1. **Advertisement/Bus-Route Message Display** on VGA Monitor
Repeated display of advertisement/bus-route strings of 10-15 characters on the Monitor connected using VGA connector. Characters of the message will enter from right side of the screen and exit from left, repeatedly. There will be two modes, selected using a switch
1- select if the user will select the message or
2- messages will be displayed one after the other.
In mode 1, switches can be used to select different messages

The speed at which characters of the message are shifted on the screen is controlled using another switch. Display Messages can be pre stored in the memory.

2. **Festival Lights and Message**
LEDs on the board can be programmed to give a light effect of festival lights similar to what we use during Diwali, by controlling their on-off frequency and pattern in a regular and periodic fashion. In addition a short text message about the festival can be displayed on SSDs scrolling from left to right. There can be multiple light effects 3 to 4, one of which can be selected using switches on the board. We can also control their on-off frequency to change the patterns of the light.

3. **Temperature Display**

Reading the temperature using a temperature sensor and display of temperature on SSD. Display should be changed if temperature varies above some threshold value. Temperature sampling rate can be selected by the user using the 2 switches.
00: Sample at 1Hz
01: Sample at 0.5Hz
10: Sample at 0.2Hz
11:Sample at 0.1Hz

In addition one switch is used to select the unit of the display in degree Celsius or Fahrenheit.

There will be two user selected units of display
1. Current temperature
2.
3. Previous hour temperature

Temperature samples can be stored in memory for past few hours and user can select the time (in units of hours) to display previous hours temperatures.

4. **Nearest Station/Post Office Locator**

GPS coordinates of stations (latitude and longitude) and their station code can be saved in the board memory to create a data base. User can query the nearest Station or post office by sending its GPS location using UART. Algorithm should search for the nearest station and display its station code on the SSD.

5. Waveform generator

Create a waveform generator which displays waveform of certain type on a monitor connected via VGA port to the Basys board. Possibly, we can allow them to use an existing IP, but write the controller logic for the below.

- 3 switches determine the frequency: 8 values (values could be chosen so that they fit in the monitor).
- 2 switches determine the type of waveform: square, triangular, square, sine.
- 2 switches control the colour of waveform on the display: blue, green, red, black.
- 3 switches determine the amplitude: 8 values (values could be chosen so that they fit in the monitor).

VGA interface IP can be reused for this project.

6. CRC computation

Compute CRC of numbers (use standard CRC16-CCITT or any other standard). We will use 16-bit numbers as input (provided by 16 switches on the board). One of the push-button will act as a load input and one of the push-button will be a reset input. Every time, we load a new value from the switches, the CRC checksum is updated considering the new value as another data and the updated checksum is displayed on the SSD. Pressing reset will reset the checksum to the seed value. One of the LED could be used to indicate that the existing computation is over and the system is ready to take-in new input.

7. Car light controller (sequential)

There are several switches that influence the behaviour of the interior light. These are as follows.

* A three position switch (let us call it main switch), the positions of the switch are - OFF/DOOR/ON
* Door switches, operated by opening/closing of the door (one for each door)
* Door key switch, operated by inserting/removing the key in the driver side door
* Ignition switch, operated by turning the ignition ON or OFF

Two positions of the main switch, namely OFF and ON, override other switches and turn the light OFF or ON, respectively. When this switch is in DOOR position, the light responds to other switches in the following manner.

Typical entry sequence (light turns on at step 2 and gradually turns off at step 5):
1. Unlock the door
2. Open the door
3. Enter the car
4. Close the door
5. Put key in ignition switch
6. Switch on ignition

Typical exit sequence (light turns on at step 2 and gradually turns off at step 6):
1. Switch off ignition
2. Take ignition key out
3. Open the door
4. Exit the car
5. Close the door
6. Lock the door

When the light is off, opening any door turns it on and closing the door makes it turn off gradually. When the light is on, it times out in 10 sec if there is no event, turning off gradually. Turning off gradually takes 3 seconds in all cases.

There are numerous other sequences physically possible that need to be identified and the controller needs to be designed to ensure reasonable response in all cases.

Other displays on the FPGA board can be used to display additional information - 7-seg displays to show the timers (for time-out and gradual turning off) and LED displays for the switches and error conditions (if any).

8. Alarm clock

Implement an alarm clock. The clock is displayed as MM:SS (minutes:seconds) in the SSD. The user could set the switches to a certain time and then press a button (alarm-set) and this sets the alarm. Whenever, the particular time is reached in the clock, one LED (indicating the alarm) starts blinking. When we press a button (turn off alarm), the blinking goes off. Instead, if we press another button indicating snooze, then the alarm is turned off and the alarm value is updated with a new value (old value + 1 minute).
We could add upto 3 alarm values (there are 3 alarm-set buttons for each of the alarm).
These are some general thoughts, we could define clearly based on how our mobile alarm works.

9. Floating-point adder unit with 23-bit Fraction, 8-bit exponent and 1 bit sign.

Design a Floating-point adder unit (with two inputs and output in the Single-precision floating-point format). Suppose you want to add two floating point numbers, X and Y. For the sake of argument, assume the exponent in Y is less than or equal to the exponent in X. Let the exponent of Y be y and let the exponent of X be x. Following steps need to be performed:

1. To add, we need the exponents of the two numbers to be the same. We do this by rewriting Y. This will result in a Y being not normalized, but the value is equivalent to the normalized Y.
2. Add x - y to Y's exponent. Shift the radix point of the mantissa (significand) Y left by x - y to compensate for the change in the exponent.
3. Add the two mantissas of X and the adjusted Y together. Let us call the result unnormalized Z.

4. Normalize (with one nonzero decimal digit before the decimal point.), round (down) the result of the previous step and output the result Z (in the Single-precision floating-point format)

Although not described above, your implementation should be able to handle the following:

1. Negative Values (in inputs and outputs)
2. Indicate Overflow/Underflow (as outputs)
3. It's possible for a result to overflow (a result that's too large to be represented) or underflow (smaller in magnitude than the smallest denormal, but not zero).

Procedure to display the result on the Basys board will be updated in some time. If something is unspecified above, please make reasonable assumptions and proceed. Please use following references:

1. Single-precision floating-point format
2. Floating Point Numbers
3. “Computer Architecture A Quantitative Approach”. By John L Hennessy & David A Patterson. Appendix. J.5 Floating-Point Addition

10. Ping-Pong game :

The game is played by two players with two "push-button" switches for each player and a reset switch for initializing the game. Out of the two switches with each player, one switch (called "return") is for returning the ball and the other (called "return_speed") for changing the speed of return (two speeds permissible). There is a set of 16 LED's which show the position of the ball. There are two single digit displays (on seven segment) for showing the scores of the two players. The "return" switch is to be pressed only when the ball is on the last LED corresponding to the player (the two end LEDs correspond to the two players). In case the player fails to "return" the ball or "hits" earlier, he loses the point. The game stops when the first player reaches **9** points.

11. Reflex Tester:

This project aims to test how fast you can respond after seeing a visual stimulus or rather hand-eye coordination. There would be a “reset” button to reset all the states and make the system ready for the next reflex test. It will then display a “hi” message on 7-segment display to show that it is ready for next input. The user needs to press the “start” button to start a random number generator that turns on the “output led” after a random interval. (Randomness is important here, otherwise the time of led on can be memorized). After the led is on, the timer starts and counts the time till the user presses “stop” button. At this time, the reaction time should be displayed on the 7-segment display and hi message should be removed from 7-segment display.

12. Secure Car Parking System:

Design a **Finite State Machine (FSM)** to implement an efficient parking system. Assume you have three sensor inputs (from any of the push button or switches): “entry sensor” ,“park sensor” and “exit sensor”. The “entry sensor” detects if a vehicle has come to the gate of the parking system. The “park sensor” detects if the incoming vehicle has passed the gate and entered the car parking zone. The “exit sensor” indicates the car is heading on to the exit door. There should be an “IDLE” state that waits for a car to ask for entrance to parking. Design the system such that each car is asked for a password at the entrance of the car parking gate (if the parking space is available, assume finite parking space) and wait for 4 cycles for the user to enter the password. If you get no password, assume it to be wrong password and give it one more chance. Turn on (or blink, whichever is easier, but remember to turn it off before the other car says the correct password because you want to indicate this other car’s authentication too) the led to indicate the correct password. If the password is wrong, give the car another chance. (You can decide to now not allow this car, as it might be blocking other cars in the queue. This implies go to idle state). On correct password, open the gate and let the car in and assign it a unique id. If another car comes in from behind while the door is opened for the first car, stop it there and ask for the password. Track the information when the car exits and display the duration for which the car was parked along with its id on 7-segment display.

13. CORDIC

Implement the CORDIC algorithm using pipelined method. Have a switch to select between the vectoring and the rotation modes. CORDIC takes in the phase information and calculates sine and cosine of the angle given to it. Take in the phase information using 16- bit input. Read the algorithm from internet and implement it. Display the result on 7-segment display. Assume the Multiplication by scaling factor and input format conversion is done manually.

14. Implementation of searching algorithms

1. The initial part should be for N=8,where you will input the numbers using switches.Then initialize Bram and store inputs to it.
2. Take another input ,which is the number to be searched and display the search value location on the LEDs.
3. If the number occurs multiple times ,then display the multiple locations on led at an interval of 1 second.
4. Sweep the value of N=8 to higher values ,taking input from COE file and display the location on LED’s . Plot various graphs between N v/s (Delay and LUT’s)

Implement the above steps for Linear and Binary search .
Compare both the search architectures for a range of inputs(N) i.e (w.r.t LUT’s, Delay, Efficiency....)

15. Traffic light controller

Design a modern traffic light control system to manage road traffic. The heart of the design is a FSM (finite state machine) that directs the unit to light main and side street lights at appropriate times and for specified intervals of time. This needs to be a simple controller with fixed timing interval for a particular duration (Morning, noon,evening). The timing interval can vary across durations.

16. currency dispenser

Implement a currency dispenser. A fixed number of coins/notes is present in the machine (eg. no of Re 1 coins = 100 , no of Rs 5 coins = 50 and so on..). You need to keep track of all the notes/coins present in the machine at any point of time. User will enter an amount and may or may not specify the type of currency required.
You need to implement 3 modes:
Auto mode : like an ATM, user will specify the amount and dispenser will generate mixed notes/coins.
Higher denomination: Dispenser will generate the highest denominations present in the dispenser for the amount specified by user.
Depending on the user input: In this mode, user specifies the requirement. If the required notes/coins are not present in the machine, dispenser needs to show an error message and also displays the other currencies available to the user. Use LCD for display.

17. Long exposure photography

The idea is to print your name in the air. Persistence of Vision enables us to see the trail of a light even after it has changed. A sequence of LED values decided such that scrolling the LEDs in a horizontal manner would generate text. This scrolling of LEDs can be captured using a camera with long exposure settings.
The idea is to do something like this: https://www.youtube.com/watch?v=yaQjTOmd0ds
Existing set of LEDs on the board can be used.
Extra effort is required to make stick on which LEDs are mounted as shown in video.

18. Mouse positioning with LED's

Using the available IP for interfacing with the mouse, write a VHDL code to interact with the mouse in a way that a horizontal scroll in the mouse makes different LEDs to glow. If this is implemented quickly, you can add functionality using clicks.
This involves using readymade libraries for USB communication for communicated to USB Device with FPGA Board as host.

19. 7-up/7-down game

This game can be played by two players, each player is given a push button to indicate what number he is betting on either >7 or <7. Display this using two leds corresponding to each player (turn on if bet is >7 else turn off). These leds should be nearby (for ease of demonstration) the leds indicating which player won the bet. Also the players will indicate the amount they are betting using switches. Decide yourself how much amount you can allow to bet based on the number of free switches you have for input. Each player can bet amount that is less than equal to his wallet balance. You can give them initial wallet sum of 1000rs (or whatever you want). When a player wins the bet, he gets the amount from the losing player's wallet. Implement a pseudo random number generator that generates a random number in the range 2 to 12 when a pushbutton is pressed. The player who wins the bet gets the wallet amount that he did bet. If a player is left with nothing to bet, the game ends. After each bet, first display the result of random number generator and then display which player won the bet using led corresponding to that player and also indicate his total amount at the end of that bet on 7-segment display.

20. ALU designing (Integer unit, signed)

Design a multi-functional ALU.

1. The ALU must have 2 inputs "X" and "Y" of 32-bit each and a 32-bit output "Z".
2. The ALU can perform any one of the following operations ,
    - I. Addition
    - II. Substraction
    - III. Comparison
    - IV. Logical Shift (both right and left shift only by 1 bit)
    - V. 32-bit Logical operations ADD, OR, XOR

Design a separate module for all operations (using arithmetic and logical symbols are not allowed). You can assign opcode for each operation. Add 4 register in your design of 32-bit each, which can be used to load inputs in the ALU or to store ALU output.

In addition, add a 1-bit Zero flag in your ALU design, which becomes high if the output of the ALU is zero.

Try to write separate testbench for each module.

21. Vending Machine

Design and construct a logic circuit for a vending machine. The machine sells chocolate and cookies, chocolate for Rs. 20/- each and cookies for Rs. 50/-. Assume the you have an option of inserting 10/-, 20/-, 50/- and 100/- rupee note. Consider each switch as one note.

Chocolates and cookies can be denoted by LED's, along with this the change to be returned to the customer can also be denoted by LED's. The machine must be able to glow the LED if inserted amount is lesser than the cost.

There has to be a fixed quantity of each cookies and chocolate in the machine, if anyone of these is required to be refilled by the owner of the machine the machine will glow an indicator (LED's can be used for it).

22. UART(Universal Asynchronous Receiver Transmitter):

Establish a UART protocol between PC and FPGA to perform serial communication. You can write your own code for UART.

UART transmit bytes of data sequentially one bit at a time from source and receive the byte of data at the destination by decoding sequential data with control bits. As the entire processes require no clock input from source hence it is termed as asynchronous communication. In the UART communication data transmission speed is measured by Baud Rate. Baud rate describes the total number of bit sent through serial communication. It includes Start bit, Data byte, Parity bit and Stop bit. Transmitter and receiver need to be maintained at the same baud rate. Design and implement a hardware (on the Basys3 board) to transmit data at the baud rate of 9600, 8 data bits, 1 stop bit, and no parity. Also, at the receiving end PC need to be set with same settings (using a HyperTerminal or Putty or minicom).

Important points to be taken care of, while performing communication the characters are sent in their ASCII format.

23. CAM (Content addressable memory) :

It is a special type of memory used for high speed searching applications. Unlike standard computer memory (RAM - random access memory) in which the user supplies a memory address and the RAM returns the data word stored at that address, a CAM is designed such that the user supplies a data word and the CAM searches its entire memory *in parallel* to see if that data word is stored anywhere in it. If the data word is found, the CAM returns a list of one or more storage addresses where the word was found . It also returns the contents of that storage address, or other associated pieces of data.
For Demonstration
You can use switches present on FPGA to give input (eg. Name (1 character) can be a dataword and it can be entered using 8 switches present on FPGA board). For Output use seven segment display.

24. Pipelined sorting

To implement Sorting of N 8bit numbers using pipelined arrangement of multiple Processing Elements.
i.e A sequential Block that accepts data serially in N clock cycle and
outputs the Sorted data in the next N clock cycle
The input to the SORTING block would be given through the BRAM using .coe file.
The output from the SORTING Block would be displayed on the LEDs on the board.
Structure:
1. N Basic blocks within SORTING block.
2. Each of which can hold a number within itself.
3. When a new number comes in the I/P port it would compare it with the number stored within it and decide which number to give as o/p.
4. Placing N such blocks and giving data serially would result in the data being sorted and would be stored within the N basic blocks .This would take N Clk cycles
5. To flush the data out we can input the maximum value say 255 (for 8 bit data) for next N clock Cycles and hence all the values stored within the Basic blocks would be given out (of the Nth Basic Block) and the same shall be displayed on the LEDs.