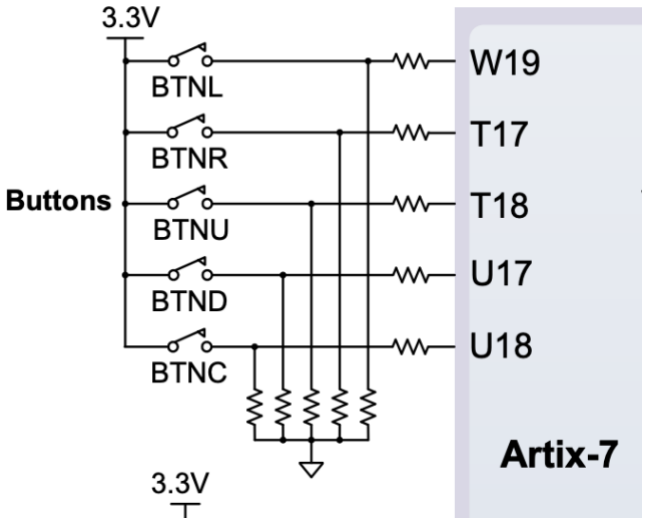
1. From the circuit diagram, the BTNx is active High or active Low? Please provide your analysis.

   Ans. **Active High.** Because from the description and diagram, they generate a low output when at rest (not pressed) and a high output only when they are pressed.

8 Basic I/O

The Basys 3 board includes sixteen slide switches, five push buttons, sixteen individual LEDs, and a four-digit seven-segment display, as shown in Fig. 16. The pushbuttons and slide switches are connected to the FPGA via series resistors to prevent damage from inadvertent short circuits (a short circuit could occur if an FPGA pin assigned to a pushbutton or slide switch was inadvertently defined as an output). The five pushbuttons, arranged in a plus-sign configuration, are "momentary" switches that normally generate a low output when they are at rest, and a high output only when they are pressed. Slide switches generate constant high or low inputs depending on their position.

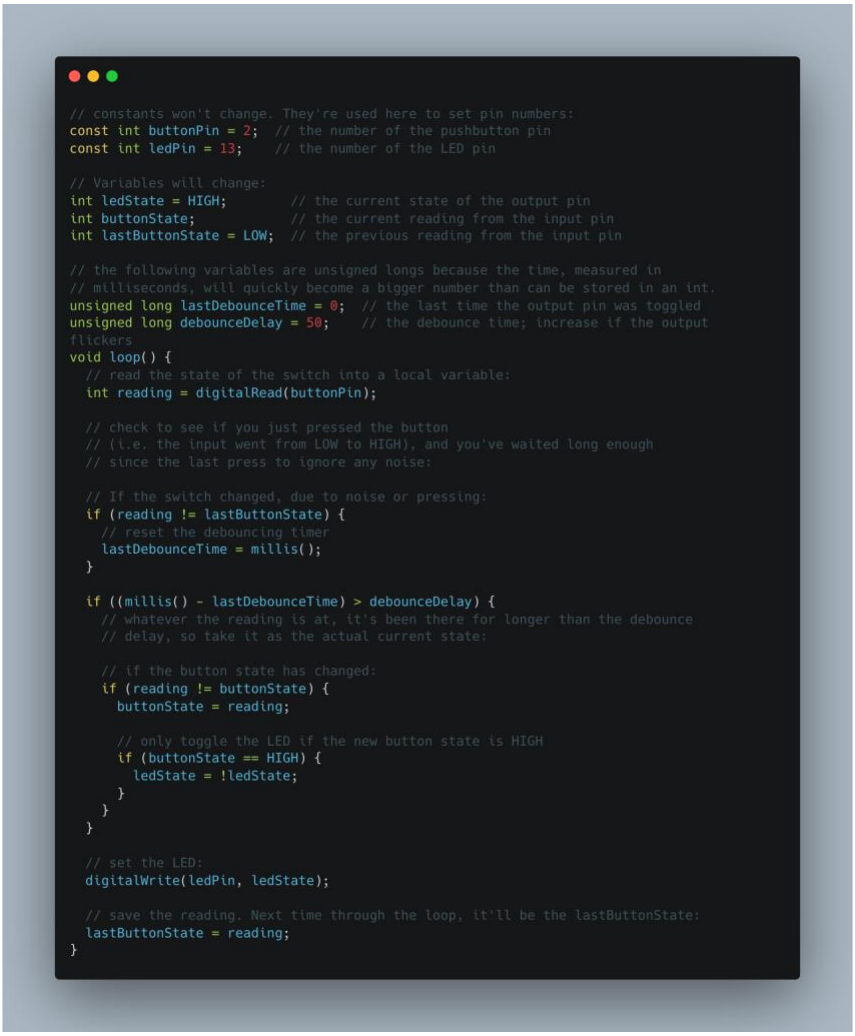


2. What is a bounce? How do you programmatically debounce the input? Please provide your analysis.

   Ans.

   When we press a button, two metal parts come into contact to short the supply. But they don't connect instantly but the metal parts connect and disconnect several times before the actual stable connection is made. It's also happening while releasing the button. This results the false triggering or multiple triggering like the button is pressed multiple times. It's like falling a bouncing ball from a height and it keeps bouncing on the surface, until it comes at rest.

   To debounce the input, we have defined a time variable. Then check that if the signal remains the same value continuously. until the specified time has elapsed. That means the signal is stable enough to read. But if not, wait until the signal is stable.



```
// constants won't change. They're used here to set pin numbers:
const int buttonPin = 2;  // the number of the pushbutton pin
const int ledPin = 13;    // the number of the LED pin

// Variables will change:
int ledState = HIGH;        // the current state of the output pin
int buttonState;            // the current reading from the input pin
int lastButtonState = LOW;  // the previous reading from the input pin

// the following variables are unsigned longs because the time, measured in
// milliseconds, will quickly become a bigger number than can be stored in an int.
unsigned long lastDebounceTime = 0;  // the last time the output pin was toggled
unsigned long debounceDelay = 50;    // the debounce time; increase if the output
flickers
void loop() {
  // read the state of the switch into a local variable:
  int reading = digitalRead(buttonPin);

  // check to see if you just pressed the button
  // (i.e. the input went from LOW to HIGH), and you've waited long enough
  // since the last press to ignore any noise:

  // If the switch changed, due to noise or pressing:
  if (reading != lastButtonState) {
    // reset the debouncing timer
    lastDebounceTime = millis();
  }

  if ((millis() - lastDebounceTime) > debounceDelay) {
    // whatever the reading is at, it's been there for longer than the debounce
    // delay, so take it as the actual current state:

    // if the button state has changed:
    if (reading != buttonState) {
      buttonState = reading;

      // only toggle the LED if the new button state is HIGH
      if (buttonState == HIGH) {
        ledState = !ledState;
      }
    }
  }

  // set the LED:
  digitalWrite(ledPin, ledState);

  // save the reading. Next time through the loop, it'll be the lastButtonState:
  lastButtonState = reading;
}
```

3. Please show your method for implementing a single pulser. (e.g. draw a state diagram, or verilogHDL code)

```
module singlePulser(
   input pushed,
   input clk,
   output reg out
   );

   reg state; // 0-unpushed 1-pushed and still pushed

   initial state=0;

   always @(posedge clk) begin
      out=0;
      case({pushed,state})
         2'b00: ; // do nothing
         2'b01: state=0; // reset state to 0
         2'b10: begin state=1; out=1; end // out to 1 only one clock
         2'b11: ; // do nothing
      endcase
   end

endmodule
```