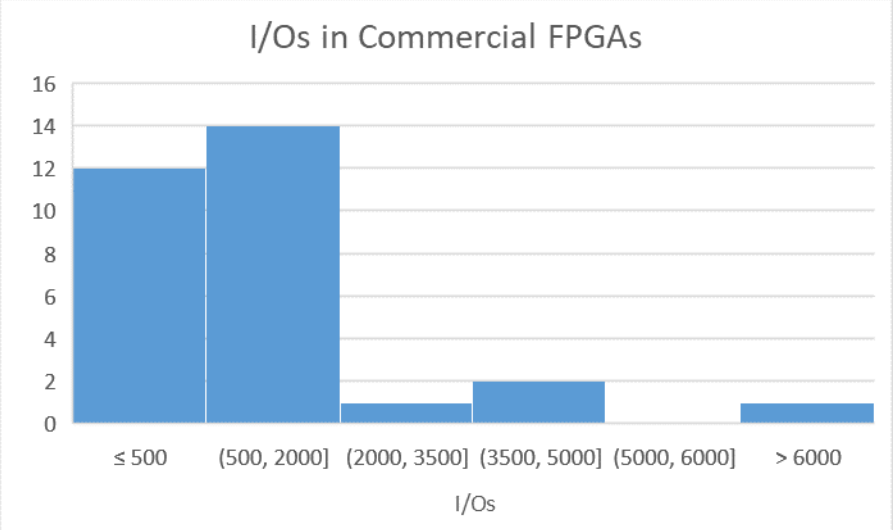

Fig. 3 Histogram of the amount of I/Os

Figure 4 displays a histogram of the number of phased-locked loops (PLLs) in the FPGAs we researched. The function of the PLL is to keep the clock uniform throughout the chip. The clock signal synchronizes the functionality of the FPGA. It also controls the timing of the functionality.

The majority of FPGAs have fewer than 15 PLLs. The Intel Stratix and Arria, however, can have up to 48 PLLs.

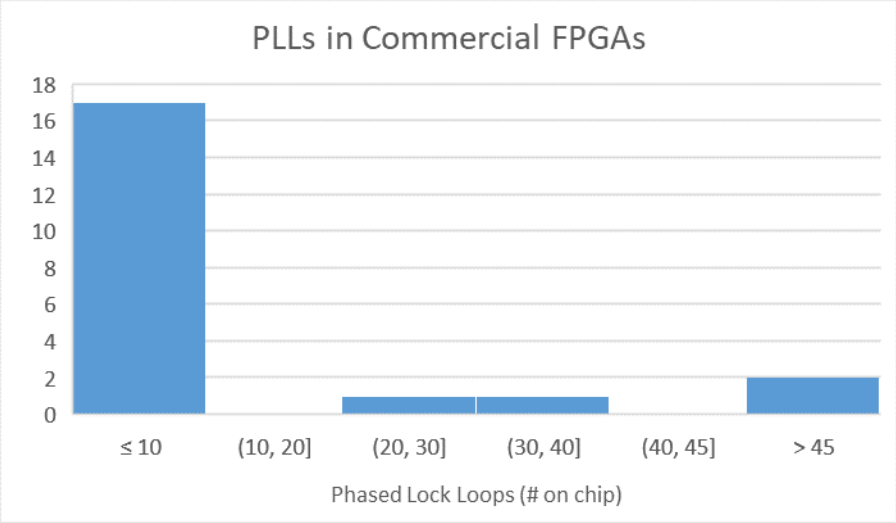

Fig. 4 Histogram of the number of PLLs

In Fig. 5 we measured the amount of memory in the FPGA in a histogram. Block RAM (or BRAM) stands for Block Random Access Memory. BRAMs are used for storing large amounts of data inside the FPGA. This plot shows memory in Mb (1 Mb = 1,000,000 bits). Most FPGAs have fewer than 30 Mb of BRAM.

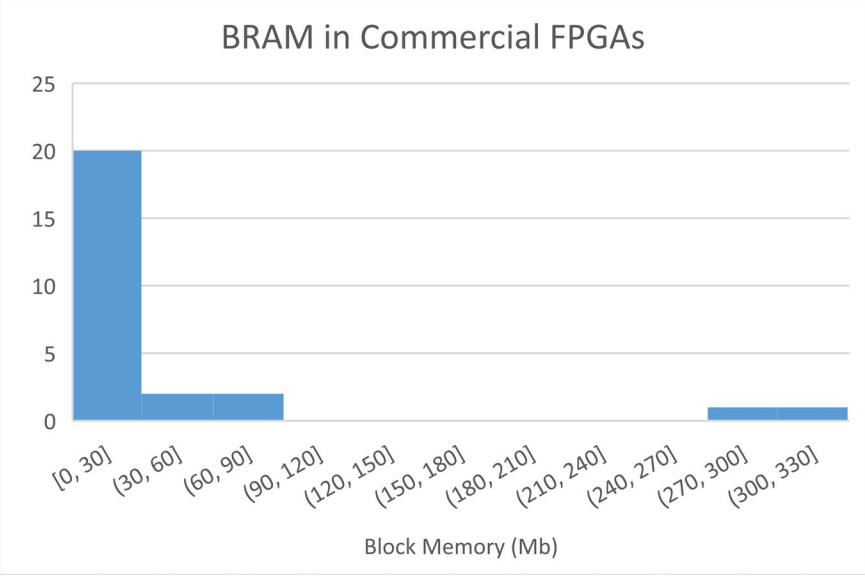

Fig. 5 Amount of BRAM in FPGAs

Figure 6 is a histogram of the number of multipliers in the FPGAs we researched. Multipliers are used to multiply efficiently because multiplication using LUTs is too expensive. Multipliers are often used for digital signal processing. Most FPGAs use fewer than 2000 multipliers. However, some implement more than 16000.

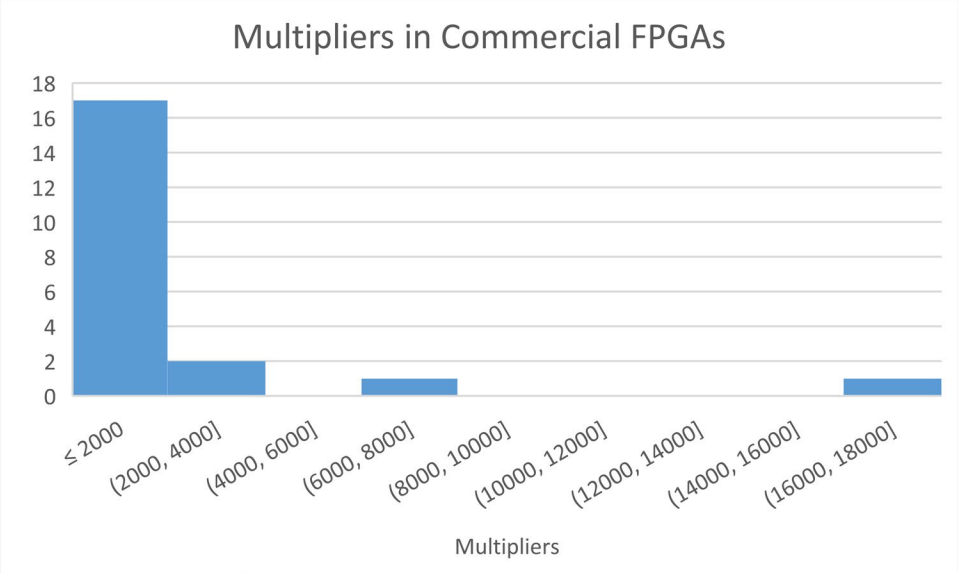

Fig. 6 Number of multipliers in the FPGAs

Table 2 describes the digital signal processors (DSPs) in the FPGAs of each of the companies we researched.

Table 2 DSP description

Company name	DSP description
Xilinx	25 × 18 Multiplier, 48 Bit Accumulator
Intel	18 × 19 Multipliers
Lattice	18 × 18 Multipliers
Flex Logix	40 DSP MACs 22-Bit
Flex Logix TSMC 40 ULP & 40LP	10 DSP MACs 22-Bit
Micro Semi	18 ×18 Multipliers
Efinix T4	18 × 18 Multipliers

2. OpenFPGA

The second half of the summer apprenticeship included working in OpenFPGA[2]. OpenFPGA is a Linux-based tool useful for mapping circuits to FPGAs and generating custom FPGA chips. We used OpenFPGA to help the developers improve their documentation and to see the effects of different design choices on FPGAs.

2.1 Setup and Compilation

We set up Ubuntu using the following steps:

1) Download the VirtualBox application.[3]
2) Download Ubuntu and follow the steps to set up.[4]
3) When given the option to install or have a trial of Ubuntu, select to fully download Ubuntu. It will not delete your current operating system.

In order to compile OpenFPGA:

1) Follow the steps in the OpenFPGA documentation.[5]
2) After cloning but before installing OpenFPGA, follow the path .github/workflows/install_dependencies_build.sh to install all of the packages that are needed to run OpenFPGA.

2.2 Work with OpenFPGA Developers

We worked alongside some of the OpenFPGA developers at the University of Utah to improve the OpenFPGA documentation.

Here are some things we brought to their attention:

- A lack of descriptions for output files. After a task was run, OpenFPGA would generate nondescript output files, which we had to decipher or inquire with the OpenFPGA developers to figure out.
- A lack of description for architecture filenames. There were many different architecture files, and it was unclear what the differences were between them. Later in this report we explain the architecture terms we figured out.
- A missing step in the compilation process, which we obtained directly from the developers as explained in Section 2.1.

Architecture File Descriptions:

- tileable_: “tileable” option is Set in VPR.
- _frac_: “frac” indicates a fracturable LUT.
- _wide_: Memory is two tiles wide.
- _thru_: “through_channel” option is set in VPR.
- _register_: BLE registers are used for additional functionality.

- _embedded_io_: I/O descriptions in the architecture file are generic.
    - Eight I/Os per block on all four sides
    - Circuit model name for I/O is “GPIO”
    - Tile name (VPR architecture) is “gpinpad/gpoutpad”
- _caravel_io_: I/O descriptions in the architecture file refer to the Caravel FPGA chip
    - Caravel I/O has one I/O per block on the top, left, and right sides of the FPGA, and six I/Os per block on the bottom.
    - Circuit model name for I/O is “EMBEDDED_IO”

2.3 Experiments in OpenFPGA

In our experiments in OpenFPGA we needed to be able to run the OpenFPGA flow, obtain an area estimate for an FPGA, and fix any unmapped circuits.

In order to run the OpenFPGA flow we used the following steps:

1) Opened the OpenFPGA directory in the terminal.
2) Sourced OpenFPGA according to the documentation.
3) Navigated to the “tasks” directory.
4) Selected the task we wished to run.
5) In the terminal, we typed “run-task” followed by the name of the folder for the task.

Once the task was run we could then use Yosys[6] to find an area estimate and other useful information. In order to gather this information, we followed these steps:

1) Return to the task directory in the file browser or terminal.
2) Navigate to the directory called “MIN_ROUTE_CHAN_WIDTH”.
3) Run Yosys.
4) Once Yosys is up and running, enter the following commands in order:
    a) read -vlog2k SRC/fabric_netlists.v
    b) read_liberty-lib-overwrite ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib

c) -check -top fpga_top

d) stat -liberty ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib

After gathering the information from Yosys, if we detected any unmapped circuits we would follow these steps:

1) Find the specific circuit in the Yosys output and get the name of the circuit.
2) Find that module in the Verilog code:
    a) Start at SRC/fabric_netlists.v.
    b) Probably one of the user-defined netlists
        i. Path: openfpga_flow/openfpga_cell_libraries
3) Find the code that is not getting mapped.
4) Find the replacement code.
5) Replace it with already-mapped code.

These steps were the fast way to fix an unmapped circuit. However, this limits the user-defined circuits to the skywater130 technology. To address this we instead used Yosys to create a synthesized version of the user-defined circuit, and used that to estimate the area. The following are the commands used to synthesize the user defined circuits:

1) read_verliog frac_mult_16x16.v
2) read_liberty –lib ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib
3) synth
4) dfflibmap -liberty ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib
5) abc -liberty ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib
6) opt_clean
7) stat -liberty ~/Desktop/OpenFPGA/skywater-pdk/libraries/sky130_fd_sc_hd/latest/timing/sky130_fd_sc_hd__tt_025_1v80.lib
8) write_verliog frac_mult_16x16_sky130.v

3. Conclusions

In this report, we have explored leading commercial FPGA architectures to get a sense of the current state-of-the-art in FPGA design and begun using OpenFPGA to conduct experimentation and design space exploration in custom FPGA architectures. These research activities put the US Army Combat Capabilities Development Command Army Research Laboratory in position to contribute to future FPGA design for Army-specific applications.

4. References

1. Xilinx.com. Field-programmable gate array (FPGA). [Accessed 2021 Oct 12]. https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html.
2. Tang X, Giacomin E, Chauviere B, Alacchi A, Gaillardon P. OpenFPGA: An open-source framework for agile prototyping customizable FPGAs. IEEE Micro. 2020;40(4): 41–48.
3. VirtualBox. Download VirtualBox. [Accessed 2021 Oct 12]. https://www.virtualbox.org/wiki/Downloads.
4. Ubuntu. Download Ubuntu Desktop. [Accessed 2021 Oct 12]. https://ubuntu.com/download/desktop
5. OpenFPGA. Welcome to OpenFPGA's documentation! [Accessed 2021 Oct 12]. https://openfpga.readthedocs.io/en/master.
6. Yosys Open Synthesis Suite. [Accessed 2021 Oct 12]. http://www.clifford.at/yosys.

Appendix. Raw Data for Commercial FPGAs

Company/chip name	Number of inputs	Number of LUTs	Lithography	I/Os
Xilinx Spartan-7	4	102	28 nm	400
Xilinx Artix-7	4	215	28 nm	500
Xilinx Kintex-7	4	478	28 nm	500
Xilinx Virtex-7	4	1955	28 nm	1200
Intel Agilex	4	2692760	10 nm	624
Intel Stratix	4	10200000	14 nm	2304
Intel Arria	4	1150000	20 nm	624
Intel Cyclone	4	220000	20 nm	284
Intel Max	4	50000	55 nm	500
Lattice Certus-NX	4	39000	28 nm	192
Lattice ECP5/ECP5-G5	4	85000	40 nm	
Lattice ECP3	4	150000	65 nm	
Lattice ECP2/M	4	95000	90 nm	520
Lattice XP2	4	95000		540
Flex Logix TSMC 12FFC+/FFC/16FFC+/FFC/FF+	6	2520	12	632(Input) 632(Output)
Flex Logix GF 12LP/12LP+	6	2520	12	632(Input) 632(Output)
Flex Logix 3 Gen2 TSMC 28HPC/HPC+/22ULP	6	2520	22	632(Input) 632(Output)
Flex Logix 4 GF 22FDX®	6	2520	22	632(Input) 632(Output)
Flex Logix TSMC 40ULP & 40LP	6	560	40	368(Input) 368(Output)
Micro Semi PolarFire FPGAs	4	50000	28 nm	584
Micro Semi IGLOO2	4	146124	65 nm	574
Micro Semi ProASIC3	4	1000000	130 nm	300
Micro Semi Fusion	4	1500000		252
Micro Semi				
Achronix Speedste7t	6	2600000	7 nm	
ADICSYS		100000	14 nm	
Efinix Ti35		36176	16 nm	146
Efinix Ti1000		969408	16 nm	268
Efinix T4		3888	40 nm	55
Efinix T120		112128	40 nm	270
Menta Small		1605		884
Menta Medium		4815		1388
Menta Large		40128		3808
Menta XL		130000		6500

Quick Logic Artic Pro			40 nm	
Quick Logic Artic Pro 2	44291	6018	22 nm	3712
Quick Logic Artic Pro 3	4		28 nm	
Quick Logic Antifuse				

Company/Chip Name	PLLs	DLLs	Block Memory	DSP Slice
Xilinx Spartan-7	8		4.2 Mb	160
Xilinx Artix-7	10		13 Mb	740
Xilinx Kintex-7	8		34 Mb	1930
Xilinx Virtex-7	24		68 Mb	3600
Intel Agilex	36		287Mb	8528
Intel Stratix	48		308 Mb	3456
Intel Arria	48		65.7 Mb	1518
Intel Cyclone	10		13.43 Mb	192
Intel Max	4		1.638 Mb	72
Lattice Certus-NX	3		0.885Mb	
Lattice ECP5/ECP5-G5	2	2	3.744Mb	
Lattice ECP3		2	4.42Mb	
Lattice ECP2/M	8	2	5.308Mb	
Lattice XP2	4	2	0.855Mb	
Flex Logix TSMC 12FFC+/FFC/16FFC+/FFC/FF+				40
Flex Logix GF 12LP/12LP+				40
Flex Logix 3 Gen2 TSMC 28HPC/HPC+/22ULP				40
Flex Logix 4 GF 22FDX®				40
Flex Logix TSMC 40ULP & 40LP				10
Micro Semi PolarFire FPGAs	8	8	33Mb	
Micro Semi IGLOO2	8		5Mb	
Micro Semi ProASIC3	1		.144Mb	
Micro Semi Fusion	2		.270Mb	
Micro Semi				
Achronix Speedste7t				
ADICSYS				
Efinix Ti35	4		1.53Mb	93
Efinix Ti1000	10			3520
Efinix T4	1		0.077Mb	
Efinix T120	8		5.407Mb	
Menta Small			0	
Menta Medium			1.422	

Menta Large			1.966	
Menta XL			5.252	
Quick Logic Artic Pro				
Quick Logic Artic Pro 2			.512Mb	
Quick Logic Artic Pro 3				
Quick Logic Antifuse				

Company/Chip Name	DSP Description	DSP Performance	Memory interfaces
Xilinx Spartan-7	25 × 18 multiplier, 48-bit accumulator	176 GMAC/s	800 Mb/s
Xilinx Artix-7	25 × 18 multiplier, 48-bit accumulator	929 GMAC/s	1,066 Mb/s
Xilinx Kintex-7	25 × 18 multiplier, 48-bit accumulator	2,845 GMAC/s	1,866 Mb/s
Xilinx Virtex-7	25 × 18 multiplier, 48-bit accumulator	5,335 GMAC/s	1,866 Mb/s
Intel Agilex	18 × 19 multipliers (17,056)		
Intel Stratix	18 × 19 multipliers (6,912)		
Intel Arria	Hardened single-precision floating-point multiplers/adders (1,518/1,518) 18 × 19 multipliers (3,036)		
Intel Cyclone	18 × 19 multipliers (384)		
Intel Max	18 × 18 multipliers (144)		
Lattice Certus-NX	18 × 18 multipliers (56)		
Lattice ECP5/ECP5-G5	18 × 18 multipliers (156)		
Lattice ECP3	18 × 18 multipliers (320)		
Lattice ECP2/M	18 × 18 multipliers (88)		
Lattice XP2	18 × 18 multipliers (32)		
Flex Logix TSMC 12FFC+/FFC/16FFC+/FFC/FF+	40 DSP MACs 22-BIt		
Flex Logix GF 12LP/12LP+	40 DSP MACs 22-BIt		
Flex Logix 3 Gen2 TSMC 28HPC/HPC+/22ULP	40 DSP MACs 22-BIt		
Flex Logix 4 GF 22FDX®	40 DSP MACs 22-BIt		

Flex Logix TSMC 40ULP & 40LP	10 DSP MACs 22-Bit		
Micro Semi PolarFire FPGAs	18 × 18 (1480)		
Micro Semi IGLOO2	18 × 18 (240)		
Efinix T4	18 × 18 (4)		
Efinix T120	18 × 18 (320)		

List of Symbols, Abbreviations, and Acronyms

AEOP	Army Educational Outreach Program
ARL	Army Research Laboratory
CLB	configurable logic block
DEVCOM	US Army Combat Capabilities Development Command
DSP	digital signal processor
FPGA	field programmable gate array
GF	GlobalFoundries
GPIO	General Purpose Input/Output
I/O	input/output
LUT	look-up table
MAC	Multiply Accumulate
PLL	phased-locked loop
SRC	Source
TSMC	Taiwan Semiconductor Manufacturing Company
VPR	Verilog Place-and-Route

1 (PDF)	DEFENSE TECHNICAL INFORMATION CTR DTIC OCA
1 (PDF)	DEVCOM ARL FCDD RLD DCI TECH LIB
1 (PDF)	DEVCOM ARL FCDD RLS SE O AYORINDE