Australian National University

FPGA Project:

Smart-Home Controller using a Basys3 FPGA Development Board

ENGN4213/6213
Digital Systems and Microprocessors

Semester 1, 2023
ANU College of Engineering, Computing and Cybernetics

Table of Contents

1 Project Statement

You are required to create a ***digital design to turn your FPGA development board into a 'Smart-Home' Controller***. A smart-home typically has several automated functions, such as controlling lighting, heating/cooling, appliances, entertainment system, smart speakers, and home security (door access and alarm system) based on inputs from various sensors and user interfaces. In this project, you will implement two of such functions using your Basys3 FPGA Development board.

Since you do not have access to sensors and actuators, you will use the available **16 switches** and **5 push buttons** on the Basys3 board to mimic the **inputs** of sensors or users. You will also use the **16 LEDs** and the **4 Seven-Segment Displays (SSDs)** to show the **outputs**. That is, we assume that we have the digital input signals from sensors which we generate through inputs available on the board. Similarly, we generate output digital signals that can drive devices, but we display/indicate these digital outputs using LEDs and SSDs in an appropriate way.

In this project, your Basys3 board should work as an integrated controller for two smart-home functions: Garage Door and Climate Control. Note that these functions should be able to work together simultaneously.

You can choose to first design and implement the two functions as separate modules. Once they work independently, you can implement the integrated controller of these two functions using a hierarchical design. You are encouraged to be creative and add additional features to your functions, i.e., this is an open-ended project where you will demonstrate your ability to consider a real-world problem, use your imagination and creativity to generate specifications, and use your knowledge of digital design and Verilog to implement your solution with the resources (FPGA board) you have.

1.1 Function I: Secure Garage Door Control

A garage door opener is a motorised device that can open or close the garage door of your home. The intended function is remote-controlled using a pin-code-secured device or a smartphone application.

As the minimum requirement for this module, we need the following operations:

a) If the garage door is *closed*, you need to enter a particular 6-digit pin code and then press the GARAGE_CONTROL button to open the door. This *6-digit pin code* should be the *binary equivalent of the Project Group Number* you selected in Wattle.
   - If the correct pin is entered, the controller will activate the motor to open the door to let the car in or out.
   - If an incorrect pin is entered, the door will remain closed, and an alarm will sound. The alarm can be reset using an ALARM_RESET button. Note that in alarmed condition, no further attempts to open the door are possible until reset.

b) If the garage door is *open* and you press the GARAGE_CONTROL button, the controller

will activate the motor to close the garage door.

c) If you press the GARAGE_CONTROL button while the door is in the motion of opening/closing, then the door should immediately stop such that it is partially open/closed. When you press the GARAGE_CONTROL button again, the motor should reverse the last movement (i.e., completely close or open the door).
d) While the door is closing, if its bumps into any obstacle, a 'COLLISION_SENSOR' will be triggered. Upon this trigger, the door should immediately stop and reopen. This is to stop the door from crushing your car bonnet/ boot if you did not park correctly.

For the Secure Garage Door automation function, we suggest the following I/O operations:

Inputs:

- One push button as the GARAGE_CONTROL
- Six switches to input the 6-digit pin code.
- One push button as the ALARM_RESET
- One switch as the COLLISION_SENSOR
- (Optional) One push button as the IDLE_RESET to reset the control function to an idle closed-door state.

Outputs:

- 4 LEDs to indicate the status of the door
  - At the start (idle) state, the door is *closed* – indicated by 4 LEDs switched ON.
  - After the correct pin code entry and pressing of GARAGE_CONTROL, the door opening can be indicated by shifting LEDs (say LEDs turn OFF one by one at every 1 second) such that all LEDs turn OFF when the door is completely *open*.
  - If a wrong pin code is entered, the alarm state can be indicated by blinking all 4 LEDs every 0.5 seconds. Once alarm is reset, the blinking stops, and 4 LEDs remain ON.
  - When the door is open (4 LEDs OFF), and you press GARAGE_CONTROL, the door starts closing which can be indicated by the reverse shifting of the LEDs (say LEDs turn ON one by one every 1 second) such that all four LEDs turn ON when the door is completely *closed*.
  - If GARAGE_CONTROL is pressed during the opening/ closing sequence (LED shifting), then the door should stop its motion and be stuck partially open, i.e., LEDs should stop shifting. Pressing the GARAGE_CONTROL button again should reverse the direction, and LEDs should reverse the shifting direction accordingly.
  - If COLLISION_SENSOR is triggered during the door closing, the door should immediately stop and reopen, i.e., LED shifting should stop and reverse the shifting direction until all LEDs are OFF (door open).

Advance Features:

- A secure pin code entry can be used instead of a button to reset the alarm.
- Two consecutive SSDs to display the status of the door. For example, 'DO' means Door Open; 'DC' means Door Close; 'CL' means Collision triggered; 'AL' means Alarm triggered.

1.2 Function II: Climate Control (Home Heating and Cooling)

The climate control function automates the home heating/cooling system depending on the current room temperature and the desired temperature level.

As the minimum requirement for this module, we need the following operations:

a) Once the climate control is ON
   - If the current temperature > desired temperature, the system should cool the room.
   - If the current temperature < desired temperature, the system should heat the room.
   - Heating/cooling the room changes the temperature at the rate of 1° *per 1 seconds.*

b) Once the climate control is OFF
   - Room temperature decrease/increase towards the outside temperature at the rate of 1° *per 3 seconds*

c) The current room temperature should be continuously displayed.

Assumptions:

- Possible temperature range is between 20-27 degrees (i.e., a range of 8 degrees).
- Outside temperature is given as an input. The initial room temperature will equal the outside temperature.

Inputs:

- One switch to switch ON/OFF the climate control.
- Three switches to input the outside temperature.
- Three switches to input the desired temperature.

  Note: Temperature input from three switches can be interpreted as an integer between 0 and 7 such that 0 indicates 20 degrees and 7 indicates 27 degrees.

Outputs:

- Two SSDs to display the current room temperature (between 20 to 27).

Advance Features:

- Use a fan speed input to control the rate of temperature change during heating/cooling.
- If the garage door is open during heating/cooling, the rate of temperature change should be 1° *per 2 second*.

1.3 Requirements of the Project

Part 1: You should **design Functions I and II as separate modules** using Verilog Hardware Description Language (HDL). Before designing, clearly define the assumptions, specifications, inputs, outputs, and intended operations. Use your knowledge from this course to design the system via Finite State Machines (FSMs) and/or combinational circuits as is most appropriate.

Part 2: You should use a **hierarchical structure to design an Integrated Smart-Home Controller consisting of Functions I and II**, i.e., ***both functions should be active simultaneously***. For instance, you should be able to open/close the garage door while your climate control is heating/cooling your room.

- You can utilise a range of I/O options available on your Basys3 board, including slide switches, push buttons, LEDs, and 7-segment displays (SSDs) to provide a meaningful user interface to your smart-home controller. For overall control of your smart-home system, you can use a MASTER_CONTROL input.
- You have the flexibility to change the input/output choices and operations according to your design with proper justifications *subject to the condition that the minimal functional requirements are satisfied*.
- You have the freedom to implement advance features other than the above-mentioned ideas. Be creative with your design. You may be awarded bonus marks for being clever and exceeding expectations.
- We strongly encourage to use FSMs for at least one of the functions.

2 Additional Question (for ENGN6213 Students Only)

Applications requiring interfacing between separate digital systems are characterised by additional complexity associated with successfully communicating data between two systems.

With reference to issues of logic standard compatibility and timing compatibility, discuss how this additional complexity presents itself and how it may be mitigated or addressed through appropriate design choices. Additionally, discuss the impact of power consumption and resource utilisation on the FPGA design and how these factors can be optimised in the design process.

You may provide a general discussion. However, applied examples would probably benefit the clarity of your answer. Please keep your answer to a limit of around 750 words in the interest of conciseness.

3 Recommended Workflow

You can complete the project step by step, starting with a relatively straightforward module and adding more modules with increasing difficulty and complexity. Apply your knowledge from the FPGA labs to implement hierarchical designs controlled by different clock signals and constructs.

The following recommendations are not prescriptive but suggestions that may aid you in completing the project. You are free to work as you prefer.

1. **Find a partner and register** your group on Wattle: (https://wattlecourses.anu.edu.au/mod/groupselect/view.php?id=2799281).
2. If you and your partner do not have access to a Vivado installed computer, you can access the desktops in Ian Ross R103 lab during the teaching break by booking in Wattle: https://wattlecourses.anu.edu.au/mod/groupselect/view.php?id=2871561. We strongly recommend that each group has access to Vivado outside of the R103 labs.
3. **Get to work**! It is a good idea to map out the input/outputs of both functions, including their sub-modules, in advance and plan how they will work together. This will make Part-2 much easier when you combine the functions. Drawing out a block diagram of the

system with sub-modules and their interconnections can aid this step.

4. Try to implement one of the simple sub-modules. You can reuse some of the code you developed throughout your labs and examples from the lectures. You can search the web for helpful hints, but make sure to reference others' work with proper citation details.
5. Use simulations in Vivado to test your designs. Simulations might be easier at the individual module level, allowing for simpler test benches. Once you are certain that your sub-modules run as intended, assembling them in a top module should be much easier to design and troubleshoot.
6. Test and confirm the working of the two smart-home control functions separately. You can parallelly prepare the documentations for each module in your report.
7. Integrate both functions so that one top module can coordinate the smart-home controller's full operation. Integration may allow you to include additional functions if you feel comfortable with your workflow so far.

4 Deliverables And Assessment

4.1 Assessment Process

For this project, **students will work in groups of 2**. The work will be assessed as group work, with the same mark awarded to both members, except for the following cases:

- If one of the members has not contributed fairly to the project, or if one member does not perform equally well in the Q&A session during the demonstration.
- The additional question for **ENGN6213** students must be worked on individually and will be marked on an individual basis.

The assessment of the work will be done by

1. **Demonstrate the design implementation in hardware**. It is **due in Week 7**. Please book your demo session time in Wattle using this link https://wattlecourses.anu.edu.au/mod/groupselect/view.php?id=2799286.
   - A Q&A session will follow the demonstration of your working hardware to check the understanding and involvement of each group member.
   - Assessment of the demo of each group should take around 10 to 15 minutes.
   - Ensure that the design is built and deployed to your Basys3 board beforehand so that the demonstration can start immediately in your selected time slot.
2. **Documentation (Report)** to explain your design, including the structure and features. It is due **on Friday 21 April 2023 4:59 pm**.

   Your delivered project document should give a *clear and concise but complete presentation of your work*. You should imagine that your document is to serve as a comprehensive technical document to be read by a technically knowledgeable user who knows nothing about your design.

   Do not tell a story about how you worked through your assignment; rather holistically

present your work. 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 report of one complex machine with several functions, not four separate machines**. It is most likely, for example, that some sub-modules 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.

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 behavior.
    - Block diagrams at the sub-module level. You should not show every logic gate or flip-flop in your design, but we also do not want to see too many "magic" black boxes in your circuit diagram whose operation is mysterious.
    - For designs 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 (nothing too verbose: 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*.)

The final report must not exceed 10 pages. Any content beyond this page limit will not be marked. Remember that title page, table of contents, references, and appendices do not count toward this page limit. You can include the Verilog code of your source and constraint files as appendixes, in addition to uploading your whole project code as a zip file in your submission.

4.1.1 Code

You are required to submit a functioning version of the Verilog code corresponding to your final implementation of the whole project, i.e., the whole project folder, including the constraints and Vivado project file of .xpr extension.

- Please ensure that you submit the correct, working version of your code, otherwise, penalisations will be applied.
- Your code must be properly commented for easy understanding during marking.

4.1.2 Submissions and Due Dates

- The project demonstration is due in **Week 7**. Please book your demo session time in Wattle using this link: https://wattlecourses.anu.edu.au/mod/groupselect/view.php?id=2799286.

- The project *Report* and *Code* submissions are due *by* **4:59 pm on Friday 21 April 2023**. Late submissions will be penalised until the cut-off date on Friday 28 April 2023 4:59 pm, beyond which submissions are no longer accepted.
  - The submission links will be made available in Wattle before the start of the mid-term break.
  - Code must be submitted as a zip file.
  - Report must be submitted as a pdf file.
  - Only one submission per group is required.
  - In the report, clearly identify the group members with University ID numbers. Otherwise, you will not receive a grade.
  - **ENGN6213** students should submit their individual responses to the additional reflective question as a separate pdf document.
  - It is strongly recommended that you re-download your submission after uploading it to Wattle to check for completeness and version accuracy before you finally send it for marking. Failing to upload the correct files will not be a ground for appeal of a poor mark.

Extensions will be granted only in exceptional circumstances with the permission of the course co-conveners. Please email course co-conveners if you have approved EAP and/or valid reasons for extension requests.

4.2 Marking Criteria

This project is **worth 25 %** of your total grade. It will be marked out of 100 with the following weightings:

Item			**Marks**	
Demo	Function 1	Minimum Requirements	12	**40**
		Advance Features	3	
	Function 2	Minimum Requirements	10	
		Advance Features	3	
	Integrated Controller	Minimum Requirements	5	
		Advance Features	2	
	Question and Answer Performance		5	
Report	Design approach and operation		35	**60**
	Design quality		15	
	Clarity of writing		7	
	Overall quality and readability of codes		3	
Additional Question for ENGN6213 students only	ENGN6213 students are marked out of 110 and then scaled to 100.		**10**	

For each component of the report, the following make up the partial marks:

Design approach and operation	• Block diagram representation of the hierarchical structure • Features and operation of the system with state diagrams/tables • Clear identification of user interface with input/output controls

Design quality	• Clear and logical design structure
	  ○ Sequential and combinational logic sections are appropriately recruited and separated.
	  ○ Sub-modules are used appropriately to isolate specific functions.
	  ○ Logical assignments of states with no ambiguity and proper reset/initialisation conditions
	  ○ Proper design to avoid excessive use of hardware resources.
	• Consideration of potential issues while dealing with asynchronous inputs, noisy external inputs, and accidental assertion of inputs.
	  ○ Use of synchronisers and debouncers. Justification of using or not using such measures.
	  ○ Management of malfunctions due to metastability and hazards
	• Creative Aspects

Clarity of writing	• Clear, concise, and comprehensive explanation. Readers should understand the design without having to refer to the fine details in the Verilog code. • Proper formatting of the document with appropriate margins, indentations, and consistent style.
Overall quality and readability of code	• Use of hierarchical design elements • Proper indentation • Presence and usefulness of comments

Note: Presenting the demo is mandatory for your report to be graded.

4.2.1 Late Delivery

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 after the due date on Friday 21 April 2023 4:59 pm until the cut-off date on Friday 28 April 2023 4:59 pm.

4.2.2 Referencing

IEEE citation style is usually used for referencing in the engineering field. You must cite any code, IP core, or design you use from external sources. Students should remain mindful of the possible disciplinary consequences of poor referencing practices.

4.2.3 Plagiarism

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 and citations in the report. Please be aware that our prevention strategies include checking your work against a broad list of internet resources.