Australian National University

FPGA Assignment: Keyboard interfacing and text display on a Digilent Nexys4-DDR or Basys3 board

ENGN3213/6213 Digital Systems and Microprocessors

Semester 1, 2019

ANU College of Engineering and Computer Science

TABLE OF CONTENTS

1. PROJECT STATEMENT

You are required to create a ***digital design to turn your FPGA development board into a simple user interface which allows a user (after having read this manual) to perform a few basic tasks involving external interfacing with a standard USB keyboard***. The requirements are as follows:

Your system should be able to receive a signal from a USB keyboard connected to the HID/emulated PS2 interface which is provided as a standard on the Nexys4DDR and Basys3 boards. It will also successfully utilise a range of available on-board I/O options, including slide switches, push buttons and 7-segment displays (7SDs), to provide meaningful user interfacing.

The overall aim of the project is to create a system capable of three functions of increasing complexity. The functions are listed below:

2.1 Function 1: single character output to 7SD

Your system should be able to detect, decode and display a character input from an external USB keyboard. Additional information regarding the relevant communication protocol as implemented on the Basys3 and Nexys4DDR boards is provided later on in the document.
User interface requirements for Function 1:

- pressing of a key on the keyboard causes the leftmost panel of the 7SD to show the character that has been pressed. The system must be able to recognise and display at least the following letters:
  - Vowels (A E I O U);
  - Consonants (b C d F G H L n P r S t)

  Please note that in order to render letters on the 7SDs in a clear and nonambiguous way, a mix of uppercase ad lowercase output will be required. You do not need to cater for the shift key.
- Pressing a subsequent key on the keyboard causes for the displayed character to be replaced with the new selection if the key is recognised, or for the 7SD to become blank if the key is not one of the recognised ones.
- A slide switch (your choice) enables/disables the update of the display as new keypresses occur.
- A further slide switch turns off the 7SD output.

2.2 Function 2: recall and display of predefined sentences

As the next feature, your system should be able to recall and display *at least* three sentences from given sets (see Appendix 1). A sentence comprises of up to 8 words, which can be shown one at a time on the display. Each word can be up to 8 characters (Nexys4) or up to 4 characters (Basys3) in length depending on the board used for the implementation.
User interface requirements for Function 2:

- Words are displayed on the 7SDs
- *Buttons East and West* on the board AND, alternatively, *Right/Left arrow keys* on the keyboard, should switch between words in one sentence
- *Buttons North and South* on the board AND, alternatively, *Up/Down arrow keys* on the keyboard, should switch between sentences.

- One of the slide-switches (your choice) turns on “auto-read” where the words of the current sentence are displayed one after another, automatically switching every 1s approximately, in a loop.
- One of the slide (your choice) switches off the 7SD output.
- You should use LED outputs in a way of your choosing to inform the user of the word/sentence being displayed (e.g., word 3 in sentence 1, etc.)

2.3 Function 3: word editing and storage

The most difficult part of the work. Your system should allow you to type, display and store a sentence of up to 8 words, with each word containing up to 8 letters (Nexys4) or 4 letters (Basys3) from the available set of letters. Individual words can also be recalled and modified.
User interface requirements for Function 3:

- *Buttons East and West* on the board AND, alternatively, *Right/Left arrow keys* on the keyboard, should switch between words for the custom sentence (as in Function 2)
- LED output indicates the word currently in use (also similar to Function 2)
- The 7SDs display the currently selected word (also similar to Function 2)
- When a word is displayed, it can be modified by typing additional letters to it (up to the maximum 4/8 letters) and/or by deleting existing letters by pressing *Backspace* on the keyboard.
- Pressing the *North* pushbutton on the board stores any changes made to a word. If changes are not saved, shifting away from that word will cause changes to be lost.
- (additional) Once you have combined the design (see function 4), you may be able to allow for a pre-defined sentence to be loaded by pressure of the *South* button.

2.4 Function 4, all functions working together

The three functions can be implemented and submitted as separate designs. However, you will get ***extra marks*** *for creating a single machine which can switch between the different functions*. Since most of your buttons are taken up by function-specific requirements, you can consider using the centre button to switch between modes, though this is not a strict requirement.
You are required to inform the user of which function is in use. This can also be done in a way of your choosing. I would recommend using your LED array to provide an informative LED output. However, if all of your LED array is already committed, you may consider using the tri-colour LEDs or some other alternative.
As you put all features together in the same machine, you will be able to complete the design by introducing the possibility of loading a pre-written sentence into Function 3 amongst those available for Function 2 so that it can be used as a basis for editing.

2. RECOMMENDED WORKFLOW

As you will have noticed, the assignment is defined in terms of levels of increasing difficulty. The following recommendations *are not prescriptive* but they might help you find your way through to the end of the project:

1. **Find a partner and register** your group on Wattle; **investigate Open Lab** options if you do not have access to a board of your own and reserve some time slots if required.

2. **Read the helpful <u>TIPS</u> section** located towards the end of this document if you want some practical suggestions
3. **Get to work!** Function 1 is core to the design and is the most valuable in terms of marks so you may want to start from there. If you prefer, however, you could start on Function 2 as a lot of its specifications do not involve keyboard input (if you choose to do this, skip to step 5, though do come back here afterwards!). You can probably do most of this design in simulations before you move on to the hardware.
    - Simulations might be easier at submodule level as this allows for simpler test benches. Once you are certain that your submodules run as intended, assembling them in a top module should be much easier to do (and troubleshoot).
4. Once you have tested and confirmed that you can receive and display characters received by the keyboard, write some clear documentation for this core part of your design.
5. Develop a solution for function 2. Where possible, do not double-up on work you have already done for Function 1.
6. Consider the requirements of function 4 and create a state machine which can toggle between the first two functions and operate them independently.
7. Test all the work done so far in hardware.
    - *If it all works, and if you <u>clearly document your work</u>, this constitutes a credit level for the assignment*
8. Then, develop function 3.
9. Depending on your level of confidence in your skills and/or on how much time you have got you can at this stage decide to integrate function 3 into the overall machine or leave it as a standalone application
10. Integrate all functions so that full operation of the system can be coordinated by one overarching state machine. Integration may allow you to include the additional loading option for function 3 if you feel comfortable with your workflow so far.

These are just suggestions. You may decide to work differently.

3. DELIVERABLES AND ASSESSMENT

3.1. Assessment process

For this project **students will work in groups of 2**. I have recommended that you work with your lab partner but this is not a strict requirement. The work will be assessed as group work, with the same mark awarded to both members. There are two exceptions to this:

- where one of the group members has not contributed to the project in a fair manner. Given that there is no group presentation, the quality of team work will be tested through an honesty system based on individual declarations (more on this later);
- for *Masters students* there will be an <u>additional reflective question on digital design which must be worked on individually and will be individually marked.</u>

The assessment of the work will be done by <u>testing the design implementation</u> in hardware and using the <u>documentation</u> as support to ensure that all design choices have been well thought out and justified. Testing will probably be conducted during weeks 8 - 10.

Written project document

Your delivered project document should give a *<u>clear and concise but complete</u> presentation of your work*. You should imagine that your document is to serve as a comprehensive technical manual to be read by a technically-knowledgeable user (the lecturer/tutors know a lot about electronics but nothing about your design, unless you tell them). At a minimum your document should include:

- a clear description of the system's user interface (how can a user operate your design: what inputs need to be asserted to perform operations and what outputs are expected)
- a detailed technical description of its structure and operation, i.e., how the inner workings of your design allow it to display the required behaviour. The following are required:
    - a block diagram at submodule level, where by submodule I intend a functional subunit of your design. You should not show every logic gate or flip-flop in your design, but I also do not want to see too many "magic" black boxes in your circuit diagram whose operation is mysterious (e.g., a "clockdivider" block clearly identifies a common digital design and does not require much more to be said, while a block called "letter generator" is insufficiently specified and would require a much more detailed explanation).
    - where you have used state machines, a state transition diagram (and/or tables if they provide additional information) is essential, along with an explanation of your choice of states. Next state or output logic equations may or may not be particularly significant depending on your design choices.
    - motivation of your design choices (<u>nothing too verbose</u>: it should be a commentary on your block diagrams, and state diagrams where applicable. For example: *function xx is enacted through the use of a down-counter enabled by signal YY. This solution was chosen as it reuses the same bus for multiple purposes, thereby reducing hardware resource usage.*)

Whatever you do, *do not tell a story* about how you worked through your assignment. Your reference document for the style of your "paper" deliverables should be <u>a technical data sheet</u>. You will be provided with some examples of good quality documentation from past years. It is not expected for the final document to exceed 8-10 pages. (Please note: there is no hard limit since this report may be figure-rich, making total length unpredictable. However, overlong documents may be penalised on clarity grounds).

You can organise your document by separate functions if it helps, but remember that, if you have performed any integration of your design, what will be expected is the **technical manual of one complex machine with several functions, not four separate machines.** It is most likely, for example, that some submodules in the design will be shared amongst the various functions and this should be made clear in your documentation as part of your description of the one overarching machine.

Code

<u>Your code will not be marked</u> for programming style, but it *will* be looked at, especially if the report is not sufficiently clear or the project does not work, hence forcing the markers to dig into the program to see what partial marks can be awarded. Please use comments for your and the markers' benefit.

<u>You are required to submit a functioning, compilable version of the Verilog code</u> corresponding to your final implementation as an attachment to your submission. Please ensure that you submit the correct, working version of your code inclusive of the constraint file.

Group work declaration
Regardless of the overall joint submission, every student will submit a one-page group work report. This will be a form where you will describe your own contribution and your partner's. The content of the form will not be disclosed to the other group member. Marks may be added or deducted to team members depending on the descriptions provided. **Failure to submit this component will result in a *penalty*** *of 10%* on the overall mark.

3.2. Due dates

The project submissions are due **before the end of the day (11.55pm) on Monday 29 April** (wk 8)

I will be very reluctant in granting extensions because in Week 8 lab activities resume so there will be very little room for flexible arrangements. The maximum I can envisage is a couple of days *for people with exceptional circumstances*.

3.3. Submission process

The project submissions will be done through Wattle.

One submission per group is sufficient, though each group member may want to upload their group work declarations individually for confidentiality reasons. For masters students, the additional question will also need to be submitted individually.

It is strongly recommended that **you re-download your own submission after uploading it to wattle** in order to check for completeness and version accuracy before you finally send it though for marking. A failure to upload the correct files will not be a ground for appeal of a poor mark.

3.4. Marking Criteria

The project will be marked as follows:

ITEM	**WEIGHT**
FUNCTION 1	35%
FUNCTION 2	23%
FUNCTION 3	28%
LEVEL 4	6% if two functions operate under the same state machine 12% if three functions operate under the same state machine 14% if three functions operate under the same state machine and any F2 sentence can be loaded into F3
EXTRA QUESTION **Master students only**	11% (Master students are marked out of 111% and then scaled to 100%)

For each component of the design project, the following make up the partial mark:

ASSESSMENT CRITERION		**WEIGHT**
Clarity of written documentation	Have the students provided documents which clearly show their design approach, the design structure and the features of their system? Does the document allow the markers to gain a clear, comprehensive understanding of the design without having to refer to the fine details of the Verilog code?	30%

Quality of design	Is the design structured in a clear and logical manner? (e.g., are sequential and combinational logic sections appropriately recruited and clearly separated; are submodules used appropriately to isolate specific functions; have states been assigned in a logical fashion?) Is the design synchronous? Is the likelihood of malfunction due to timing hazards low? Have the designers considered (where applicable) potential issues with dealing with asynchronous, potentially noisy external inputs? Have the designers created a design which is neat in the sense of avoiding the use of excessive hardware resources? Have the designers paid attention, in formulating their design, to incorrect inputs which may be asserted accidentally? Does the design respond to these concerns in a reasonable manner? (i.e. where accidental inputs are not considered, why has it been deemed safe to do so? When unwanted inputs may create ambiguity or other operation issues, does the design deal with these appropriately?)	30%
Correct operation	Does the system operate as required?	30%
Teamwork	Has the student participated adequately as a team member in this project?	+/- 10% on the mark gained from the other components

Late delivery of the project work will be penalised in accordance with ANU policy. This includes, but is not limited to, a penalty of 5% per day or part thereof. This project is worth 20% of your total course marks.

Plagiarism will not be tolerated. Make sure you are the author of your own work as ANU has very strict policies against academic misconduct. *You must acknowledge **any** external sources* (if any) you may have referred to in creating your design and specify the extent of the impact of said external resources on your work. This includes providing in-code referencing where this is allowed (see below). Please be aware that our prevention strategies include a check of your work against a broad list of internet resources as well as every submission of a past assignment for ENGN3213/6213 made over the last 7 years.

Use of externally source code is generally not allowed for this assignment. The only exception is code already written or supplied during the 2019 edition of ENGN3213/6213. You are being tested on your skill to conceptualise, write up and implement a digital design and incorporating codes written by others into your work would impede the markers' ability to evaluate those skills by taking away some of the creative effort required for this assignment. For the same reasons, the use of IP cores, though extremely useful for the experienced FPGA engineer, is also not allowed.

4. OPEN-LAB TIMES

In order to facilitate access to equipment, the R103 lab will be open for additional hours where students will be able to go in and work on their designs.

- **week 6** (Tue-Fri when course lab activities are not operating)
- **teaching break week 1** (To be advised on Wattle)
- **teaching break week 2** (To be advised on Wattle)
- **week 7** (9am-5pm every day *except lecture times*)

The lab will have minimum supervision so please be careful and considerate with the equipment. Also please do not bring any food/drink into the lab and make sure you wear enclosed footwear.

Open Lab scheduling will be based on a public calendar. This is done to ensure that every student can get a fair share of lab time. Scheduling information will be posted on Wattle.

A note about overcrowding
We have limited facilities in R103 with 14 computers. *Development boards should not be removed from the lab*. With 80+ groups, overcrowding is a potential concern.
A lot of open lab time has been allowed (at least 22 hours of board time per group). Still, in the interest of this working smoothly, I recommend the following:

- **don't leave your project to the last minute,** the project is long and access to resources is likely to become tighter as the deadline approaches
- Board time is precious!! **Conduct design and simulation work outside the lab**, e.g., on your laptops, whenever possible.
- **strictly keep to the times you have registered your group for**
- **at no time should a group take up more than one workstation in the lab**
- **if a time slot has vacant workstations, people may join on a first-come first-served basis, but people registered should always have precedence (i.e., if a registered group arrives late and finds someone has taken their workstation, they may ask the unauthorised group to leave).**

5. TIPS AND USEFUL INFORMATION

Function 1:
For you to successfully address this function, it is essential that you understand the timing requirements of an incoming PS/2 stream and can capture this accurately with your design. You will need to be clever with combining essential sequential designs to achieve this goal. If it helps, you can perform direct measurements on incoming signals and the response of your design:

- through simulations, by simulating the ON/OFF times of synchronisation and data signals and your system’s response.
- in the lab, by creating a dedicated FPGA implementation which continuously assigns a copy of the input data and selected relevant signals from your system to locations that can be easily probed with an oscilloscope, e.g., PMOD socket pins as you did in the practical labs.

Information on PS/2 emulation for USB keyboards on the Basys3 and Nexys 4 boards is provided by Digilent at the below locations:
https://reference.digilentinc.com/basys3/refmanual#usb_hid_host
https://reference.digilentinc.com/reference/programmable-logic/nexys-4-ddr/reference-manual#usb_hid_host

PS/2 is a form of serial communication. Additional information on serial communications may also be provided in the lectures.

Function 2:
This function is varied in its features but relatively straight forward. Try and keep it neat by separating it into meaningful submodules and look out for opportunities to reuse/repurpose components already designed (either in Function 1 or other practical labs). This section will also test your ability to conceptualise and implement solutions around timing and fixed information storage. Try not to think algorithmically as this poses a risk of hardware overutilization. Consider whether a state machine may be able to help you here.

Function 3:
This section has a focus on user interactions. The challenges here are posed by user-dictated events and by the need for appropriate structures to store the modified word information as it is acted upon in real-time by the user. Enforcing the character limit and storing the final text string in the face of user corrections (through backspace) will pose additional challenges.

Function 4:
If you think of your design in a coherent manner, you should be able to have multiple functions exploit the same submodules. This might save you a lot of hardware resources and will improve your marks.
Have you considered whether the wrong button pressed at the wrong time could cause your system to malfunction?

General suggestions:
Have you debounced your controls where appropriate?
Is your system initialised / returned to a safe state when needed?
Have you avoided inferred latches (incomplete conditional statements in combinational logic)?
Are all of your flip flops running from the same clock? (synchronous machine)
Have you included the system clock in your constraints file?
With the exception of Function 4, you do not have to use state machines if you do not want to. But I can guarantee that there are a number of opportunities to use state machines to make your design neat and easy to implement/troubleshoot. The cleverest of machines may be interpreted as “master machines” which can selectively enable/disable/control other parts of your design.

One final note. **The assignment is designed to be challenging and it is anticipated that not all students will be able to complete all of the work**. That is why it is designed in stages. Completing some sections well is likely to give you a better outcome than submitting all parts half-done. Be ambitious but realistic.
Good luck!

6. ADDITIONAL QUESTION <u>FOR MASTER STUDENTS ONLY</u>

Applications requiring interfacing between separate digital systems (this assignment being a simple example of such applications) are characterised by additional complexity associated with the successful communication of data between two systems.
With reference to issues of logic standard compatibility and/or timing compatibility, discuss how this additional complexity presents itself and how it may be mitigated/addressed through appropriate design choices. You may provide a general discussion, however, applied examples would probably benefit the clarity of your answer.

Please keep your answer to a word limit of around 500 words in the interest of conciseness.

APPENDIX 1: SOME SENTENCES

Example sentences for Basys3 implementation (4-letter per word limit)

- `COOL CAtS rULE thE LOnG rEd rOAd`
- `FPGA IS TOO hArd FOr US ALL`
- `PASS thE SALt TO THE IrOn ChEF`
- `thE bIG bEAr AtE thE bLUE SEAL`
- `TED IS Sad tO SEE FrEd In bEd`

- `… or you can make up your own`

Example sentences for Nexys4 implementation (8-letter per word limit)

- `ThE EnGINEEr SAId thAt FPGAs ArE COOL`
- `StROnG rAIL CurrEntS CAUSE LArGE CIrCUIt bUrnOutS`
- `SLEEPInG In LECtUrES IS GOLd FOr tIrED StUdEntS`
- `COLOUrS OF SPrInG TrEES UndEr thE POurInG rAIn`
- `bOLd LEttErS LIGht UP In brIGht rED FOnT`
- `nO OPtIOnS LEFt FOr hOPELESS PUrPLE rACOOnS`

- `… or you can make up your own`