# ENGR 850 Project Proposal- Group A.R.A

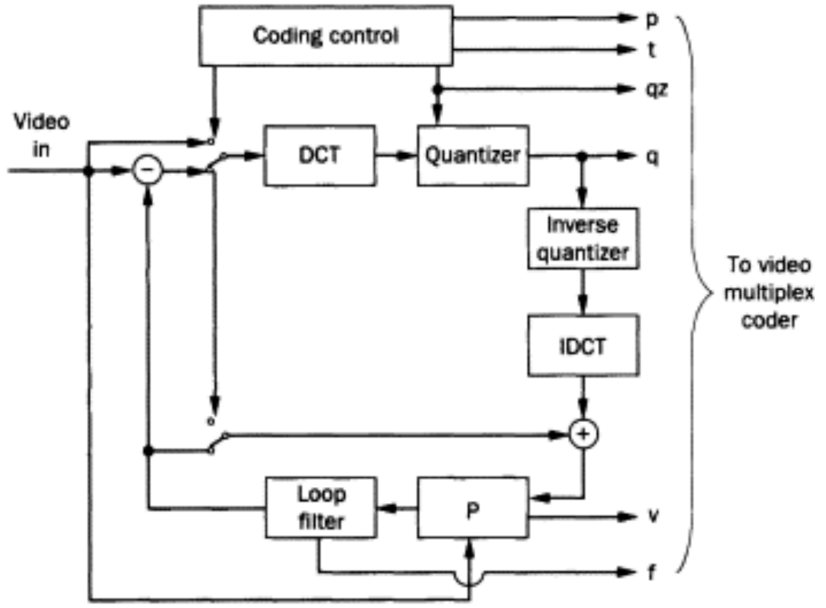
**Name of team members: Aaron, Runjun, Akshaya**

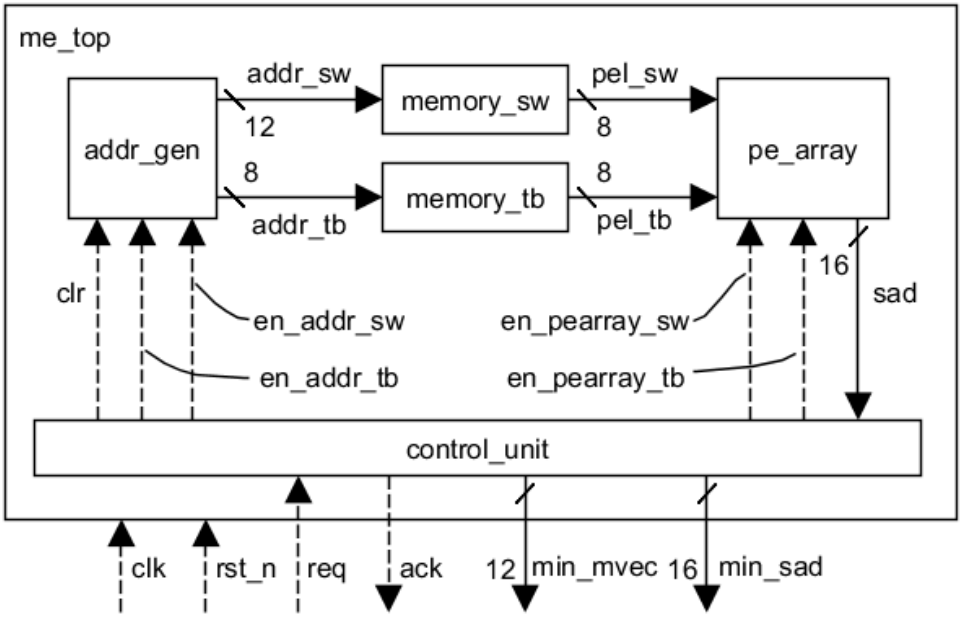
**Design to be verified: Motion Estimator**
**Project Title: Verification of Real-time Performance-Optimized Motion Estimation with Multi-level Error Masking using SystemVerilog**

**Project abstract:**
Our project aims to verify a real-time performance-optimized motion estimation algorithm with multi-level error masking using SystemVerilog. The motion estimator is crucial for video processing applications, and ensuring its real-time operation while effectively handling various errors is essential for reliable performance.Motion Estimation is a process of finding motion vectors from a reference frame to the current frame. This plays a key role in video compression and enhancement algorithms. We apply the following process for each of the macro blocks in the current frame: First, we find a block which has the lowest matching error in the search window of the reference frame. We use the sum of absolute errors between corresponding pixels for this comparison. Then, the displacement between the current macro block to the best matching macro block is the motion vector.

We will focus on implementing efficient parallelization strategies and optimizations to meet real-time processing requirements. Additionally, we will design multi-level error masking techniques to address different error sources, such as noise and occlusions, with varying impact on performance. Our verification efforts will include testing the algorithm's real-time capabilities and evaluating the effectiveness of the error masking mechanisms in different scenarios. These error prompting mechanisms will be customized for the different nature of errors encountered in the video stream, ensuring enhanced reliability and fidelity. Through comprehensive testing and evaluation, verify the performance of the algorithm under different conditions and determine whether it is suitable for video processing applications.

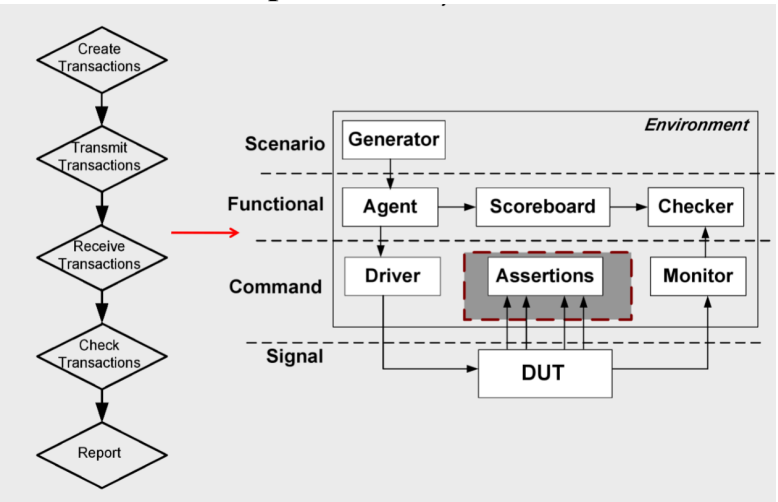
**Description of design under test:**



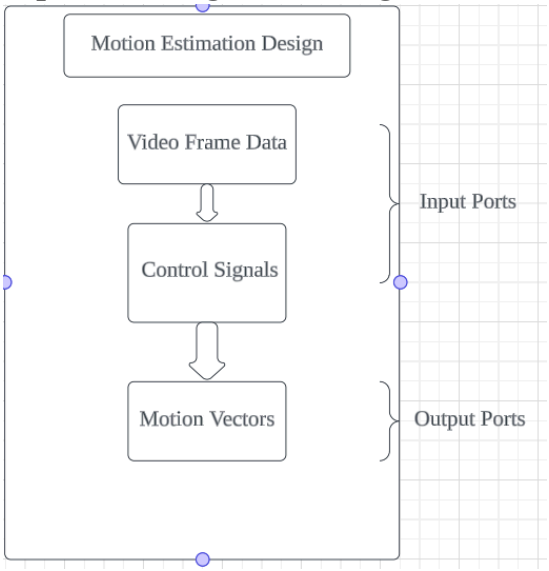



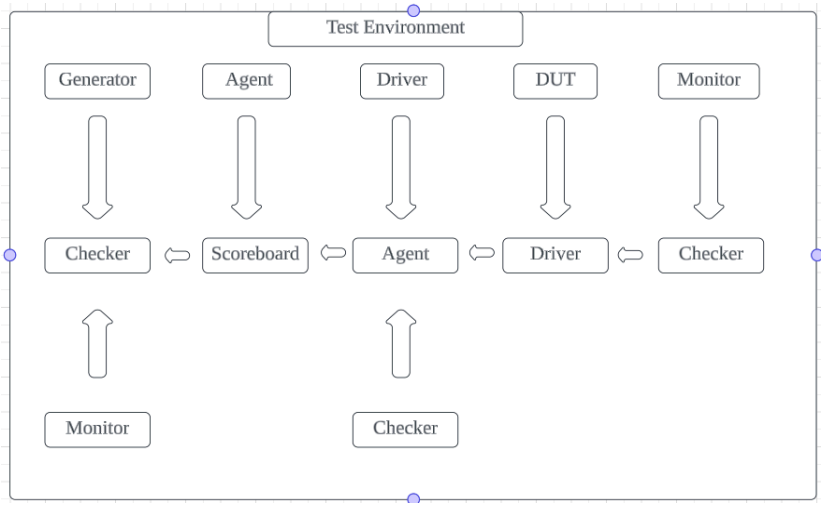

*Please include the top-level single-box diagram of the design under test showing input and output ports. Explain the function of each port.*

## Classroom Example




## Top-Level Single-Box Diagram






**Video frame data** (input port) is where the generator generates stimulus data, such as video frames, to feed into the test environment.

**Control signals** (input ports) are proxy modules that receive control signals from external sources. Configure various parameters of the motion estimation algorithm, receive stimulus data from the generator, pass it to the driver, and receive the driver's response.

**Motion vectors** (output ports) are monitor blocks that observe and capture the DUT during simulation. It captures the output produced by a motion estimation algorithm, which includes motion vectors representing macroblock displacements between frames. Monitor the output of the DUT for analysis and comparison with expected results.

**The driver module** drives stimulus data to the design under test (DUT) through input ports. Feedback can also be received from the DUT.

**The design under test** (DUT) is the actual system (motion estimation design) being tested. Receives data stimulus from the driver, processes it, and produces output data through the output port.

**The scoreboard module** can compare the output produced by the DUT with the expected results to verify correctness.

**The checker module** performs additional checks on the output data to ensure compliance with specified requirements or standards.

**Design Complexity:**

**Number of parallel single-bit input ports**: 2 (Video Frame Data, Control Signals)
(Video Frame Data:Serves as input for the pixel data of the current frame being processed. It provides the necessary information for the algorithm to analyze and compare with a reference frame.)
(Control Signals:Behavior and operation, controlling parameters such as search window size, motion estimation method, and error masking strategy)
**Number of serial ports × packet size**: 0 (Assuming no serial communication,For simplicity and efficiency, use direct tool communication)
**Number of lines of Verilog code**: Exceeds 50 (Exact count dependent on design implementation)
**The design** includes the necessary input ports for video frame data, motion search parameters, and control signals, along with output ports for motion vectors. Additionally, the **Verilog code** implementing the motion estimation algorithm contributes to the design's complexity, ensuring it meets the minimum requirement for lines of code.

**The RTL design** employs algorithms to calculate 8x8 blocks sequentially and utilizes the hexagon searching algorithm to find the best-matched block efficiently, this optimizes the motion vector search process, balancing accuracy and computational complexity. Furthermore, simultaneous **SRAM** read of the current frame and write of the previous frame optimizes processing time, ensures seamless motion estimation without noticeable delays. Pipelining the **Sum of Absolute Differences (SAD)** calculation enhances performance, breaking calculations into multiple stages allows for parallel processing and increases overall throughput. However, challenges arise in handling boundary conditions, which may lead to overflow issues. Synthesis presents difficulties in pipelining the critical path and requires iterative adjustments to the **RTL** code for successful compilation. In **APR**, **NanoRoute** may encounter cut short problems, necessitating the addition of blockage and rerouting to resolve DRC violations, to ensure compliance with design rules and optimize physical layout to improve performance and manufacturability.

‘

***Verification plan:***

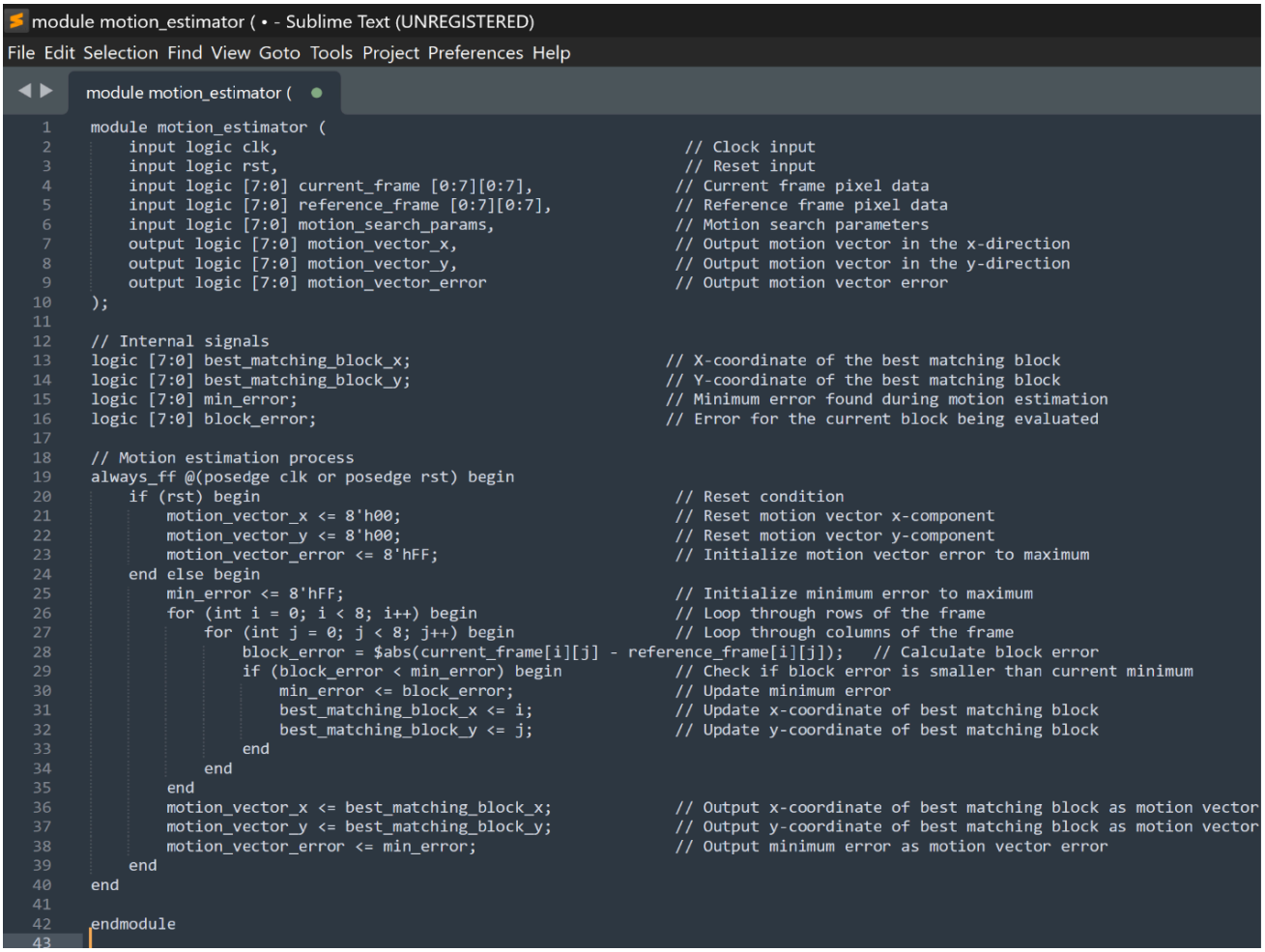
- Planning the testbench design: Deadline - **March 2nd**
- Building a simple directed test case: Deadline - **March 9th**
- Building SystemVerilog random stimulus based test: Deadline - **March 16th**
- Adding self-checking features such as assertions or model-based checking to the testbench: Deadline - **March 23rd**
- Adding test coverage definition and reporting to the testbench: Deadline - **March 30th**
- Finalizing the coverage-driven random-stimulus and self-checking based SystemVerilog testbench: Deadline - **April 6th**
- Collecting test results along with coverage report: Deadline - **April 13th**

***Project Timeline:***

- Testbench design planning: **February 16th - March 2nd**
- Simple directed test case: **March 3rd - March 9th**
- SystemVerilog random stimulus test: **March 10th - March 16th**
- Self-checking features implementation: **March 17th - March 23rd**
- Test coverage definition and reporting: **March 24th - March 30th**
- Finalization of testbench: **March 31st - April 6th**
- Test results collection and coverage report: **April 7th - April 13th**
- Report writing and presentation preparation: **April 14th - April 16th**

**Design Verilog Code:**

*Please copy and paste or attach the Verilog code of the design.*




```
module motion_estimator (
    input logic clk,                                        // Clock input
    input logic rst,                                        // Reset input
    input logic [7:0] current_frame [0:7][0:7],             // Current frame pixel data
    input logic [7:0] reference_frame [0:7][0:7],           // Reference frame pixel data
    input logic [7:0] motion_search_params,                 // Motion search parameters
    output logic [7:0] motion_vector_x,                     // Output motion vector in the x-direction
    output logic [7:0] motion_vector_y,                     // Output motion vector in the y-direction
    output logic [7:0] motion_vector_error                  // Output motion vector error
);

// Internal signals
logic [7:0] best_matching_block_x;                         // X-coordinate of the best matching block
logic [7:0] best_matching_block_y;                         // Y-coordinate of the best matching block
logic [7:0] min_error;                                     // Minimum error found during motion estimation
logic [7:0] block_error;                                   // Error for the current block being evaluated

// Motion estimation process
always_ff @(posedge clk or posedge rst) begin
    if (rst) begin                                         // Reset condition
        motion_vector_x <= 8'h00;                          // Reset motion vector x-component
        motion_vector_y <= 8'h00;                          // Reset motion vector y-component
        motion_vector_error <= 8'hFF;                      // Initialize motion vector error to maximum
    end else begin
        min_error <= 8'hFF;                                // Initialize minimum error to maximum
        for (int i = 0; i < 8; i++) begin                  // Loop through rows of the frame
            for (int j = 0; j < 8; j++) begin              // Loop through columns of the frame
                block_error = $abs(current_frame[i][j] - reference_frame[i][j]);   // Calculate block error
                if (block_error < min_error) begin         // Check if block error is smaller than current minimum
                    min_error <= block_error;              // Update minimum error
                    best_matching_block_x <= i;            // Update x-coordinate of best matching block
                    best_matching_block_y <= j;            // Update y-coordinate of best matching block
                end
            end
        end
        motion_vector_x <= best_matching_block_x;          // Output x-coordinate of best matching block as motion vector
        motion_vector_y <= best_matching_block_y;          // Output y-coordinate of best matching block as motion vector
        motion_vector_error <= min_error;                  // Output minimum error as motion vector error
    end
end

endmodule
```

**Reference Documents:**


Sumit Kumar Chatterjee, Sravan Kumar Vittapu, "FPGA implementation of EFSME for high efficient video coding standard", *Multimedia Tools and Applications*, vol.81, no.23, pp.34087, 2022.https://ieeexplore.ieee.org/document/7800438