Australian National University

# *FPGA Project:*

## *Smart Vault Controller using Basys3 FPGA Development Board*

ENGN4213/6213
Digital Systems and Microprocessors

Semester 1, 2024
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-Vault' controller***. The smart-vault is a highly secure room in a bank that has several automated functions, such as multi-factor authentication, remote locking, real-time monitoring, alarm system, and smart room settings like control of lights and temperature. These functions work 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-vault functions: Vault Door Access Control and Automated 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: Vault Door Access Control

The vault door is a secured two-way access-controlled device with the following operational features:

### *1.1.1 Features for Entry*

*This feature set is defined for when the vault door is closed and there are no people inside. The user accompanied by bank official(s) wishes to enter the vault.*

To open the vault door, you need to enter a particular 7-digit account pin and then press ENTER to open the door. This *7-digit account pin* should be the *binary equivalent of the Project Group Number* you selected in Wattle.

a) If the correct pin is entered, the controller will open the door to let the user and the bank official(s) inside.

b) If an incorrect pin is entered, the door will remain closed, and an alarm will start sounding but the system will give you a second entry chance.
   - In the second chance, you should enter the correct pin within 20 seconds. Then the door will open, and the alarm will stop.
   - In the second chance, if you enter the wrong code or fail to enter the correct pin within 20 seconds, you will get enclosed inside a trap structure. The alarm sound will continue, and the vault door will remain closed. In the trapped state, no further attempts to open the door are possible. The alarm can be reset, and the trap can be opened using a SECURITY_RESET control.

### *1.1.2 Features for Exit*

*This feature set is defined for when the vault door is closed and there are people inside who wish to exit.*

To open the door in this condition, the bank official inside should enter a *2-digit pin in Morse code* to open the door.

Morse code uses short and long signals, referred to as dots and dashes, respectively, to encode letters and numbers. For background information on Morse code and its timing you can refer to these links (Morse code chart and timing). The length of a dot is 1 time unit. A dash is 3 time units. For our purpose, we do not need to consider space between characters and words.

a) A 2-digit pin corresponds to a 10-signal Morse sequence, where each signal is a dot or a dash. Once the bank official completely inputs the 10-signal sequence, the system automatically accepts the pin and checks for its validity, i.e., no extra trigger is needed to accept the pin.
   - There is a set of 12 valid 2-digit pins that can open the door. They are:
     - 07 (i.e. – – – – – – – · · ·)
     - 10
     - 23
     - 34
     - 56
     - 67
     - 72
     - 78
     - 37
     - 45
     - 89
     - 92

     If the bank official enters any one of the *above pins in Morse code*, the door should open and people can exit.
   - You should choose the duration of time unit of Morse code such that entering the whole 2-digit pin in your board takes no longer than 30 seconds. Clearly mention the chosen time unit duration in your report and during the demo session.
   - The pin input buffer should reset after the validity checking.

b) Upon wrong pin input, the bank official can attempt unlimited times to enter the correct code.

### *1.1.3 Common features*

a) Once the door is completely open (for entry or exit), the door will start closing after 30 seconds.

b) When there are people already inside the closed vault, no new entry is allowed.
c) The system should sense the number of people entering and exiting through the vault door. The maximum number of people allowed inside the vault is 9.
d) A display outside the vault should continuously show the number of people inside the vault.
e) The door can be opened from outside anytime using a DOOR_MASTER control.

Based on the above features, we suggest the following I/O operations:

**Inputs**:

- Seven slide switches to input the 7-digit pin code.
- One push button as ENTER.
- One push button as SECURITY_RESET.
- One push button to enter the 2-digit pin in Morse code.
- One push button as DOOR_MASTER.
- Two slide switches: one to count the entry of each person into the vault and another to count the exit of each person from the vault. The positive/negative edges of the switching action can be used to update the count value.

**Outputs**:

- 4 LEDs to indicate the status of the vault door.
  - By default, the door is *closed* – indicated by 4 LEDs switched ON.
  - After the correct pin code input and pressing of ENTER, the door opening can be indicated by shifting LEDs (say, LEDs turn OFF one by one every 1 second) such that all LEDs turn OFF when the door is completely *open*.
  - Once the door is completely open (4 LEDs OFF), the door will start closing after 30 seconds. The closing action 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*.
- 4 LEDs to indicate the status of the alarm and trap structure.
  - By default, the alarm and trap structure is not activated. This is indicated by 4 LEDs switched OFF.
  - If the first attempt for entry fails, the alarm state can be indicated by flashing all 4 LEDs every 2 seconds.
  - If the second attempt for entry fails, the alarm and trapped state can be indicated by flashing all 4 LEDs every 0.5 seconds.
- One SSD to show the number of people inside the vault. Assume it is zero by default.

## 1.2 Function II: Automated Climate Control

The climate control function automates the heating/cooling system inside the vault depending on the current room temperature, desired temperature level and the number of people inside the vault. It has the following operational features:

a) The climate control turns ON automatically if the vault is occupied, i.e., people count >0. Once it is ON:
   - The current room temperature should be displayed.
   - If the current temperature > desired temperature, the system should cool the room towards the desired temperature.
   - If the current temperature < desired temperature, the system should heat the room towards the desired temperature.
   - If people count ≤ 5, heating/cooling the room changes the temperature at the rate of 1° *per 0.5 seconds.*
   - If people count > 5, heating/cooling the room changes the temperature at the rate of 1° *per 1 seconds.*

b) If the occupied vault is being vacated, the climate control will turn OFF in 10 seconds after the last person exits the vault.
   - In this condition, room temperature decrease/increase towards the outside temperature at the rate of 1° *per 2 seconds.*

c) Once the climate control is OFF, the temperature display should turn off in 20 seconds.

**Assumptions**:
- Possible temperature range is between 20-27 degrees (i.e., a range of 8 degrees).
- Outside temperature is given as an input. By default, people count = 0 and the current room temperature will be equal to the outside temperature.

**Inputs:**
- 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).

## 1.3 Advanced Features

Section 1.1 and 1.2 provides the mandatory features of the Smart Vault system. Apart from these features, you can implement any *two* additional features. Some of the suggestions are below:

- Automatically switch on lights inside the vault if there are people inside. Once all people exit, switch them off after 10 seconds. To indicate this function, make use of remaining LEDs.
- A different 7-digit account pin using the same seven slide switches can be used to reset trapped state instead of using SECURITY_RESET button.
- Display the state of the climate control ('ON' or 'OFF') using SSDs. Since we only have four SSDs, you can toggle this feature with the temperature display.

- Entering '36' (close to SOS) with Morse code button will turn all LEDs and SSDs off to mimic system shutdown.
- During exit attempt, use the remaining one SSD to display the dot/dash signal on each button press. Upon correct pin entry, display the BCD equivalent of the 2-digit pin using the remaining LEDs.

### 1.4 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 the Smart-Vault Controller consisting of Functions I and II**, i.e., ***both functions should be active simultaneously following the mentioned functionalities and dependencies***.

- 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-vault controller. For overall control of your smart-vault 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 functional requirements mentioned in Sections 1.1 and 1.2 are satisfied*.
- You have the freedom to implement advance features other than the above-mentioned ideas. Be creative with your design.
- We strongly encourage the use of 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 and demo session** on Wattle: (https://wattlecourses.anu.edu.au/mod/groupselect/view.php?id=3042858)
2. **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.
3. 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.
4. 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.
5. Test and confirm the working of the two smart-vault control functions separately. You can parallelly prepare the documentations for each module in your report.
6. Integrate both functions so that one top module can coordinate the smart-vault 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**.
    - The demonstration will include Q&As to check the understanding and involvement of

each group member.
- Assessment of the demo of each group should take around 10 to 12 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. No extra time will be allowed to complete the demo. The Q&A will follow strict time control.
- We suggest you prepare a simple handout of I/O pins used on your board and their associated signal names as mentioned in the function requirements. You can share this with the tutor during the demo to efficiently manage the time.

2. **Documentation (Report)** to explain your design, including the structure and features. It is due **on Friday 19 April 2024 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=3042858.
- The project *Report* and *Code* submissions are due *by* **4:59 pm on Friday 19 April 2024**. Late submissions will be penalised until the cut-off date on Friday 26 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.
    - The coversheet (provided in project resources) including the contribution statement must be appended to the report.
    - Clearly identify the group members with University ID numbers in the report coversheet. Otherwise, you will not receive a grade.
  - Only one submission per group is required.
  - **ENGN6213** students should submit their individual responses to the additional question given in Section 2 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 convener. Please email course convener if you have approved EAP and/or valid reasons for extension requests.

## 4.2 Marking Criteria

This project is **worth 35 %** of your total grade. It will be marked out of 100 with the following weightings:

| Item | | Marks | |
|---|---|---|---|
| **Demo** | Function 1 and its Q&A | 13 | **30** |
| | Function 2 and its Q&A | 9 | |
| | Integrated functioning | 2 | |
| | Advance Features and its Q&A | 6 | |
| | | | |
| **Report** | Design approach and operation | 40 | **70** |
| | Design quality | 15 | |
| | Clarity of writing | 10 | |
| | Overall quality and readability of codes | 5 | |
| | | | |
| **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<br>• Features and operation of the system with state diagrams/tables<br>• Clear identification of user interface with input/output controls |
| Design quality | • Clear and logical design structure<br>  ○ Sequential and combinational logic sections are appropriately recruited and separated.<br>  ○ Sub-modules are used appropriately to isolate specific functions.<br>  ○ Logical assignments of states with no ambiguity and proper reset/initialisation conditions<br>  ○ Proper design to avoid excessive use of hardware resources.<br>• Consideration of potential issues while dealing with asynchronous inputs, noisy external inputs, and accidental assertion of inputs.<br>  ○ Use of synchronisers and debouncers. Justification of using or not using such measures.<br>  ○ Management of malfunctions due to metastability and hazards<br>• 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.<br>• Proper formatting of the document with appropriate margins, indentations, and consistent style. |
| Overall quality and readability of code | • Use of hierarchical design elements<br>• Proper indentation<br>• 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 19 April 2024 4:59 pm until the cut-off date on Friday 26 April 2024 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.