Australian National University

System Overview

This FPGA-based project features an advanced smart vault controller, designed for secure access and climate management within a vault. It employs the Basys3 FPGA Development Board, leveraging switches, buttons, LEDs, and a seven-segment display to facilitate interaction. The system is equipped with functionalities for vault door access control and automated climate control, offering a comprehensive user interface for operation.

User Interface Description

Inputs

- Switches (sw[0] to sw[15]):
  - sw[0] to sw[6]: Input for the 7-digit pin code to access the vault.
  - sw[7] to sw[8]: Indicators for a person entering or exiting.
  - sw[9] to sw[11]: Selection of the outside temperature (for climate control).
  - sw[12] to sw[14]: Setting the desired temperature within the vault.
  - sw[15]: Acts as a system reset switch.
- Buttons:
  - btnC (Center): Confirm pin code entries and operations.
  - btnU (Up): Security reset to deactivate alarms or exit trap mode.
  - btnL (Left): Input the 2-digit pin in Morse code for exit.
  - btnR (Right): Master control for external door operations.

Outputs

- LEDs (led[0] to led[15]):
  - led[0] to led[3]: Display vault door status (open, opening, closing, closed).
  - led[4] to led[7]: Indicate alarm or trap status.
  - led[15]: System heartbeat, indicating operation.
  - led[14]: Reflects the reset status.
- Seven-Segment Display (seg[0] to seg[6], an[0] to an[3]):
  - Displays the number of people inside the vault.
  - Shows the room temperature as managed by climate control.

Operation

1. **Door opening/closing status**
   - Once the door is prepared to open, LEDs 0 through 3 go out one at a time every second; these LEDs all show that the door is open and will stay open for thirty seconds. Next, one by one, the LEDs in the reverse direction (LED3 to LED0) light up to show that the door is closed once more.
2. **Vault Entry Process with Pin Code "57"**
   - Users need to enter the pin code "57" in order to access the vault. The binary representation of this is 0111001. Users must ensure that sw[1], sw[2], and sw[6] are set to low (or "0") and set sw[0], sw[3], sw[4], and sw[5] to high (or "1") in order to input this code successfully. Once the switches have been adjusted appropriately, press btnC to verify the entry.
   - Correct Password Entry: The LEDs and the seven-segment display update to show the vault door opening and the updated number of people inside if the switches correctly reflect the binary code for "57". The system then validates the entry.
   - Incorrect Password Attempt: The system will flash once every 2 seconds to indicate that an incorrect pin was entered. This will cause led[4] to led[7] to blink. The user is subsequently given one more opportunity to enter the accurate pin:
     - Right on the Second Try: The door opens in the expected manner.

- Incorrect or No Entry Within 20 Seconds: The blinking of led[4] to led[7] increases to once per half-second, signifying that the system has gone into trap mode. In this state, the only way to get the system out of trap mode and back into pre-entry state is to click the security reset button (btnU).

3. Vault Occupancy Control and Display

2.1 **People Count Adjustment**

- Incrementing People Count (SW7): The technology increases the number of persons within the vault by one every second when SW7 is kept at a high state. This increase keeps on until the maximum of nine individuals can fit inside.
- Decrementing People Count (SW8): On the other hand, if sw8 is kept high, the system will reduce the number of people by one every second until it reaches zero.
- Stable Count (SW7 and SW8 at the same time): The person's count stays constant if both sw7 and sw8 are maintained in a high state.

2.2 **Display Mechanism**

- The rightmost SSD dynamically displays the current number of persons within the vault. Upon system reset or startup, this display starts at zero and updates in real time to reflect variations in occupancy.

2.3 **Switch Operation Restrictions**

- Active Door State: Only while the vault door is open can you use SW7 and SW8 to change the number of individuals counted. Accurate tracking of people entering and departing the vault is ensured by this function.
- Inactive Upon Door Closure: SW7 and SW8 stop having an impact on the people count as soon as the vault door starts to close (as shown by LED0 to LED4).

4. **Exiting the Vault:**
- By pushing the btnL key in accordance with the 2-digit pin's Morse code sequence, the user starts the application. pushing for a duration more than three seconds is referred to as a dash, whereas pushing for a duration of one to three seconds is called a dot. Without further confirmation, the system will automatically verify that the code entered is valid after the input sequence has been completed.
- If it's input correctly, the procedure will cause the door to open. The system has 12 valid password options set, which are 07, 56, 37, 10,,67, 45, 23, 72, 89, 34, 78, and 92 (which must be transformed into a binary sequence before being entered).

5. **Climate Control:** An increase in the people count indicates that at least one person has been spotted within the vault, which triggers the system. Switches sw[12] to sw[14] are used by the user to set the desired temperature; each switch represents a bit in a 3-bit binary number that corresponds to a temperature range of 20°C to 27°C. The SSDs on the left are used to monitor and display the current temperature. It automatically modifies itself to heat when the temperature drops and cool when it rises. In order to account for body heat, the rate of temperature change is dependent on occupancy; if five or less persons are detected, the change happens more quickly; if more than five, it happens more slowly.

After a 10-second pause, the system will deactivate when individuals leave and the vault becomes deserted, gradually bringing the temperature back to the outside ambient temperature, which is controlled by sw[9] to sw[11].

6. **Security Reset:**
- Press btnU to reset all outputs (LEDs, SSDs) and the system to the initial state from which the entry password was input.
- It is advised that the user activate this button as soon as the system reaches the trap state.

7. **Door Master:** btnR opens the vault door, it is recommended for use inside the door, then no need to enter Morse code.

Code Before Improved:

We partially implemented function 1 in the prior demo session; After this week's work, we were able to implement function 2 and address some display and logic issues with function 1.

1. Alarm and Trap: In the code demonstration now, after entering the wrong password the first time as well as the second time, reps LED4-7 behave the same way, i.e., the display of alarm and trap are the same. After debugging, we found that the problem lies in the state machine settings. In the previous code, if the user enters a wrong password in DOOR_CLOSED state and presses ENTER, the system will directly enter DOOR_TRAP, which causes the alarm state to be ignored, but after fixing it, the alarm and trap can be clearly distinguished.

```
S_DOOR_CLOSED:
    if      (ENTER && pin_code == PASSWD) state <= S_DOOR_ENTER_OPENING;
    else if (ENTER && pin_code != PASSWD) state <= S_DOOR_CHANCE;
    else                                  state <= S_DOOR_CLOSED;
```

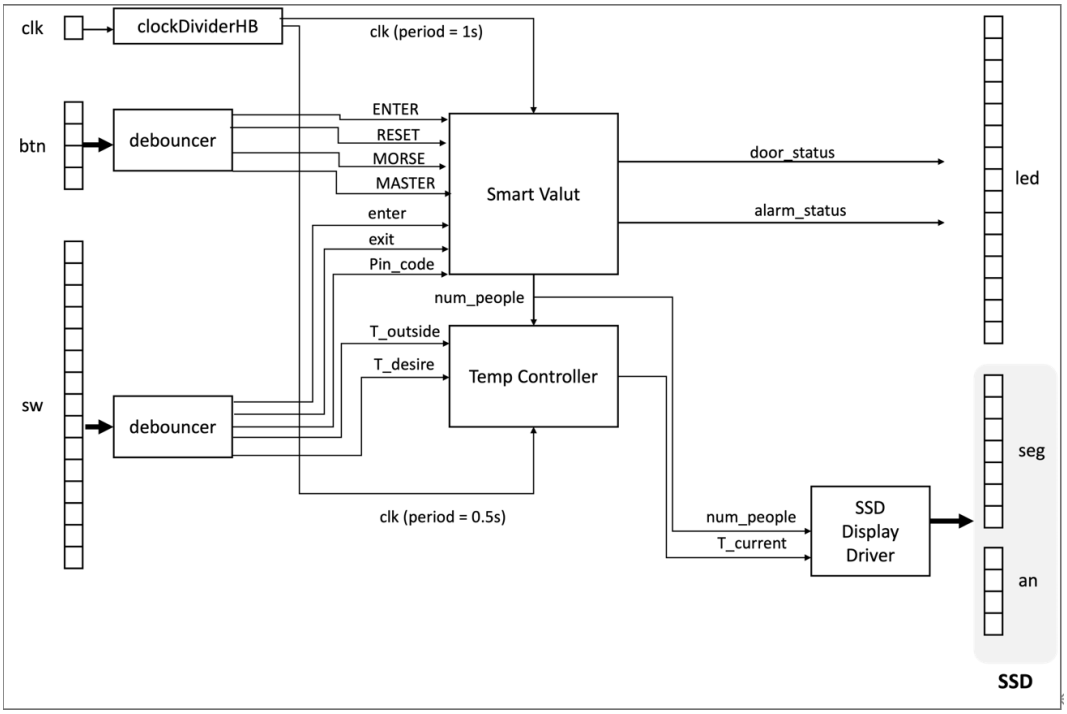
2. People number displayed by SSD: While the logic behind adding and subtracting people in the demo is sound, the SSD used to display the number of people in the vault does not always display the number correctly (it is sometimes out of order and does not even display a normal number). Upon debugging and comparing the code of lab 3, it was discovered that the digital tube's order in the constraints file was written backwards, resulting in the segment[0–6] being written as segment[6-0]. With this corrected, the modified SSD is able to update the number of people in real time normally.

3. Morse Code: Previously, the Morse code sequence was designed with each bit initially low and shifted to the left by one bit every second. During each time unit, the system would automatically press a dot if the user did not input anything and a dash if the user pressed a button. After each time unit ended, the system would test the sequence once to see if it matched any of the list of pin codes or not. However, the tutor notes that this isn't relevant to the topic, which requests a reflection stating that the DOT takes three times as long as the DASH. So we re-designed the morse code and returned to the original scheme, i.e., long-pressing btn for more than three seconds is regarded as dash, and pressing btn for between one and three seconds is regarded as dot. Notice the improvement of Morse code was not entirely successful; some of the twelve ciphers could not be recognized or opened the door before they were finished being typed in. The current theory is that it has a high degree of overlap with the binary sequence of another string of ciphers, so it is recognized as another cipher. This has not yet been verified, but sadly, there isn't time to make any more refinements.

4. We have basically implemented the functionality of function2, see the user interface for details and the description below.

Technical Description of Structure and Operation:

Block Diagram:



First, this paper will introduce the modules involved in the implementation process, and then this paper will introduce the respective variables and functions of different modules.

1. The TOP module serves as the top-level module of the entire system and coordinates the operation of all other sub-modules. It connects the user inputs (such as buttons and switches) and controls the outputs (such as seven-segment displays and LEDs).

```
wire [ 3:0] numbers;
wire        reset = sw[15];
wire        clk_div, clk_temp;


wire        btnC_stable, btnU_stable, btnL_stable, btnR_stable;
wire [15:0] sw_stable;

wire [ 2:0] temp_outside = sw_stable[11:9];
wire [ 2:0] temp_desired = sw_stable[14:12];
wire [ 3:0] temp_lo;
wire [ 3:0] temp_hi = 4'd2;
wire        temp_en;

wire        seg_clk;

reg [2:0] an_cnt;
```

2. Seven-segment display module: Converts binary-coded numbers into signals suitable for seven-segment displays for the numbers 0-9.

```
always @(posedge seg_clk) begin
    an_cnt <= an_cnt + 1'b1;
end

assign an = (~(4'b0001 << an_cnt[2:1]) | {~temp_en, ~temp_en, 1'b1, 1'b0}) | {4{an_cnt[0]}};
wire [3:0] bcd = ({4{an_cnt == 3'b000}} & numbers) |
                 ({4{an_cnt == 3'b100}} & temp_lo) |
                 ({4{an_cnt == 3'd110}} & temp_hi) ;
```

This module is called in top module, an_cnt is a counter for the seven-segment display driver. It starts counting when the display clock is on the rising edge. an and bcd, two key elements, play a significant role in managing the segments and content of the display, providing precise control over its output.

3.Clock divider module: a parameterized clock divider that, depending on the given THRESHOLD, produces a divided clock signal that is slower than the input clock.

```
clockDividerHB #(.THRESHOLD(100_000)) clockDividerHB_seg(
    .reset          (reset      ),
    .enable         (1'b1       ),
    .clk            (clk        ),
    .dividedClk     (seg_clk    ),
    .beat           (           )
);

clockDividerHB #(.THRESHOLD(25_000_000)) clockDividerHB_temp(
    .reset          (reset      ),
    .enable         (1'b1       ),
    .clk            (clk        ),
    .dividedClk     (clk_temp   ),
    .beat           (           )
);
```

This module is called in top module. The system produces two signals after the frequency divider: seg_clk and clk_temp; these two clock signals have different frequencies, respectively, on behalf of the clock signal of the seven-segment display and the clock signal of the temperature module.

4.key debounce module: To provide a steady signal input to the system, jitter from switch or button signal inputs must be minimised. Every physical button needs to be debounced.

```
debouncer debouncer_btnC (
    .switchIn       (btnC       ),
    .clk            (clk        ),
    .reset          (reset      ),
    .debounceout    (btnC_stable)
);

debouncer debouncer_btnU (
    .switchIn       (btnU       ),
    .clk            (clk        ),
    .reset          (reset      ),
    .debounceout    (btnU_stable)
);
```

```
debouncer debouncer_btnL (
    .switchIn       (btnL       ),
    .clk            (clk        ),
    .reset          (reset      ),
    .debounceout    (btnL_stable)
);

debouncer debouncer_btnR (
    .switchIn       (btnR       ),
    .clk            (clk        ),
    .reset          (reset      ),
    .debounceout    (btnR_stable)
);
```

5.temp module: controls the climate inside the vault by automatically changing the temperature based on the number of persons inside, the specified temperature, and the current temperature.

By using basic binary inputs and producing a predictable output of the current temperature, complex control logic can be made simpler.

```
temp temp_i(
    .clk            (clk_temp    ),
    .reset          (reset       ),
    .temp_out       (temp_outside),
    .temp_set       (temp_desired),
    .num            (numbers     ),
    .enable         (temp_en     ),
    .temp_curr      (temp_lo     )
);
```

The module contains several counters and status registers inside:

disable_cnt: A countdown timer used to turn off the climate control system after the vault is emptied.

cnt: used to generate clock signals of different frequencies to control the temperature adjustment rate

6.smart vault module: Manage the vault's door control system, including entry and exit privilege verification, door opening and closing status, and alarm systems.

```
smart_valut smart_valut_i(
    .clk            (clk_div    ),
    .reset          (reset      ),
    .pin_code       (sw_stable[6:0]),
    .ENTER          (btnC_stable),
    .SECURITY_RESET (btnU_stable),
    .MORSE          (btnL_stable),
    .DOOR_MASTER    (btnR_stable),
    .person_enter   (sw_stable[7]),
    .person_exit    (sw_stable[8]),

    .status_door    (led[3:0]   ),
    .status_alarm   (led[7:4]   ),
    .person_num     (numbers    )
);
```

7. smart_vault and finite state machine of the door:

```
module smart_valut(
    input           clk,
    input           reset,
    input   [ 6:0] pin_code,
    input           ENTER,
    input           SECURITY_RESET,
    input           MORSE,
    input           DOOR_MASTER,
    input           person_enter,
    input           person_exit,

    output [ 3:0] status_door,
    output [ 3:0] status_alarm,
    output reg [ 3:0] person_num
);
```

```
localparam S_DOOR_CLOSED         = 4'h0;
localparam S_DOOR_ENTER_OPENING  = 4'h1;
localparam S_DOOR_ENTER_OPENED   = 4'h2;
localparam S_DOOR_ENTER_CLOSING  = 4'h3;
localparam S_DOOR_ENTER_CLOSED   = 4'h4;
localparam S_DOOR_EXIT_OPENING   = 4'h5;
localparam S_DOOR_EXIT_OPENED    = 4'h6;
localparam S_DOOR_EXIT_CLOSING   = 4'h7;
localparam S_DOOR_CHANCE         = 4'h8;
localparam S_DOOR_TRAP           = 4'h9;

localparam PASSWD                = 7'd57;

reg   [ 3:0] state;
wire         person_full;

reg   [ 3:0] door_open_close_cnt;
```

As shown in the above left figure, the smart_vault module is defined , and the smart_vault_i of the previous safe access management module is an instantiation of this module. In addition, the following figure defines the logic and scenarios for user access to the vault door:

First, as shown as above right figure, the code above defines the possible states of the door and the passwords to access.Secondly, we define a state machine for the door, where the state of the door changes under different conditions, as shown below:

```
S_DOOR_CLOSED:
    if      (ENTER && pin_code == PASSWD) state <= S_DOOR_ENTER_OPENING;
    else if (ENTER && pin_code != PASSWD) state <= S_DOOR_CHANCE;
    else                                  state <= S_DOOR_CLOSED;
S_DOOR_ENTER_OPENING:
    if (door_open_close_cnt == 3'd4) state <= S_DOOR_ENTER_OPENED;
    else                             state <= S_DOOR_ENTER_OPENING;
S_DOOR_ENTER_OPENED:
    if (door_open_close_timeout_cnt == 5'd30) state <= S_DOOR_ENTER_CLOSING;
    else                                                state <= S_DOOR_ENTER_OPENED;
S_DOOR_ENTER_CLOSING:
    if (door_open_close_cnt == 3'd4) state <= S_DOOR_ENTER_CLOSED;
    else                             state <= S_DOOR_ENTER_CLOSING;
```

In the state machine above, we can see that the door has multiple states and conditional transitions between states. The specific detail of the states can be seen below.

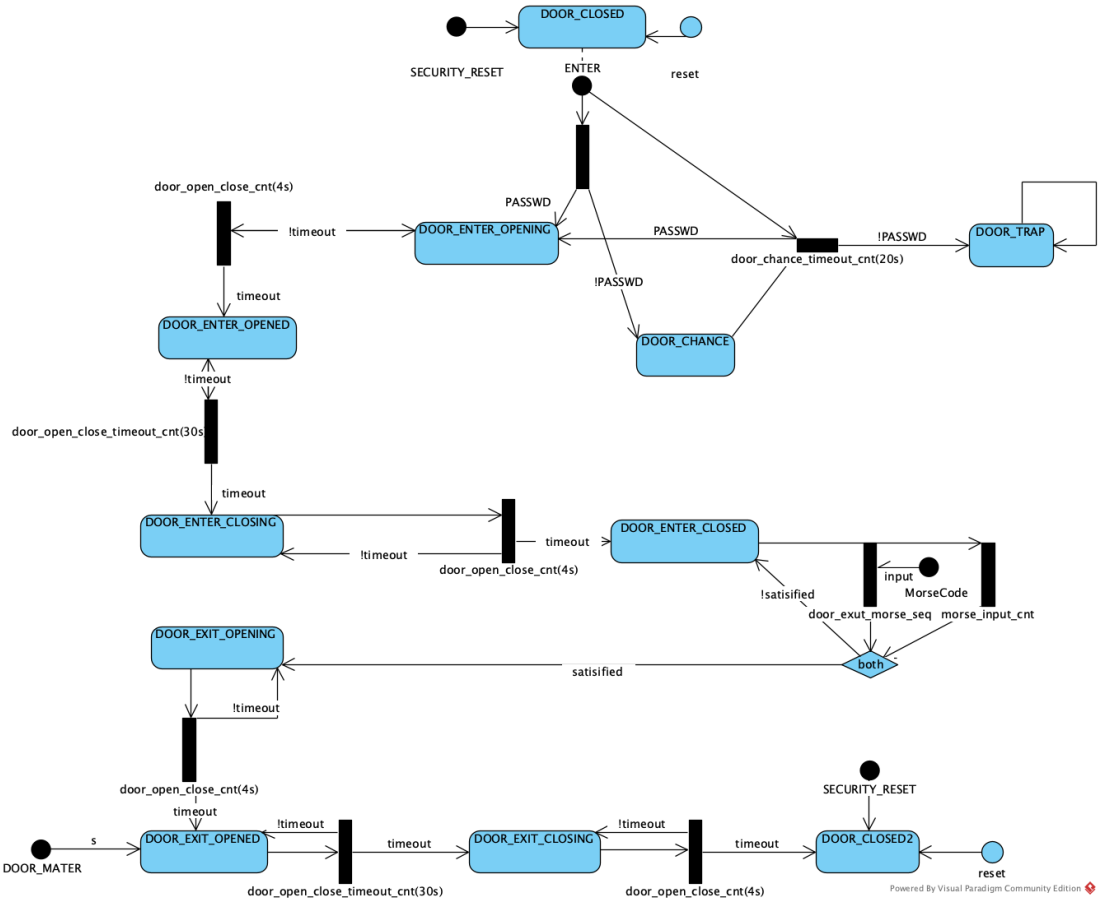
9. Processing of passwords on entry and Morse code on exit: As shown in the following figure, the detailed processing for entering the password on entry and entering the Morse code on exit is carried out:

```
S_DOOR_ENTER_CLOSED:
    if((door_exit_morse_seq == 10'd07 || door_exit_morse_seq == 10'd56 ||
        door_exit_morse_seq == 10'd37 || door_exit_morse_seq == 10'd10 ||
        door_exit_morse_seq == 10'd67 || door_exit_morse_seq == 10'd45 ||
        door_exit_morse_seq == 10'd23 || door_exit_morse_seq == 10'd72 ||
        door_exit_morse_seq == 10'd89 || door_exit_morse_seq == 10'd34 ||
        door_exit_morse_seq == 10'd78 || door_exit_morse_seq == 10'd92 )  && morse_input_cnt >= 32'd10)  state <= S_DOOR_EXIT_OPENING;
    else                                                                  state <= S_DOOR_ENTER_CLOSED;
S_DOOR_EXIT_OPENING:
    if (door_open_close_cnt == 3'd4) state <= S_DOOR_EXIT_OPENED;
    else                             state <= S_DOOR_EXIT_OPENING;
S_DOOR_EXIT_OPENED:
    if (door_open_close_timeout_cnt == 5'd30) state <= S_DOOR_EXIT_CLOSING;
    else                                      state <= S_DOOR_EXIT_OPENED;
S_DOOR_EXIT_CLOSING:
    if (door_open_close_cnt == 3'd4) state <= S_DOOR_CLOSED;
    else                             state <= S_DOOR_EXIT_CLOSING;
S_DOOR_CHANCE:
    if (door_chance_timeout_cnt <= 5'd20 && ENTER && pin_code == PASSWD)         state <= S_DOOR_ENTER_OPENING;
    else if (door_chance_timeout_cnt > 5'd20 || (ENTER && pin_code != PASSWD))   state <= S_DOOR_TRAP;
    else                                                                         state <= S_DOOR_CHANCE;
S_DOOR_TRAP: state <= S_DOOR_TRAP;
```

When the user enters the correct Morse code, the door will change from a closed state to an open state, and when the user enters the code within the specified conditions, the door will open to allow the user to enter. On the contrary, the trap will change to 1

Above is the structure of the project, in which the required functionality for two functions is successfully implemented: intelligent vault access control and climate control.

State Transition Diagram



State Information Table

A binary verification code is set for each state so that they can be found more quickly.

DOOR_CLOSED(0000): The vault door stays closed in this safe and default state of the system Selected as the point of entry to guarantee that security is preserved in the event that the system is inactive or has been reset

DOOR_ENTER_OPENING(0001): Represents the state that changes from closed to open when a PIN is entered successfully.It was selected to offer a buffer time, mirroring the actual door opening action, during which the system gets ready to permit admission.

DOOR_ENTER_OPENED(0010): Shows that the system is prepared for the user to enter and that the door is fully open. In order to prevent the door from being left unlocked, this state is essential for permitting admission and setting a timer to track how long the door is left open.

DOOR_ENTER_CLOSING(0011): signifies the act of shutting the vault door to secure it. Essential to restore the system to a secure state following user entry or if no entry is made within a predetermined amount of time.

DOOR_ENTER_CLOSED(0100): The state after the door has closed post-entry. It prepares the system for a potential exit sequence. Essential for maintaining security while also setting up the conditions needed for the exit process to initiate.

DOOR_EXIT_OPENING(0101): The door transitions to this state when a valid exit command or Morse code is received. It provides a parallel to the entry process, enabling a controlled and monitored exit from the vault.

DOOR_EXIT_OPENED(0110): Denotes an open door that is prepared for the user to leave; it is similar to the DOOR_ENTER_OPENED state but just for leaving. Important for encouraging a safe and user-friendly atmosphere and guaranteeing that the user can leave without hurry.

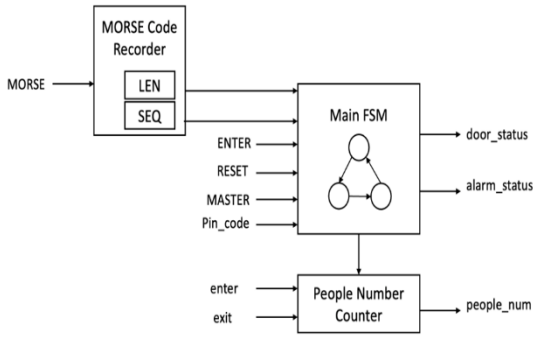
DOOR_EXIT_CLOSING(0111): Identifies with the DOOR_ENTER_CLOSING condition, but it proceeds from an exit instead of an entrance. Essential for automatically locking the vault back when someone has left.

DOOR_CHANCE(1000): allows the user to try again after a failed attempt to enter the right PIN. This mode strikes a compromise between security and usability, permitting inadvertent PIN entering errors without affecting security.

DOOR_TRAP(1001): A safety measure that kicks in when DOOR_CHANCE timeouts or many erroneous PIN entries are found. This state acts as a clear security procedure and is a significant barrier against efforts at unauthorised entry.

Motivation of design choices

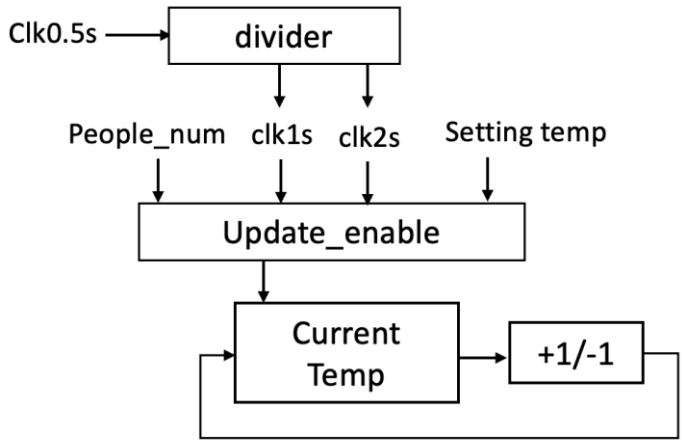
1. Smart Valut includes a master state machine that accepts passwords and control inputs. For Morse code processing, we use a shift register (SEQ) to save the input sequence: when the key is pressed for 3 seconds, a dash (0) is recorded the next clock cycle after the release, and a dot (1) is recorded if the key is pressed for 1 second. There is also a design decision: we set up an additional length register (LEN) to record the length of the Morse code that has been entered. If the length is less than 10 bits, Morse code will not be accepted. This design ensures that Morse code does not accept any input other than a given string of digits due to an initial value of all zeros.
2. For the output of this module, we decided that door_status and alarm_status can be generated by combining logic based on the current state. However, for the number of people, we need to set up an additional register for counting: in addition to receiving the state input from the state machine, it also accepts the enter and exit



signals as external drivers.

Australian National University

3. The following figure shows the structure of the temperature control module. At its core is a register that records the current temperature and will update the temperature (up or down by 1 degree) for each cycle that the enable is turned on. The prerequisite for enabling this function is that the current temperature is different from the set temperature. In addition, due to the different temperature growth rates at different stages, our design decision here is to use the fastest rate (0.5 seconds) as the clock cycle. In addition, we use a clock with a period of 0.5 seconds to split out clocks with the same phase and periods of 1 and 2 seconds. To update every 1 second or every 2 seconds, only the enable signal and the corresponding clock signal need to perform "logical AND (&)" operation.



Reference:

1.https://stackoverflow.com/questions/74621790/how-do-i-make-a-passcode-fsm-when-the-inputs-are-buttons The Morse code design inspired by this code. (top module, line 308-315)

2.https://www.youtube.com/watch?v=3lMLtQekqn8

The logic of this entry password, declaration of PASSWD, use of clocks, etc.

3.https://github.com/neelabhro/FSM-Verilog-

The structure of FSM in smart vault module.

4.https://www.scribd.com/doc/297877093/Morse-code-on-FPGA#:~:text=Report%20this%20Document-,This%20Verilog%20code%20implements%20a%20Morse%20code%20generator%20using%20a,timing%20for%20Morse%20code%20symbols.

Morse code and LED connection.

5. ChatGPT:

- Create clockDividerHB, temp, debouncer instances in top module (line 46-98)
- Temperature change expression (temp module: line 185-192)
- Uni-codes for states declaration (smart_valut module, line 220-229)