**Sequential Multiplier**

**7D**

The given Verilog module is a seven-portion show (SSD) decoder with a multiplexing highlight. It takes two information signs, switch and in, and produces two result signs, fragments, and anode active. A breakdown of the module's capabilities is as follows: For the inputs toggle with two bits, the seven-segment display's active anode is determined by this signal. four bits have the input value that will be displayed on the SSD. Signal controls the segments of the SSD, indicating which segments should be turned on to display the desired input value. anode active (4 bits): This signal activates the appropriate anode of the SSD to display the output. Regarding behavior, the module has two always blocks that specify the behavior of the outputs based on the inputs. The first always @ (in) block: This block triggers whenever the in-signal changes. It uses a case statement to map the input value (in) to the corresponding segments that need to be turned on to display the desired value. Each case corresponds to a decimal digit (0 to 9) or a specific value (10). For example, when in is 0, the segments output is set to 7'b0000001, which activates segments a and b of the seven-segment display. The default case (when none of the specified values match) sets segments to 7'b11111111, turning off all the segments of the display. The second always @ \* block: This block triggers whenever any of the input signals change (toggle in this case). It uses a case statement to determine which anode of the seven-segment display should be active based on the toggle value. Each case corresponds to a specific toggle value (0 to 3). For example, when toggle is 0, the anode active output is set to 4'b1110, activating the first anode of the SSD and deactivating the rest. There is no default case specified, so if the toggle value is outside the defined range, the anode active output will retain its previous value.

**Binary to BCD**

The given Verilog module addresses a parallel to-BCD (Twofold Coded Decimal) converter. It generates five BCD output signals, D0, D1, D2, D3, and D4, from a binary input of 16 bits (Binary). An explanation of the module's capabilities can be found here: Inputs: 16-bit binary: This sign addresses the parallel worth to be changed over completely to BCD. For the outputs D0, D1, D2, D3, and D4 (each five bits), the BCD equivalent of the input binary value is represented by these signals. Regarding Internal indicators: Decimal: Twenty bits Throughout the conversion, this signal stores the intermediate BCD value. The Behavior of the module has one generally @(Binary) block, which triggers at whatever point the Paired information changes. Using a technique known as the double-dabble algorithm, it converts binary data into BCD. Within the always block, the beginning of each trigger, the Decimal signal is reset to zero (Decimal = 0). Each bit of the binary input is iterated over in reverse order using a for loop (I range from 0 to 15). The BCD conversion is carried out inside the loop by means of a series of conditional statements. Three bits are added to the least significant four bits of Decimal [3:0] when those four bits are greater than or equal to 5. The same procedure is followed for the remaining four groups of four bits Decimal [11:8], Decimal [15:12], and Decimal [19:16], and so on. This step guarantees that BCD digits more noteworthy than or equivalent to 5 are accurately changed by adding 3. In the end, the value in Decimal is changed by moving it one bit to the left and making the least important bit the current bit in Binary. As the loop progresses, the converted BCD digits in Decimal are effectively accumulated by this. After the consistently block, the BCD digits are allocated to their results. The four decimal bits D0 are the least important Decimal [3:0]. D1 is assigned to the following four pieces Decimal [7:4]. The following four bits are assigned to D2 Decimal [11:8]. D3 is allocated the following four pieces Decimal [15:12]. D4 receives the four bits with the greatest significance Decimal [19:16]. In summary, this Verilog module uses the double-dabble algorithm to implement a binary-to-BCD converter. It switches a 16-bit double contribution over completely to five separate BCD yields (D0 to D4), permitting the portrayal of decimal digits in a BCD design.

**Counter**

A counter with configurable parameters is represented by the Verilog module provided. It counts from 0 to a predetermined value and returns to 0 upon assertion of a reset signal. An explanation of the module's capabilities can be found here. The Parameters are x (the default is: 4) is used to represent the counter value is determined by this parameter. n (default esteem: 10) before resetting, this parameter determines the maximum count value. For the inputs, clk is the clock signal that sets off the counter on the positive edge. Reset is utilized to reset the counter to 0 when declared. En works when the counter is active, it can increment thanks to this signal. For the outputs, count works with a width of x bits, this signal indicates the counter's current value. For the Internal indicators, the counter's current value is stored in this register using the count parameter. The behavior of the module has one generally @ (posedge clk, posedge reset) block, which triggers on the positive edge of the clock sign or when the reset signal is declared. Inside the always block if the reset signal is 1, showing a reset demand, the count register is set to 0 (count <= 0) and the counter can go up if the en signal is active (non-zero). Assuming that the ongoing worth of count is equivalent to n-1, demonstrating the most extreme count esteem has been reached, the count register is hindered to 0 to reset the counter (count <= 0). In any case, if the greatest count esteem has not been reached, the count register is augmented by 1 (count <= count + 1). Based on the clock and input signals, the module continuously updates the count register, allowing the counter to increment or reset in accordance with the defined conditions. In conclusion, this Verilog module executes a counter with configurable boundaries. It counts from 0 to a predefined esteem (n) and resets back to 0 when a reset signal is stated. The counter additions on the positive edge of the clock signal when empowered (en signal). The ongoing include esteem is put away in the count register, which is x pieces wide.

**Debouncer**

It produces a debounced output signal (out) from an input signal (in). The debouncer is intended to eliminate clamor or errors from the info signal and give a steady result. For the inputs, clk is the clock signal that starts the debouncer on the positive edge is this one. Rst is utilized to reset the debouncer to its underlying state when affirmed. In is the information sign to be debounced. For the outputs, out addresses the debounced yield. During the debouncing process, the intermediate values are stored in these registers q1, q2, q3. Regarding behavior, the module has one always @ (posedge clk, posedge rst) block that fires when the reset signal is asserted or the positive edge of the clock signal. Inside the always block, rst signal is stated (high), demonstrating a reset demand, every one of the three registers (q1, q2, q3) are set to 0, successfully resetting the debouncer. The debouncing process begins if the initial signal is not asserted (low) as q1 is given the value of the input signal (in). The input signal is effectively delayed by one clock cycle when the value of q1 is assigned to q2. The worth of q2 is appointed to q3, further deferring the info signal by two clock cycles. An assign statement is used to assign the out signal based on the debounced values following the always block, the out signal is set to 0 if the first signal is asserted. If the rst signal isn't tested, the out signal is given the legitimate AND activity of q1, q2, and q3. This indicates that a stable and debounced input signal indicates that the out signal will be high only if all three registers have high values simultaneously. In conclusion, this Verilog module uses a debouncer circuit to get rid of glitches or noise in an input signal. It compares the values of three registers to provide a stable output signal after delaying the input signal. The debouncing process takes place on the clock signal's positive edge, and the debouncer can be reset to its initial state with a reset signal.

**SM**

An unsigned multiplier is represented by the Verilog module provided. It takes an 8-digit multiplier (Multiplier) and multiplicand (Multiplicand) as sources of info and produces a 16-bit result (resultant) of the increased activity. A done signal is also provided by the module to let you know when the multiplication operation is finished. An explanation of the module's capabilities can be found here. 8-bit multiplier addresses the multiplier for the increase activity. The multiplication operation is represented by Multiplicand. Load represents when new values should be loaded into the multiplier, multiplicand, and accumulator. In conclusion, an unsigned multiplier is implemented by this Verilog module. It performs the multiplication operation by accumulating the result in the accumulator while iteratively shifting the multiplier and multiplicand. When the multiplication is finished, the done signal indicates it, and the resultant signal stores the result.

**SS driver**

The given Verilog module addresses an SSD (Seven-Section Show) driver with extra usefulness for showing different BCD (Parallel Coded Decimal) values in view of a switch empower signal and a sign. Inputs are sign which addresses the sign worth, utilized to decide the showed esteem on the SSD. Clk for internal operations clock signal. The module is reset to its initial state with this signal. The SSD's BCD values are represented by these signals BCD1, BCD2, and BCD0. Modules that were created are divider module (clock divider) that separates the info clock recurrence to create a slower clock signal (flashing Clock). The toggling feature is made possible by this instance of a counter module (Counter) that counts from 0 to 3, a seven-section decoder module (SD) that translates the BCD values into portions to show on the SSD. In summary, this Verilog module is a driver for a Seven-Segment Display (SSD) with additional functionality to display different BCD values based on a toggle enable signal and a sign signal. It uses a clock divider and a counter to control the toggling functionality, and a Seven-Segment Decoder (SD) module to decode the BCD values into segments for display on the SSD.

**Shift register**

The given Verilog module addresses a shift register with a selectable information. The three outputs of the shift register—S0, S1, and S2—store the values of the input signals (D0, D1, D2, D3, and D4) in accordance with the input that has been selected. The shift register can be controlled utilizing clock (clk) and reset (rst) signals, as well as compose control signals (w1 and w2). D0, D1, D2, D3, and D4 are the data that needs to be entered into the shift register is represented by these signals. The shift register's clock signal is represented by this sign (clk) and the shift register is reset to its initial state with this signal.

.

**Sign calculator**

A sign calculator for two input numbers is represented by the Verilog module. It determines the sign of the result and calculates the absolute values of the input numbers (num1 and num2). An explanation of the module's capabilities can be found here: Numbers 1 and 2 (eight bits) are the numbers that will be used to calculate the sign and absolute values. The sign of the result is represented by this signal. If the numbers have different signs (one is positive, and the other is negative), it will be 1 and 0 respectively. If the numbers have the same sign, it will be 0. The result of the XOR operation between the input numbers' most significant bits (num1[7] and num2[7]) is assigned to the sign output. The results of this XOR operation are used to see if the numbers have different signs. The sign output will be 1 if they do; If not, it will be 0. A conditional expression is used to assign the absolute value of num1 to the num1\_abs output. If num1[7] (the sign piece) is 0 (showing a positive number), num1\_abs will be relegated the worth of num1 itself. Otherwise, the bitwise complement operator is used to calculate the two's complement of num1 and add 1, and the result is assigned to num1\_abs if num1[7] is 1 (indicating a negative number). The negative number is effectively transformed into the positive number. A conditional expression is used to assign the absolute value of num2 to the num2\_abs output in a similar manner. num2\_abs will be assigned num2 if num2[7] is 0; Otherwise, the two's complement of num2 is calculated and assigned to num2\_abs if num2[7] is 1. Finally, this Verilog module determines the absolute values and the sign of two input numbers. The positive equivalents of the input numbers are provided by num1\_abs and num2\_abs, while the sign output indicates whether the input numbers have distinct signs.

**Synchronizer**

The given Verilog module addresses a synchronizer circuit, which is utilized to synchronize an offbeat sign (sig) to a clock space (clk). The signal sig needs to be synchronized with this, which is the clock signal of the clock domain. The information signal that should be synchronized to the clock area is done through sig. The synchronizer circuit is reset with this synchronous reset signal. The synchronizer circuit is reset when the reset signal rst is asserted (a positive edge). Assuring a known initial state, the META register and the sig1 output are both set to 0 (1'b0). The input signal sig is synchronized with the clock domain when the reset signal is not asserted, indicating that the circuit is not in a reset state. The worth of sig is caught in the META register on the positive edge of clk, guaranteeing appropriate synchronization. The output signal sig1, which represents the synchronized version of sig, is then given the synchronized value that is stored in META. In conclusion, this Verilog module implements a fundamental synchronizer circuit that synchronizes an asynchronous input signal, sig, with a clock domain that is specified by clk. The synchronized result signal sig1 addresses the synchronized rendition of sig in the clock space of clk. To ensure a known initial state for the synchronizer circuit, the module also includes synchronous reset functionality.

**Clock divider**

A clock divider circuit that divides the frequency of an input clock signal (clk) to generate an output clock signal (clk\_out). The parameter n determines the frequency division factor. Clock signal, or clk is the input clock signal that requires division is this one. Reset signal is used to reset the clock divider circuit. clk\_out is the signal that addresses the separated adaptation of the information clock. Parameters are n which is the recurrence division factor. It figures out how many clock cycles are necessary to produce one cycle of the divided clock signal. Variables are count (register with 32 bits) which monitors the ongoing count an incentive for recurrence division. To summarize, this Verilog module uses a counter (count) to keep track of the clock cycles and toggles the output clock signal (clk\_out) at the desired frequency division points to implement a clock divider circuit that divides the frequency of an input clock signal by the factor specified by the parameter n. To ensure a known initial state for the counter and output clock signal, the module also includes asynchronous reset functionality.

**Push button detector**

The given Verilog module addresses a pushbutton identifier circuit. Using a variety of sub-modules, it finds the rising edge of a pushbutton signal (w). The pushbutton detection circuit uses a reference for the input clock signal. The circuit is reset with this asynchronous reset signal. W is the pushbutton's input signal that needs to be found. Z is the result signal showing the identification of a rising edge on the pushbutton signal. Moving forward to the sub modules, the clock divider module produces a divided version of the input clock signal. The Debouncer module synchronizes and debounces the pushbutton signal, is demonstrated here. The Synchronizer module further synchronizes the debounced pushbutton signal in this instance. The rising edge of the synchronized pushbutton signal is detected by this instance of the rising\_edge\_detector module.

**Rising edge detector**

A rising edge detector circuit is represented by the given Verilog module. Using a finite state machine, it finds the rising edge of an input signal (w). clock is the information clock signal utilized as a source of perspective for the rising edge discovery circuit. The circuit is reset with this asynchronous reset signal. W is the rising edge needs to be identified for this input signal. The output signal that indicates the presence of a rising edge in the input signal is z. The register of two bits addresses the present status of the limited state machine. In conclusion, a rising edge detector is implemented by means of a finite state machine in the Verilog module. It provides the detection result on the output signal z after it has identified the rising edge of the input signal w. The rising edge detection takes place when the input signal moves from low to high.

**Contribution**

Adham: shit register and 7 segment display

Omar: buttons and sign calculator

Tarek: signed sequential multiplier.

Farida: decoder and double dabble.

Adham and Tarek were also responsible for collecting all parts of the code and putting it on GitHub and Omar and Farida were responsible for collecting report parts and putting them on GitHub