Lab Exercise: Understanding TMS and TCK in JTAG

Objective

This lab aims to provide hands-on experience with the **JTAG Test Access Port (TAP)** controller, focusing on the role and importance of **TMS (Test Mode Select)** and **TCK (Test Clock)** signals in managing state transitions. By the end of this exercise, you will understand how these signals control state transitions in a finite state machine (FSM) and their role in debugging, testing, and programming hardware.

Introduction

The **JTAG (Joint Test Action Group)** standard is widely used for testing and debugging embedded systems. The TAP controller, a core part of JTAG, operates as a finite state machine. Two essential signals, **TMS** and **TCK**, play a pivotal role:

- **TMS (Test Mode Select)**: Determines the state transition path within the FSM.
- **TCK (Test Clock)**: Synchronizes state transitions by providing clock pulses.

In this lab, you will:

- Implement a TAP controller simulation.
- Explore how TMS and TCK interact to control state transitions.
- Visualize the FSM states using LEDs.

Materials Required

1. **Arduino Uno** or compatible microcontroller.
2. 2 push buttons.
3. 5 LEDs.
4. 7 resistors:
    - 2 x **1kΩ** pull-down resistors for buttons.
    - 5 x **1kΩ** current-limiting resistors for LEDs.
5. Breadboard and jumper wires.
6. Arduino IDE installed on your computer.

Circuit Setup

Component	Pin/Connection 1	Pin/Connection 2	Notes
Button (TCK)	One leg to Pin 2 (TCK)	Other leg to 1kΩ resistor and **GND**	Connect a 1kΩ pull-down resistor between the TCK pin (2) and **GND**.
Button (TMS)	One leg to Pin 3 (TMS)	Other leg to 1kΩ resistor and **GND**	Connect a 1kΩ pull-down resistor between the TMS pin (3) and **GND**.
LED 1	Longer leg to Pin 8	Shorter leg to 1kΩ resistor and **GND**	Represents `TEST_LOGIC_RESET`. Add a 1kΩ resistor in series with **GND**.
LED 2	Longer leg to Pin 9	Shorter leg to 1kΩ resistor and **GND**	Represents `RUN_TEST_IDLE`. Add a 1kΩ resistor in series with **GND**.
LED 3	Longer leg to Pin 10	Shorter leg to 1kΩ resistor and **GND**	Represents `SELECT_DR_SCAN`. Add a 1kΩ resistor in series with **GND**.
LED 4	Longer leg to Pin 11	Shorter leg to 1kΩ resistor and **GND**	Represents `CAPTURE_DR`. Add a 1kΩ resistor in series with **GND**.
LED 5	Longer leg to Pin 12	Shorter leg to 1kΩ resistor and **GND**	Represents `UPDATE_IR`. Add a 1kΩ resistor in series with **GND**.
Arduino GND	All GND connections		Ensure all GND points are connected to the Arduino's GND pin.

Arduino Code

Complete TAP Controller Code

```
#define TCK_PIN 2 // Test Clock
#define TMS_PIN 3 // Test Mode Select

// LED Pins for representing states
#define LED_1 8  // TEST_LOGIC_RESET
#define LED_2 9  // RUN_TEST_IDLE
#define LED_3 10 // SELECT_DR_SCAN
```

```
#define LED_4 11 // CAPTURE_DR
#define LED_5 12 // UPDATE_IR

enum TAP_State {
  TEST_LOGIC_RESET,
  RUN_TEST_IDLE,
  SELECT_DR_SCAN,
  CAPTURE_DR,
  SHIFT_DR,
  EXIT1_DR,
  PAUSE_DR,
  EXIT2_DR,
  UPDATE_DR,
  SELECT_IR_SCAN,
  CAPTURE_IR,
  SHIFT_IR,
  EXIT1_IR,
  PAUSE_IR,
  EXIT2_IR,
  UPDATE_IR
};

TAP_State currentState = TEST_LOGIC_RESET;
bool lastTckState = LOW; // To track the previous state of TCK

void setup() {
  pinMode(TCK_PIN, INPUT);
  pinMode(TMS_PIN, INPUT);

  pinMode(LED_1, OUTPUT);
  pinMode(LED_2, OUTPUT);
  pinMode(LED_3, OUTPUT);
  pinMode(LED_4, OUTPUT);
  pinMode(LED_5, OUTPUT);

  Serial.begin(9600);
  Serial.println("Starting JTAG TAP Controller Simulation");
  indicateState(currentState);
}

void loop() {
  bool tms = digitalRead(TMS_PIN);
  bool tck = digitalRead(TCK_PIN);

  Serial.print("TMS: ");
  Serial.print(tms);
  Serial.print(" | TCK: ");
  Serial.println(tck);
```

```
if (tck == HIGH && lastTckState == LOW) {
  Serial.println("Rising edge detected on TCK");

  Serial.print("Previous State: ");
  Serial.println(stateToString(currentState));

  switch (currentState) {
    case TEST_LOGIC_RESET:
      currentState = tms ? TEST_LOGIC_RESET : RUN_TEST_IDLE;
      break;
    case RUN_TEST_IDLE:
      currentState = tms ? SELECT_DR_SCAN : RUN_TEST_IDLE;
      break;
    case SELECT_DR_SCAN:
      currentState = tms ? SELECT_IR_SCAN : CAPTURE_DR;
      break;
    case CAPTURE_DR:
      currentState = tms ? EXIT1_DR : SHIFT_DR;
      break;
    case SHIFT_DR:
      currentState = tms ? EXIT1_DR : SHIFT_DR;
      break;
    case EXIT1_DR:
      currentState = tms ? UPDATE_DR : PAUSE_DR;
      break;
    case PAUSE_DR:
      currentState = tms ? EXIT2_DR : PAUSE_DR;
      break;
    case EXIT2_DR:
      currentState = tms ? UPDATE_DR : SHIFT_DR;
      break;
    case UPDATE_DR:
      currentState = tms ? SELECT_DR_SCAN : RUN_TEST_IDLE;
      break;
    case SELECT_IR_SCAN:
      currentState = tms ? TEST_LOGIC_RESET : CAPTURE_IR;
      break;
    default:
      break;
  }

  indicateState(currentState);
  Serial.print("New State: ");
  Serial.println(stateToString(currentState));
}

lastTckState = tck;
```

```
  delay(50);
}

void indicateState(TAP_State state) {
  digitalWrite(LED_1, LOW);
  digitalWrite(LED_2, LOW);
  digitalWrite(LED_3, LOW);
  digitalWrite(LED_4, LOW);
  digitalWrite(LED_5, LOW);

  switch (state) {
    case TEST_LOGIC_RESET:
      digitalWrite(LED_1, HIGH);
      break;
    case RUN_TEST_IDLE:
      digitalWrite(LED_2, HIGH);
      break;
    case SELECT_DR_SCAN:
      digitalWrite(LED_3, HIGH);
      break;
    case CAPTURE_DR:
      digitalWrite(LED_4, HIGH);
      break;
    case UPDATE_DR:
      digitalWrite(LED_5, HIGH);
      break;
    default:
      break;
  }
}

String stateToString(TAP_State state) {
  switch (state) {
    case TEST_LOGIC_RESET: return "Test-Logic-Reset";
    case RUN_TEST_IDLE: return "Run-Test/Idle";
    case SELECT_DR_SCAN: return "Select-DR-Scan";
    case CAPTURE_DR: return "Capture-DR";
    case SHIFT_DR: return "Shift-DR";
    default: return "Unknown State";
  }
}
```

Lab Procedure

Step 1: Set Up the Circuit

1. Assemble the components on a breadboard as per the circuit table.
2. Connect buttons to pins `2` (TCK) and `3` (TMS), ensuring 1kΩ pull-down resistors are in place.
3. Connect LEDs to pins `8-12` with 1kΩ current-limiting resistors.
4. Ensure all GND connections return to the Arduino's GND.

Step 2: Upload the Code

1. Open the Arduino IDE and paste the provided code.
2. Select the correct board and port.
3. Upload the code to your Arduino.

Step 3: Observe Behavior

1. Open the Serial Monitor (set baud rate to `9600`).
2. Press the `TCK` button and observe:
    - State transitions printed in the Serial Monitor.
    - LEDs lighting up corresponding to the current state.
3. Use the `TMS` button to influence transitions and explore different paths in the FSM.

Expected Results

1. Pressing `TCK` advances the TAP controller through states.
2. The `TMS` button changes the state transition path.
3. LEDs light up based on the current state, and the Serial Monitor logs transitions.

Discussion Questions

1. What role does the TMS button play in the state transitions?
2. Why is a pull-down resistor necessary for TCK and TMS?
3. How could this FSM be extended to include more states or additional functionality?

Conclusion

In this lab, you successfully implemented and tested a TAP controller simulation using an Arduino. You learned about the importance of TMS and TCK in controlling state transitions and gained practical experience with finite state machines in embedded systems.