**DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING**

**COLLEGE OF E&ME, NUST, RAWALPINDI**

**Digital System Design**

**Progress Report**

**SUBMITTED TO:**

* Dr. Yasin

**SUBMITTED BY:**

* Muhammad Ahmar Noor
* Zumar Binat Afzal

DE - 43 - CE

Syndicate: B

**Submission Date:** 7 Dec, 2024

**Task:**

**Fast/parallel/pipelined matrix of size 3x3 working on FPGA with UART, 10%**

**Code:**

///////////////////////////////////

/////////// TOP MODULE ////////////

////////////////////////////////////

module matrix\_multiplication\_with\_uart (

input clk,

input rst,

input [7:0] rx\_data,

input rx\_ready,

output reg tx\_data,

output reg tx\_status,

output [7:0] debug\_data,

output reg multiplication\_done // Add multiplication\_done signal

);

wire [7:0] matrix\_a [0:8];

wire [7:0] matrix\_b [0:8];

wire [15:0] matrix\_c [0:8];

reg start\_transmission;

// Instantiate UART receiver to receive Matrix A and Matrix B

uart\_receiver uart\_rx (

.clk(clk),

.rst(rst),

.rx\_data(rx\_data),

.rx\_ready(rx\_ready),

.matrix\_a(matrix\_a),

.matrix\_b(matrix\_b)

);

// Matrix multiplication module

matrix\_multiplication matrix\_mult (

.clk(clk),

.rst(rst),

.a(matrix\_a),

.b(matrix\_b),

.c(matrix\_c)

);

// UART transmitter to send Matrix C

uart\_transmitter uart\_tx (

.clk(clk),

.rst(rst),

.tx\_data(tx\_data),

.tx\_status(tx\_status),

.matrix\_c(matrix\_c),

.start(start\_transmission)

);

// Monitor the completion of matrix multiplication and signal multiplication\_done

always @(posedge clk or posedge rst) begin

if (rst) begin

multiplication\_done <= 0; // Reset multiplication\_done signal

end else begin

// Set multiplication\_done when multiplication is completed (assuming valid data in matrix\_c)

if (matrix\_c[8] != 16'd0) begin

multiplication\_done <= 1;

$display("Matrix multiplication is complete.");

end else begin

multiplication\_done <= 0;

end

end

end

// Display received data for debugging

always @(posedge clk or posedge rst) begin

if (rst) begin

$display("Resetting system...");

end else if (rx\_ready) begin

$display("Data received: rx\_data = %d", rx\_data);

end

end

// Control the start of transmission after multiplication is done

always @(posedge clk or posedge rst) begin

if (rst) begin

start\_transmission <= 0;

end else begin

if (multiplication\_done) begin

start\_transmission <= 1;

$display("Matrix multiplication completed, starting transmission of Matrix C...");

end else begin

start\_transmission <= 0;

end

end

end

// Debugging the transmission process

always @(posedge clk or posedge rst) begin

if (rst) begin

$display("Resetting the transmission.");

end else if (start\_transmission) begin

$display("Transmission started, sending Matrix C data...");

end else begin

$display("Waiting for transmission to start...");

end

end

// Final display to show multiplication status

always @(posedge clk or posedge rst) begin

if (rst) begin

$display("System Reset.");

end else if (multiplication\_done) begin

$display("Multiplication done, sending results.");

end

end

endmodule

/////////////////////////////////////////////////////

/////////// Matrix Multipication MODULE ////////////

/////////////////////////////////////////////////////

module matrix\_multiplication (

input clk,

input rst,

input [7:0] a[0:8], // 3x3 matrix A (1D array of 9 elements)

input [7:0] b[0:8], // 3x3 matrix B (1D array of 9 elements)

output reg [15:0] c[0:8] // 3x3 matrix C (1D array of 9 elements, 16-bit wide for result)

);

integer i, j;

always @(posedge clk or posedge rst) begin

if (rst) begin

for (i = 0; i < 9; i = i + 1) begin

c[i] <= 16'd0; // Reset matrix C to zero on reset

end

end else begin

// Matrix multiplication

for (i = 0; i < 3; i = i + 1) begin

for (j = 0; j < 3; j = j + 1) begin

c[i\*3 + j] <= a[i\*3] \* b[j] + a[i\*3 + 1] \* b[j + 3] + a[i\*3 + 2] \* b[j + 6];

end

end

end

end

endmodule

/////////////////////////////////////////////////////

/////////// Receiver MODULE ////////////

/////////////////////////////////////////////////////

module uart\_receiver (

input clk,

input rst,

input [7:0] rx\_data, // Data coming from UART

input rx\_ready, // Data is ready to be received

output reg [7:0] matrix\_a[0:8], // Matrix A received

output reg [7:0] matrix\_b[0:8] // Matrix B received

);

reg [3:0] counter\_a = 0, counter\_b = 0; // Counter for matrix\_a and matrix\_b

always @(posedge clk or posedge rst) begin

if (rst) begin

counter\_a <= 0;

counter\_b <= 0;

end else if (rx\_ready) begin

if (counter\_a < 9) begin

matrix\_a[counter\_a] <= rx\_data; // Store data in matrix\_a

counter\_a <= counter\_a + 1;

end else if (counter\_b < 9) begin

matrix\_b[counter\_b] <= rx\_data; // Store data in matrix\_b

counter\_b <= counter\_b + 1;

end

end

end

endmodule

/////////////////////////////////////////////////////

/////////// Transmitter MODULE ////////////

/////////////////////////////////////////////////////

module uart\_transmitter (

input clk,

input rst,

output reg tx\_data, // Data being sent out via UART

output reg tx\_status, // Transmission status (busy or ready)

input [15:0] matrix\_c[0:8], // Matrix C (3x3 result)

input start // Start transmission when matrix multiplication is complete

);

reg [7:0] current\_data; // Current data to send

reg [3:0] counter = 0; // Counter to track the data position

always @(posedge clk or posedge rst) begin

if (rst) begin

tx\_status <= 0;

counter <= 0;

end else if (start && counter < 9) begin

tx\_status <= 1; // Set transmission status as busy

current\_data <= matrix\_c[counter][7:0]; // Load the lower byte of matrix\_c

tx\_data <= current\_data; // Send the current data byte

counter <= counter + 1; // Move to next data point

end else begin

tx\_status <= 0; // Set transmission status as ready when done

end

end

endmodule

/////////////////////////////////////////////////////

/////////// TESTBENCH MODULE ////////////

/////////////////////////////////////////////////////

module tb\_matrix\_multiplication\_with\_uart;

// Testbench signals

reg clk;

reg rst;

reg [7:0] rx\_data;

reg rx\_ready;

wire tx\_data;

wire tx\_status;

wire [7:0] debug\_data;

wire multiplication\_done; // Add the multiplication\_done wire

// Instantiate the DUT (Device Under Test)

matrix\_multiplication\_with\_uart uut (

.clk(clk),

.rst(rst),

.rx\_data(rx\_data),

.rx\_ready(rx\_ready),

.tx\_data(tx\_data),

.tx\_status(tx\_status),

.debug\_data(debug\_data),

.multiplication\_done(multiplication\_done) // Connect multiplication\_done

);

// Clock generation

always begin

#5 clk = ~clk; // Clock period = 10 ns

end

// Testbench stimulus

initial begin

// Initialize signals

clk = 0;

rst = 0;

rx\_data = 0;

rx\_ready = 0;

// Open VCD file for waveform dump

$dumpfile("dump.vcd"); // Name of the VCD file

$dumpvars(0, tb\_matrix\_multiplication\_with\_uart); // Dump all signals in this testbench

// Apply reset

rst = 1;

#10 rst = 0;

// Apply data to rx\_data and simulate UART ready

rx\_ready = 1;

// Simulate receiving matrix\_a (3x3 matrix as 9 elements)

rx\_data = 8'd1; #10;

rx\_data = 8'd2; #10;

rx\_data = 8'd3; #10;

rx\_data = 8'd4; #10;

rx\_data = 8'd5; #10;

rx\_data = 8'd6; #10;

rx\_data = 8'd7; #10;

rx\_data = 8'd8; #10;

rx\_data = 8'd9; #10;

// Simulate receiving matrix\_b (3x3 matrix as 9 elements)

rx\_data = 8'd9; #10;

rx\_data = 8'd8; #10;

rx\_data = 8'd7; #10;

rx\_data = 8'd6; #10;

rx\_data = 8'd5; #10;

rx\_data = 8'd4; #10;

rx\_data = 8'd3; #10;

rx\_data = 8'd2; #10;

rx\_data = 8'd1; #10;

// Wait for matrix multiplication to finish

wait(uut.multiplication\_done == 1); // Wait for multiplication to be done

$display("Matrix multiplication complete, displaying results.");

$display("Matrix A:");

$display("A[0] = %d, A[1] = %d, A[2] = %d", uut.matrix\_a[0], uut.matrix\_a[1], uut.matrix\_a[2]);

$display("A[3] = %d, A[4] = %d, A[5] = %d", uut.matrix\_a[3], uut.matrix\_a[4], uut.matrix\_a[5]);

$display("A[6] = %d, A[7] = %d, A[8] = %d", uut.matrix\_a[6], uut.matrix\_a[7], uut.matrix\_a[8]);

$display("Matrix B:");

$display("B[0] = %d, B[1] = %d, B[2] = %d", uut.matrix\_b[0], uut.matrix\_b[1], uut.matrix\_b[2]);

$display("B[3] = %d, B[4] = %d, B[5] = %d", uut.matrix\_b[3], uut.matrix\_b[4], uut.matrix\_b[5]);

$display("B[6] = %d, B[7] = %d, B[8] = %d", uut.matrix\_b[6], uut.matrix\_b[7], uut.matrix\_b[8]);

// Display the final matrix multiplication result (matrix\_c)

$display("Matrix C (Multiplication Result):");

$display("C[0] = %d, C[1] = %d, C[2] = %d", uut.matrix\_c[0], uut.matrix\_c[1], uut.matrix\_c[2]);

$display("C[3] = %d, C[4] = %d, C[5] = %d", uut.matrix\_c[3], uut.matrix\_c[4], uut.matrix\_c[5]);

$display("C[6] = %d, C[7] = %d, C[8] = %d", uut.matrix\_c[6], uut.matrix\_c[7], uut.matrix\_c[8]);

// Finish test

$finish;

end

// Monitor the outputs

initial begin

$monitor("At time %t, tx\_data = %b, tx\_status = %b, debug\_data = %d", $time, tx\_data, tx\_status, debug\_data);

end

endmodule

**Output:**

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated

A white background with black numbers and a white background

Description automatically generated with medium confidence

