# Indian Institute of Technology (IIT-Kharagpur)

**AUTUMN Semester, 2025**
**COMPUTER SCIENCE AND ENGINEERING**

**Computer Organization and Architecture Laboratory**
**Verilog Assignment 8: Sequential Booth Multiplier**

September 9, 2025

***Instructions:*** *This is an individual assignment. You must demonstrate your working design on the Nexys4 DDR board to a TA for evaluation. Submit a single zipped folder named `VerilogAssignment8_<Your_Roll_No>.zip` containing all well-commented Verilog source files and the XDC constraints file used for pin mapping. Ensure each source file includes a header with your name and roll number.*

## Question

### Objective

Design and implement a synchronous, sequential 8-bit signed multiplier using Booth's Multiplication Algorithm on the Nexys4 DDR FPGA board. The multiplicand and multiplier values will be provided via slide switches, and the final product will be displayed on the user LEDs.

### Hardware Mapping

- **Inputs:**
  - 'clk': The 100MHz system clock from the board.
  - 'reset': An active-high asynchronous reset. Map this to a pushbutton on the board.
  - 'multiplicand[7:0]': An 8-bit signed number (2's complement). Map this to switches 'SW7' down to 'SW0'.
  - 'multiplier[7:0]': An 8-bit signed number (2's complement). Map this to switches 'SW15' down to 'SW8'.
- **Output:**
  - 'product[15:0]': The final 16-bit signed product. Map this to the 16 user LEDs, 'LED15' down to 'LED0'.

## Functional and Architectural Requirements

1. **Clock and Reset Functionality:**
   - The ‘clk‘ input drives all sequential elements in your design.
   - When the ‘reset‘ signal is asserted (pressed), the system must immediately initialize to a known starting state. All internal registers (accumulator, counter, etc.) must be cleared, and the control unit must return to its initial state, ready for a new multiplication.
2. **Sequential (Iterative) Design:**
   - The multiplier must NOT be a purely combinational design. A design that computes the product in a single clock cycle will be awarded zero marks.
   - Your architecture must be partitioned into a datapath and a control path (implemented as a Finite State Machine - FSM).
   - The datapath contains the hardware units that store and manipulate data (e.g., registers for A, Q, M; an adder/subtractor; a shifter).
   - The control path generates the sequence of control signals that command the datapath modules to execute the steps of Booth’s algorithm iteratively, one step per clock cycle (or a fixed number of cycles per step).
3. **Resource Reusability:**
   - To enforce the iterative design methodology, the core datapath modules (e.g., the adder/-subtractor unit and the shifter) must be instantiated only once.
   - Your control unit must manage and reuse these single hardware instances across multiple clock cycles to complete the multiplication. For example, the same adder/subtractor will be used for every addition or subtraction step required by the algorithm.
4. **Operation and Output Hold:**
   - Upon de-assertion of reset, the control unit should begin the multiplication process using the values currently set on the switches.
   - After the required number of iterations, the algorithm will terminate. The control unit must then transition to a ”done” state.
   - In this ”done” state, the final 16-bit product must be stably held and continuously displayed on the LEDs. The output should not change until the system is reset again.

The standard flowchart for Booth’s Algorithm is provided in Figure 1 for your reference.

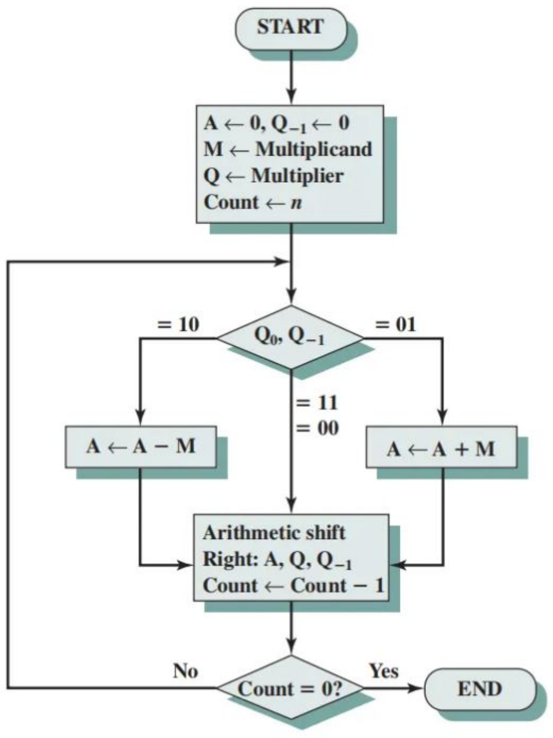

Figure 1: Booth's Algorithm Flowchart