perplexity

give a verilog code for the description given on the board

Based on the blackboard diagram, I'll create a Verilog code for the digital clock using an FPGA with a 100MHz clock and 7-segment displays.

```
module digital_clock(
    input wire clk,             // 100MHz clock from FPGA
    input wire rst,             // Reset signal
    output reg [6:0] segments,  // 7-segment display output (abcdefg)
    output reg [3:0] anodes     // Digit select outputs (AN3-AN0)
);

    // Parameters from blackboard
    // 100MHz clock, 1 minute = ~6x10^9 cycles (33 bits needed)
    parameter MINUTE_CYCLES = 33'd6_000_000_000;

    // Clock divider registers
    reg [32:0] cycle_counter;   // 33-bit counter for minute generation
    reg min_clk;                // 1-minute clock pulse

    // Time registers
    reg [5:0] minutes;          // 0-59 (6 bits as shown on board)
    reg [4:0] hours;            // 0-23 (5 bits as shown on board)

    // Display control signals
    reg [16:0] refresh_counter; // Counter for display multiplexing
    wire [1:0] active_digit;    // 2-bit value to select display digit

    // BCD digit values for display
    wire [3:0] hr_tens, hr_ones, min_tens, min_ones;
    reg [3:0] current_digit;    // Current digit value to display

    //----------------------------------------------------------------
    // 1. Minute clock generator (33-bit counter)
    //----------------------------------------------------------------
    always @(posedge clk or posedge rst) begin
        if (rst) begin
            cycle_counter &lt;= 0;
            min_clk &lt;= 0;
        end else if (cycle_counter &gt;= MINUTE_CYCLES - 1) begin
            cycle_counter &lt;= 0;
            min_clk &lt;= 1;
        end else begin
            cycle_counter &lt;= cycle_counter + 1;
            min_clk &lt;= 0;
        end
    end
```

```
//------------------------------------------------------------------
// 2. Hour Counter (H_clk module on board)
//------------------------------------------------------------------
always @(posedge min_clk or posedge rst) begin
    if (rst) begin
        hours &lt;= 0;
    end else if (minutes == 59) begin
        if (hours == 23)
            hours &lt;= 0;
        else
            hours &lt;= hours + 1;
    end
end

//------------------------------------------------------------------
// 3. Minute Counter (M_clk module on board)
//------------------------------------------------------------------
always @(posedge min_clk or posedge rst) begin
    if (rst) begin
        minutes &lt;= 0;
    end else if (minutes == 59) begin
        minutes &lt;= 0;
    end else begin
        minutes &lt;= minutes + 1;
    end
end

//------------------------------------------------------------------
// 4. Display refresh counter for multiplexing
//------------------------------------------------------------------
always @(posedge clk or posedge rst) begin
    if (rst)
        refresh_counter &lt;= 0;
    else
        refresh_counter &lt;= refresh_counter + 1;
end

// Use MSBs of refresh counter to select which digit to display
assign active_digit = refresh_counter[16:15];

//------------------------------------------------------------------
// 5. Binary to BCD conversion
//------------------------------------------------------------------
assign hr_tens = hours / 10;
assign hr_ones = hours % 10;
assign min_tens = minutes / 10;
assign min_ones = minutes % 10;

//------------------------------------------------------------------
// 6. Display multiplexer (MUX shown on board)
//------------------------------------------------------------------
always @(*) begin
    case (active_digit)
        2'b00: begin  // Hour tens digit
            current_digit = hr_tens;
            anodes = 4'b0111;  // AN3 active (active low)
            end
            2'b01: begin  // Hour ones digit
                current_digit = hr_ones;
                anodes = 4'b1011;  // AN2 active
            end
            2'b10: begin  // Minute tens digit
                current_digit = min_tens;
                anodes = 4'b1101;  // AN1 active
            end
            2'b11: begin  // Minute ones digit
                current_digit = min_ones;
                anodes = 4'b1110;  // AN0 active
            end
        endcase
    end

    //------------------------------------------------------------------
    // 7. Seven-segment decoder (LUT_H on board)
    //------------------------------------------------------------------
    always @(*) begin
        case (current_digit)
            // abcdefg encoding as shown on board
            4'd0: segments = 7'b1000000;  // 0
            4'd1: segments = 7'b1111001;  // 1
            4'd2: segments = 7'b0100100;  // 2
            4'd3: segments = 7'b0110000;  // 3
            4'd4: segments = 7'b0011001;  // 4
            4'd5: segments = 7'b0010010;  // 5
            4'd6: segments = 7'b0000010;  // 6
            4'd7: segments = 7'b1111000;  // 7
            4'd8: segments = 7'b0000000;  // 8
            4'd9: segments = 7'b0010000;  // 9
            default: segments = 7'b1111111;
        endcase
    end

endmodule
```

Implementation Notes

1. **Clock Specifications**: The design uses a 100MHz input clock from the FPGA board as shown at the top of the blackboard.
2. **Time Counters**:
   - Hours: 5-bit counter (0-23) as specified on the board
   - Minutes: 6-bit counter (0-59) as specified on the board
3. **Minute Generation**:
   - The 33-bit counter implements the "Cycle: ~6×10^9 (1 min) = 33bit" calculation shown on the board
   - This generates a pulse every minute for updating the time
4. **Display Multiplexing**:

    - The board shows a multiplexed 4-digit design with hour and minute digits
    - Anodes (AN0-AN3) control which digit is active at any time
    - Follows the multiplexer structure shown on the right side of the board
5. **7-Segment Encoding**:
    - Uses the segment pattern shown on the board (abcdefg)
    - Values for each digit (0-9) are encoded as shown in the bottom section

For actual implementation, you would need to map this code to your specific FPGA pins and possibly adjust the display refresh rate for optimal visibility.

⁂