A Project-Based Approach to FPGA-Aided Teaching of Digital Systems

Fajar Suryawan
Department of Electrical Engineering
Universitas Muhammadiyah Surakarta
Surakarta, Jawa Tengah 57102, Indonesia
Email: Fajar.Suryawan@ums.ac.id

***Abstract*—This article shares experience and lessons learned in teaching course on programmable logic design at Universitas Muhammadiyah Surakarta, Indonesia. This course is part of bachelor of engineering (electrical) degree program. Project-based approach is chosen to strengthen these students' understanding and practical skills. Each year's project involves challenges for the students to solve by implementing digital system on an FPGA design board.**

Here, background and curriculum context of the course will be presented. The projects and their challenges will be discussed. Finally, lessons learned and future improvement on the student projects will be discussed.

***Index Terms*—project-based learning, field programmable gate arrays, education, programmable logic design, hardware design languages, laboratories**

I. Introduction

Digital systems play a central role in our era. Digital devices are found in mobile phones, computers, cars, microwave ovens, TVs, TV remote controls, routers, switch hubs; practically every place where electricity exists. Generally this technology is embedded and hidden inside appliances in the form of integrated circuits (ICs). Given the prevalence of the technology, it is imperative for technical higher education institutions to have better ways to develop students' digital design skill.

One of the recent developments in the digital design is *programmable logic*, with one of its specific technologies: *field-programmable gate array* or FPGA. FPGA industry began to enter the market in 1980s and then enjoyed the explosive phase in 1990s [34]. This technology has enabled designers (and students) to program hardware, down to the logic block, using hardware description languages.

Aside from becoming a full-fledged application platform, FPGA (and other variants of programmable logic devices) is also a prototyping platform before a design (including design of *microprocessors*) is etched on ASIC (*application-specific integrated circuit*) boards [26], [42], [23], [4], [6], [8], [28], [30], [15], [40], [24]. Power electronics is also handled by FPGA [39], [38], [18], [33]. Traditionally computing-intensive applications have now enjoy the new platform. These applications include robotics and control [37], [39], [12], [32], [5], [46], signal processing and communication [35], [3], [17], [19], [32], [36], computer vision [2], [14], [43], [7], soft computing and artificial intelligence [44], [10], [25], numerical computing and cryptography [11], [13], [47], [21].

One can see that the immediate applications abound. For faculties and students, this platform's capability opens a whole new way to teach and learn digital system design. Several textbooks using this approach exist [29], [31], [22], [16], [27], [41].

While this learning concept has been in use at every major universities in developed countries wherever electrical/computer engineering bachelor program is offered, FPGA-based digital system teaching has not reached its full potential in Indonesia. This problem may be attributed to several factors such as department's lack of awareness, non-mobility of lecturers, or simply lack of implementable examples. This is despite the fact that an entry-level FPGA development/educational kit is very affordable.

This paper attempts to promote FPGA-based digital system education by sharing a workable practice that has been conducted for three years in an Indonesian university. It is of our opinion that FPGA-based digital system education serves not only as a vehicle to gain understanding of the inner working of digital systems (including a processor), but also as a training of the technology itself. The learning by doing approach used here has enhanced students' involvement, and hence their understanding of the subject. Several excellent works on this teaching method are available [1], [9], [20].

The subject under discussion in this paper is *Programmable Logic Design*, a third-year course.

II. Context and set-up

Electrical engineering bachelor degree program at Universitas Muhammadiyah Surakarta ("UMS") is a 4-year technical education program aiming at graduating skillful students ready for the industry. In the first and second years, all students undertake common basic courses on electrical engineering. At their third year of study there are two majors that students must choose: Electrical Power Systems or Electronics and Computer Systems. During those four years of their study, students take courses that can be categorized into several *streams*. These are *Mathematics and Physics* stream, *Electrical and Electronics* stream, and *Digital Systems* stream. (There are other courses such as English, Bahasa Indonesia, Islamic

978-1-5386-0549-3/17/$31.00 ©2017 IEEE

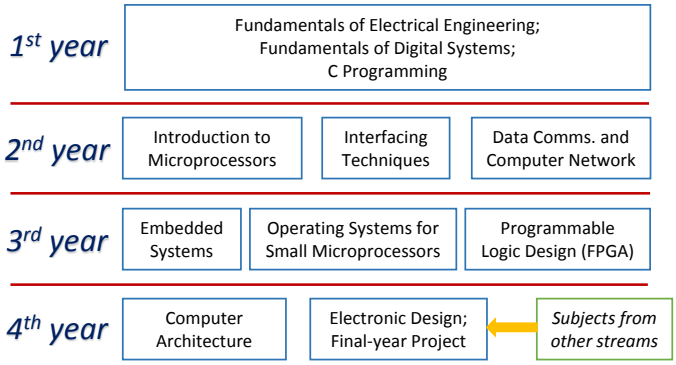


Fig. 1: Subjects in the digital-system "stream" in an electrical engineering bachelor degree curriculum. This is roughly what is implemented at UMS. Our focus in this paper is "Programmable Logic Design", a third-year subject.

Studies, and Muhammadiyah Studies, throughout the years, which can be categorized as *Life Skills* stream.)

The Digital Systems stream comprises of several subjects which includes *Fundamentals of Digital Systems*, *Introduction to Microprocessors*, *Programmable Logic Design*, *OS for Small Microprocessors*, *Embedded Systems*, and *Computer Architecture*. This is depicted in Figure 1.

Students enroll to the *Programmable Logic Design* subject if they choose to take the "electronics and computer systems" concentration (the other option is "electrical power systems" concentration). In this subject they will have the first exposure to FPGA design. However, as can be seen from Figure 1, this is not the first subject which presents digital world to the students. This fact introduces both relief and challenge. On the one hand students will have known some digital design technique which will make the exposition easier, but on the other hand they will also ask questions such as: what advantage does it offer? what is the difference between programming a microprocessor and programming an FPGA device?

Our objectives when teaching *Programmable Logic Design* ("PLD") are hence:

- Students understand that PLD is about designing hardware, and *not* about programming it.
- Students understand and can execute steps in digital design.
- Students are able to design and build a digital system using VHDL and other FPGA techniques, and then implement it on an Altera DE1 board.

Since digital design using FPGA can be much effortless compared to "74xx" approach, a number of more-technical objectives are also imposed. They include:

- Students understand the difference between combinatorial logic and sequential logic, and can construct both circuits using VHDL.
- Students can implement complex counter on Altera DE1 board using VHDL.
- Students understand and can build a finite state-machine using VHDL.

Engineering skills, most of the times, can be fully understood only when one is involved in a hands-on practice. This project has to be sufficiently complex so the students will be encouraged and intellectually challenged; but it should not be too big that the students would be overwhelmed and discouraged.

In the next two sections we will discuss the content of the subject, including the project assignment.

III. Syllabus and Structure of Delivery

Since students taking the subject have already had some background knowledge of digital systems, a lecturer can principally skip some topics already covered in the first or second year. However, in teaching this subject we try to rehearse some fundamental aspects of digital systems in the first few lectures. This is done with two objectives in mind: to reinforce the foundation and to establish the hardware-design mindset.

The course attempts to cover the following materials (shown here in the order of delivery)

- Principles of digital systems (sampling and quantization of analog signals, logic levels, binary and hexadecimal number system, digital waveforms)
- Logic gates (basic and derived logic functions, DeMorgan theorems, gate equivalence, enable and inhibit properties of logic gates)
- Combinatorial logic (boolean algebra, simplification of SOP/POS expression, K-map, simplification by DeMorgan equivalent gates, universal properties of NAND and NOR gates.)
- Introduction to VHDL and Quartus II (Altera's Quartus II, design flow in Quartus II, block diagram file, VHDL basics, hierarchical design).

(from this point onward, most lectures will contain examples in VHDL and block diagram file)

- Combinatorial logic functions (decoders, encoders, multiplexers, demultiplexers)
- Digital arithmetic and its circuits (signed binary numbers: 2's complement, signed binary arithmetic, hexadecimal arithmetic, binary adders and subtractors)
- Sequential logic (latches, NAND/NOR latches, Gated latches, edge-triggered D flip-flops, JK flip-flops, flip-flops in FPGA)
- Counters and shift registers (synchronous counters, binary counters for FPGA, control options for synchronous counters, presettable and bidirectional counters, shift registers).
- State machine design (state machines with no control inputs, state machines with control input,).
- Memory device and systems (RAM, ROM, dynamic RAM modules)

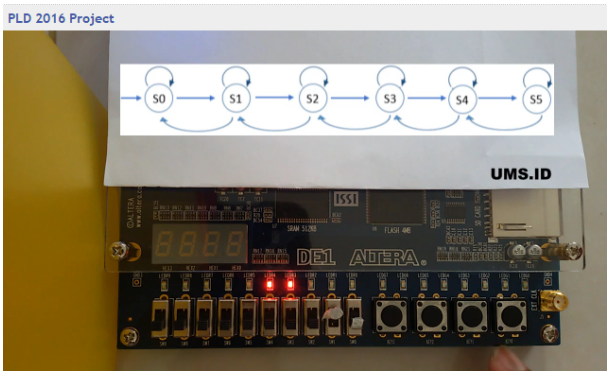
The main reference for the above is the excellent textbook by Dueck [16].

Week	Topic	Lab./Assignment
1	Princ. of digital systems	
2	Logic gates	
3	Combinatorial logic	
4	Quartus II and VHDL	Lab. session starts (every week onwards)
5	Comb. logic funtions	Assignment 1 out
6	Comb. logic funtions	Assignment 1 due
Mid semester examination		
7	Digital arithmetic	
8	Sequential logic	Assignment 2 out
9	Sequential logic	
10	Counters, shift regs.	Assignment 2 due
11	State machine	Project specification out
12	State machine	Mid-report due
13	Memory systems	Full report due. Project demo.
Final semester examination		

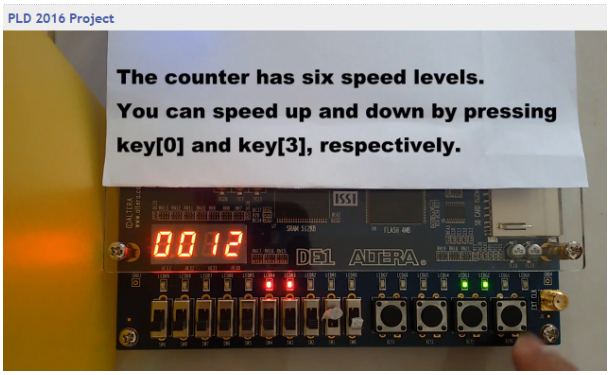
Fig. 2: Typical delivery schedule in this course. Each lecture is about 100 minutes, and a laboratory session –starting at week 4– takes about 120 minutes. For more details of each topic, see text.

As noted above, in this course basic principles are rehearsed with an emphasis on using a hardware description language. Figure 2 depicts schedule in this course. Starting from lecture 4, VHDL examples are presented extensively in every lecture onwards where appropriate.

Laboratory session starts from week 4, where students will have hands-on experience of FPGA design. Lab sessions proceed until end of semester before student vacation week. With typically 25 students per semester taking the course, our lab has four Altera DE1 boards [45]. See also Figure 3.



(a) Describing possible states



(b) Describing some functionalities

Fig. 3: Two screen-shots of the video describing the project. A link to it is provided in the text.

IV. Project Assignment

A. Administration

At about week 11, an FPGA project is assigned to the students. Typically, we first explain the project's objectives in the class accompanied by a video of the final product expected from the students. A website dedicated to the project is made available where students can watch the video, download initial code, and access links to more resources on the Internet. For 2016 class, the project website is:
`http://fajar-suryawan.sites.ums.ac.id/pld-2016` .
Figure 3 shows two screen-shots of the video.

The project requires student to work with specific terms and conditions, which include

- The team should consist of 2 or 3 members: no more, no less. This is to encourage team work.
- The top-level-entity must be a .bdf file. The lower level files can be in BDF or VHDL (or Verilog or SystemVerilog if they want). BDF as top level entity will make project demonstrations easier and hopefully drive students to visualize the big picture.
- The report should contain
  - explanation for the design,
  - print-out of the design files,
  - number of hours they put into the work,
  - list of contributions. That is, "who has done what" (including individuals outside the team). This is to discourage students from being "free-rider" and encourage them to contribute as much as possible.
- Students may (but don't have to) use the files provided as a starting point.

Students are allowed access to laboratory to work on the project. A week after project explanation, a mid-report is due which contains state-machine design. A week later, a final report is to be submitted. No change is allowed to the final report. One or two days after report submission, project demonstration is performed in the laboratory. Each and every team presents their work with 15 minutes duration to the lecturer.

B. Content

Typical project, so far, contains works on LEDs and seven-segment displays in the DE1 board, with the push-button and switch as the input. Internally, it involves designs of concepts such as counter, state machine, flip-flop, interface, and other combinatorial/sequential logic element.

In every project, students are principally asked to design mealy-type state machine with inputs and outputs mentioned above. Figure 4 illustrates the mealy type state machine in the students' project with its inputs and outputs.

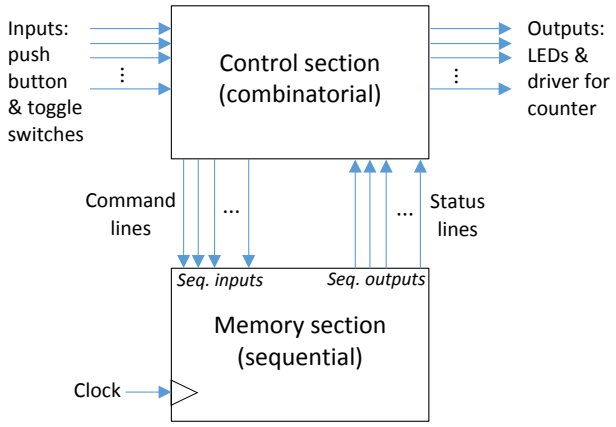


Fig. 4: State machine of mealy-type is at the heart of the design in the student project.

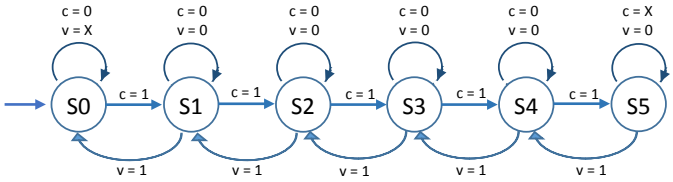


Fig. 5: State transition diagram of the student project, to be implemented using VHDL.

In the lecture session where we deliver the project specification, state machine diagram is explained before the students (see Figure 5 for an example). Students are expected to adhere to this state machine diagram and built it using VHDL.

To help students starting their works, some initial files are made available in the project's website. These files are stripped-down version of their functional counterparts. Students *may* use the files, but they *don't have to*. In fact, there were teams that had *better* design than the initial files.

C. Example student project

In 2016 we assigned project to students in which they are to build counter that has six speed levels: from speed 0 (not counting) to speed 5 (fastest). The counter is sped up one step at a time using a push button switch, and sped down using another one. If the counter does not receive any input, it will stay in that speed level.

These speed levels are the states, depicted in Figure 5. The initial state is S0. There is one input ("speed up") to advance from S0 to S1, up to S5. To move the state the other way, another input ("speed down") is used.

Outside the state-transition logic, the counter can also be paused and reset. A lapse toggle switch is also specified: to "freeze" the display while the counting continues. The counter should also be able to change direction, commanded by a toggle switch. These inputs and outputs in the Altera DE1 board are depicted in Figure 6.

The 10 red LEDs are used to indicate the speed level (that is, the state) and the direction. The 5 rightmost red LEDs are

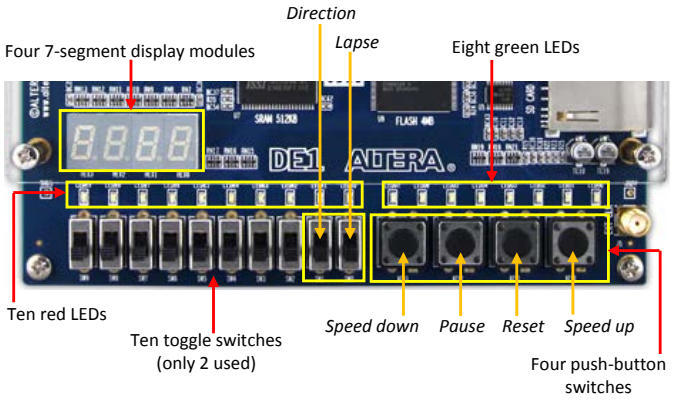


Fig. 6: DE1 inputs and outputs used in the student project.

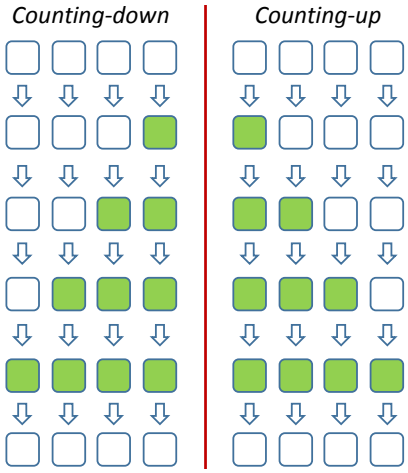


Fig. 7: Greed LEDs flashing sequence. Right half is for 'counting-up' and the pattern moves –in loop– faster and faster as the state goes from S1 to S5. Left half is the flashing sequence for 'counting-down' with similar explanation about the speed.

used indicate 'counting-up'. Starting from the innermost LED going rightwards, if state is S1 then one LED is on. If state is S2, two LEDs is on. And so on until S5.

Similar explanation is for the 'counting-down' direction, where the 5 leftmost red LEDs are used in the same way.

The green LEDs are used to show blinking lights with specific sequence pattern similar to turn-sign tail light of Ford Thunderbird, which is shown in Figure 7.

The set of four 7-segment display modules is used to show decimal number indicating the counter's current value. When counting up, it shows "0000" to "9999" and then rollovers, and vice versa when counting down.

D. Exploration challenge

The projects sometimes involve notions not explicitly taught in the class. This is done deliberately since students are expected to be able to explore more advanced techniques on their own. However, to help the students – especially the less experienced ones – we disclose the challenges involved, and

give hints on how to address those challenges. An example follows.

In the student project above, the state machine design is such that the transition between those six states is controlled by only two inputs (see Figure 5). This might pose a challenge since if the state machine is implemented naively, it may not work as expected. This is due to single input that drives consecutive state transition, which will lead to "fall-through" phenomenon. For example, when the current state is S0 and one wants to advance to S1, he would set the corresponding input. But instead of advancing one state only, the state "falls-through" to S5, the last state.

We told students, in the beginning of project assignment, that this problem will occur if they implement it as it is. We then gave hints on how to address it.

V. Discussion and Conclusion

A. Lessons learned

This course is always in continuous improvement, sources of which include students' feedback, students' results, technological advances, and input from peers.

From our experience, students do not immediately grasp the idea of certain notions, especially the more abstract ones like state-machine. Three to two year ago, most student design reports did not include any state machine design; just a contrivance of sequential circuitry. While the specifications are met, it was very difficult to extend and generalize from – and it certainly does not accord to the course objectives.

This sad truth led us to change the project assignment strategy. When this course was first offered in 2013, the full report was due two weeks after the assignment, without any mid-report. In a hindsight, this might hinder students from carefully planning their design since there is no feedback and the design was a one-shot attempt. Beginning in 2016, we changed the reporting scheme. Students are required to submit an initial design, particularly the state-machine part. We expected that students are forced to carefully design the state-machine. Much to our delight, a couple of teams managed to craft fully working products using proper state machine design.

We learned that ***students need deliberate guidance – in this case a mid-report submission for a specific design – when challenged with relatively complex notion.***

As discussed in the previous section, students are asked to do a research to tackle some engineering challenges not explicitly addressed in lecture sessions. The solution to these challenges are expected to be not just coldly carried out but well justified and explained. Some teams only copied from the Internet – they could not explain the inner-workings of their solution. But better teams indeed learned a lot by exploring solutions and could easily explain the solution.

We conclude that ***students learn a lot if, in addition, the project involves concepts not explicitly taught and they are forced to explore for solutions.*** However, ***a lecturer must gauge students' capability and if necessary give hints regarding the challenge.***

As an added bonus, this challenge also makes grading the report easier, since it highly discriminates good and bad teams. Good designs in this project assignment are generally lean, easy to follow, efficient, and comply with best practices. Figure 8 shows a final .bdf top-level entity by one of our best teams. On average, each team needs 40 hours to complete the project.

Feedback from students have been encouraging too. One of them stated that this course has motivated him to pursue career in IC design.

B. Future improvement

This course attempts to familiarize students to modern technique in digital design. The project in 2013 started simple. As the years progress, student clubs (e.g., Student Robotics Club) will already have knowledge-base of FPGA, and there will be transfer of knowledge from one class to their younger colleagues. This, and advances in the industrial world, necessitates more complex project assignments. Future projects will involve, among others: more inputs and outputs, more applications, and more engineering challenges.

C. Conclusion

We presented out experience in teaching a digital-design course utilizing FPGA technology, which enables an accessible practical approach. We believe that learning-by-doing – by building hardware project – is the best approach for the students to gain skills and confidence. Gibbs in his book [20] often uses the *experience* → *reflection* → *conceptualization* → *experimentation* →(goes back to)→ *experience* cycle to illustrate this learning process. In improving our methods, we try our best to be a critically reflective teacher to best benefit the students [9].

Acknowledgment

This work was partially funded by Universitas Muhammadiyah Surakarta through its Doctoral Grant program, grant number 216.8/A3-III/LPPM/X/2013. The author would like to thank PLD lab. assistants and technicians. The author is grateful for the students in all the PLD classes.

References

[1] A handbook for teaching and learning in higher education: Enhancing academic practice, 2015.

[2] U. Alqasemi, H. Li, A. Aguirre, and Q. Zhu. FPGA-based reconfigurable processor for ultrafast interlaced ultrasound and photoacoustic imaging. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 59(7):1344–1353, 2012.

[3] A. Amira and S. Chandrasekaran. Power modeling and efficient FPGA implementation of FHT for signal processing. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 15(3):286–295, 2007.

[4] P. J. Ashenden. *Digital Design: An Embedded Systems Approach Using VHDL*. Morgan Kaufmann, 2008.

[5] T. Atalik, M. Deniz, E. Koc, C. Gercek, B. Gultekin, M. Ermis, and I. Cadirci. Multi-DSP and -FPGA-based fully digital control system for cascaded multilevel converters used in FACTS applications. *Industrial Informatics, IEEE Transactions on*, 8(3):511–527, 2012.

[6] P. Athanas, D. Pnevmatikatos, and N. Sklavos. *Embedded Systems Design with FPGAs*. Springer, 2013.

[7] D. G. Bailey. *Design for Embedded Image Processing on FPGAs*. IEEE, 2011.

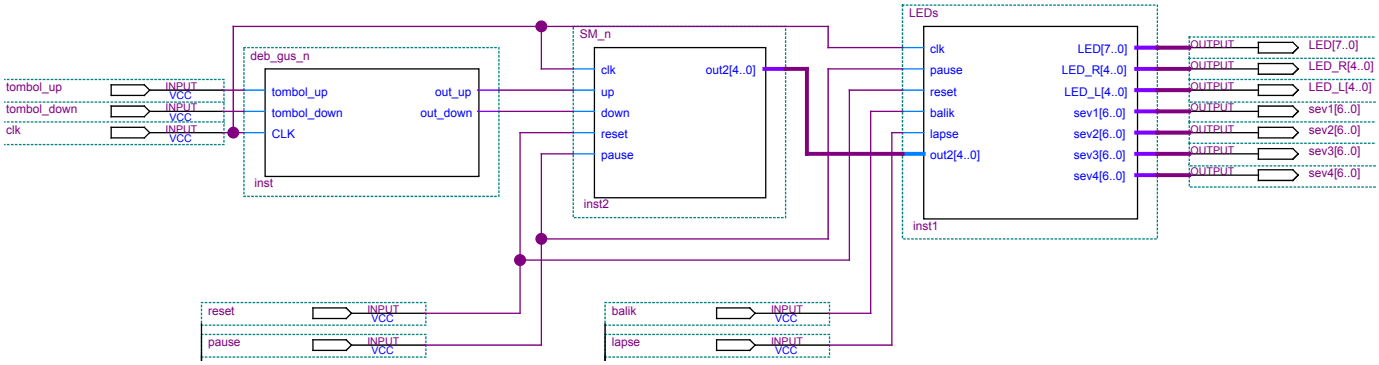


Fig. 8: Top level entity of one of our best student teams. Inside the blocks are hundreds of lines of vhdl code.

[8] L. E. M. Brackenbury, L. Plana, and J. Pepper. System-on-chip design and implementation. *Education, IEEE Transactions on*, 53(2):272–281, 2010.

[9] S. D. Brookfield. *Becoming a Critically Reflective Teacher*. John Wiley & Sons, 2^{nd} edition, 2017.

[10] H. Caner, H. Gecim, and A. Alkar. Efficient embedded neural-network-based license plate recognition system. *Vehicular Technology, IEEE Transactions on*, 57(5):2675–2683, 2008.

[11] W. Chelton and M. Benaissa. Fast elliptic curve cryptography on fpga. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(2):198–205, 2008.

[12] J. U. Cho, Q. N. Le, and J. W. Jeon. An FPGA-based multiple-axis motion control chip. *Industrial Electronics, IEEE Transactions on*, 56(3):856–870, 2009.

[13] J.-P. Deschamps, G. D. Sutter, and E. Cantó. *Guide to FPGA Implementation of Arithmetic Functions*. Springer, 2012.

[14] J. Diaz, E. Ros, R. Carrillo, and A. Prieto. Real-time system for high-image resolution disparity estimation. *Image Processing, IEEE Transactions on*, 16(1):280–285, 2007.

[15] R. Dubey. *Introduction to Embedded System Design Using Field Programmable Gate Arrays*. Springer, 2009.

[16] R. Dueck. *Digital Design with CPLD Applications and VHDL*. Cengage Learning, 2^{nd} edition, 2011.

[17] C. Erdogan, I. Myderrizi, and S. Minaei. FPGA implementation of BASK-BFSK-BPSK digital modulators [testing ourselves]. *Antennas and Propagation Magazine, IEEE*, 54(2):262–269, 2012.

[18] R. Ghosh and G. Narayanan. Control of three-phase, four-wire pwm rectifier. *Power Electronics, IEEE Transactions on*, 23(1):96–106, 2008.

[19] G. Gibb, J. Lockwood, J. Naous, P. Hartke, and N. McKeown. NetFPGA – an open platform for teaching how to build gigabit-rate network switches and routers. *Education, IEEE Transactions on*, 51(3):364–369, 2008.

[20] G. Gibbs. *Learning by doing: A guide to teaching and learning methods*. Oxford Centre for Staff and Learning Development, Oxford Brookes University, 1988.

[21] M. Gokhale and P. S. Graham. *Reconfigurable Computing: Accelerating Computation with Field-Programmable Gate Arrays*. Springer, 2005.

[22] I. Grout. *Digital Systems Design with FPGAs and CPLDs*. Newnes, 2008.

[23] D. M. Harris and S. L. Harris. *Digital Design and Computer Architecture*. Morgan Kaufmann, 2^{nd} edition, 2013.

[24] J. L. Hennessy and D. A. Patterson. *Computer Architecture: a Quantitative Approach*. Morgan Kaufmann, 5^{th} edition, 2012.

[25] S. Himavathi, D. Anitha, and A. Muthuramalingam. Feedforward neural network implementation in FPGA using layer multiplexing for effective resource utilization. *Neural Networks, IEEE Transactions on*, 18(3):880–888, 2007.

[26] E. O. Hwang. *Digital Logic and Microprocessor Design with VHDL*. Cengage Learning, 2005.

[27] S. T. Karris. *Digital Circuit Analysis and Design with* SIMULINK® *Modeling and Introduction to CPLDs and FPGAs*. Orchard Publications, 2^{nd} edition, 2007.

[28] C. Kellett. A project-based learning approach to programmable logic design and computer architecture. *Education, IEEE Transactions on*, 55(3):378–383, 2012.

[29] W. Kleitz. *Digital Electronics: a Practical Approach with VHDL*. Pearson, 9^{th} edition, 2012.

[30] J. H. Lee, S. E. Lee, H.-C. Yu, and T. Suh. Pipelined CPU design with FPGA in teaching computer architecture. *Education, IEEE Transactions on*, 55(3):341–348, 2012.

[31] S. Lee. *Advanced Digital Logic Design: Using VHDL, State Machines, and Synthesis for FPGAs*. Thomson, 2006.

[32] T. Li and Y. Fujimoto. Control system with high-speed and real-time communication links. *Industrial Electronics, IEEE Transactions on*, 55(4):1548–1557, 2008.

[33] O. Lopez, J. Alvarez, J. Doval-Gandoy, F. Freijedo, A. Nogueiras, A. Lago, and C. Penalver. Comparison of the FPGA implementation of two multilevel space vector PWM algorithms. *Industrial Electronics, IEEE Transactions on*, 55(4):1537–1547, 2008.

[34] C. Maxfield. *The Design WarriorŠs Guide to FPGAs: Devices, Tools, and Flows*. Newnes, 2004.

[35] P. Meher, S. Chandrasekaran, and A. Amira. FPGA realization of FIR filters by efficient and flexible systolization using distributed arithmetic. *Signal Processing, IEEE Transactions on*, 56(7):3009–3017, 2008.

[36] U. Meyer-Baese. *Digital Signal Processing with Field Programmable Gate Arrays*. Signals and Communication Technology. Springer, 3^{rd} edition, 2007.

[37] E. Monmasson, L. Idkhajine, and M.-w. Naouar. FPGA-based controllers. *Industrial Electronics Magazine, IEEE*, 5(1):14–26, 2011.

[38] A. Moradewicz and M. Kazmierkowski. Contactless energy transfer system with FPGA-controlled resonant converter. *Industrial Electronics, IEEE Transactions on*, 57(9):3181–3190, 2010.

[39] M.-w. Naouar, A. Naassani, E. Monmasson, and I. Slama-Belkhodja. FPGA-based predictive current controller for synchronous machine speed drive. *Power Electronics, IEEE Transactions on*, 23(4):2115–2126, 2008.

[40] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: the Hardware/Software Interface*. Morgan Kaufmann, 3^{rd} edition, 2005.

[41] V. A. Pedroni. *Circuit Design and Simulation with VHDL*. MIT Press, 2^{nd} edition, 2010.

[42] R. S. Sandige and M. L. Sandige. *Fundamentals of Digital and Computer Design with VHDL*. McGraw-Hill, 2012.

[43] N. Sudha and A. Mohan. Hardware-efficient image-based robotic path planning in a dynamic environment and its FPGA implementation. *Industrial Electronics, IEEE Transactions on*, 58(5):1907–1920, 2011.

[44] T. Sutikno, M. Facta, and G. A. Markadeh. Progress in artificial intelligence techniques: from brain to emotion. *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 9(2):201–202, 2013.

[45] Terasic Incorporation. *Altera DE1 Board*, `de1.terasic.com`, accessed July 2017.

[46] A. Wills, G. Knagge, and B. Ninness. Fast linear model predictive control via custom integrated circuit architecture. *Control Systems Technology, IEEE Transactions on*, 20(1):59–71, 2012.

[47] L. Zhuo and V. Prasanna. High-performance designs for linear algebra operations on reconfigurable hardware. *Computers, IEEE Transactions on*, 57(8):1057–1071, 2008.