SUSTECH-CS207-PROJECT: FPGA-Based Matrix Calculator

I. Project Overview

The project aims to design and implement a matrix calculator circuit based on FPGA. The circuit should support functions such as matrix generation, display, and matrix operations (with a default maximum matrix dimension of 5×5). Data interaction with users and the system is achieved through basic input/output devices on the development board and UART serial communication.

II.Core Functions

After system startup, the main menu is displayed for users to select specific mode: 1. Matrix input and storage, 2. Matrix generation and storage, 3. Matrix display, 4. Matrix operations. At the end of each mode, users can choose to remain in the current mode or return to the main menu.

Under the "Matrix Operations" function, users can select specific operation types. After one operation is completed, users can choose to continue with the current operation type, switch to another operation type, or return to the main menu.

The system uses the switches, button on the development board, and UART serial communication software for input, while the LEDs, 7-

segment tubes on the development board, and UART serial communication software are used for output.

Note: Unless otherwise specified, matrices in this system, such as matrix A with dimensions m×n, have m and n as integers between 1 and 5, and matrix element values are integers between 0 and 9.

2.1 Input and Output Processing

2.1.1 Input

1. Function selection is done using the development board's switches and buttons: For example, switches set the system's working mode, and buttons serve as the "confirmation key."
2. Users input matrix-related information (e.g., matrix dimensions, matrix elements) in the UART transmission software and click the "Send" button to transmit data to the system, as shown in the red box in the figure below.

Note: If the UART function is not implemented, using the keyboard for equivalent input will not result in point deduction.

2.1.2 Output

1. The development board's LED lights and 7-segment tubes: Perform validity checks on the user-input matrix and selected operands. If an error is detected, an LED light is illuminated; if there

is no error, the LED light is turned off. Countdown information is synchronously displayed on the 7-segment tubes.

2. The UART transmission software's receiving window: Displays the operands and operation results of matrix operations via UART output, as shown in the red box in the figure below.

Note: If the UART function is not implemented, using VGA for equivalent output in the same format will not result in point deduction.

2.2 Matrix Input, Generation, and Storage

Note: The system supports storing a maximum of x matrices of each specification (x is configurable, default is 2). The system supports matrix A (m×n), where the values of m and n are configurable, defaulting to integers between 1 and 5.

2.2.1 Support for User Input and Matrix Storage

1. User input: When inputting a matrix, users should first input the matrix dimensions, then input the value of each matrix

element in row-major order. For matrix A with dimensions m×n, input m first, then n, and finally Aij (i from 0 to m-1, with j from 0 to n-1 for each fixed i). For example, when a user inputs a 2×3 matrix, the data in the sending window is as shown in the table below.

Input data in the sending window	Generated matrix
2 3 4 5 6 7 8 9	4 5 6 7 8 9

2. System detection

(1) Dimension range: If the user-input dimensions exceed the limit (1–5), the error detection LED light is illuminated, and the user must re-enter the data.

(2) Matrix element value range: If the user-input array elements exceed the 0–9 range, the error detection LED light is illuminated, and the user must re-enter the data.

(3) Number of matrix elements:

- If the user inputs fewer array elements than required (e.g., matrix size is 2×3 but the user only inputs 4 data points), these 4 data points are sequentially assigned as matrix elements A00, A01, A02, A10, and the remaining elements(A11, A12) are assigned a value of 0.

- If the user inputs more array elements than specified (e.g., matrix size is 2×3 but the user inputs 8 matrix elements), only the first 6 matrix elements are used, and the 7th and 8th input matrix elements are ignored.

2.2.2 Support for Generating and Storing Matrices Based on User Specifications

When inputting a matrix, users should first input the matrix dimensions (for matrix A with dimensions m×n, input m first, then n), the number of matrices (no more than 2), and then generate the value of each matrix element in row-major order.

For example, if the user inputs 1 3 2 (as shown in the left cell of the table below), it indicates the need to generate 2 matrices A and B of size 1×3. The circuit then generates 6 random numbers: 8 2 6 5 7 9, and matrices A and B are as shown in the middle and right cell of the table below.

Note: Numbers generated in ascending or descending order are not considered random and will not earn points.

Data user input	Matrix A	Matrix B
1 3 2	8 2 6	5 7 9

2.2.3 System Matrix Storage

The system can store user-input matrices, system-generated matrices based on user specifications, and matrix numbers (the numbering method is designed and implemented by the designer and is not specified by the user). A maximum of x matrices of each specification can be stored (x defaults to 2). If x matrices of the same specification are already stored and a new matrix of the same specification is input, the new matrix overwrites the existing ones. It should be ensured that all originally stored matrices can be overwritten by newly input matrices of the same specification. For example, if matrices A and B are stored, inputting C overwrites A, and inputting D overwrites B.

2.3 Matrix Display

The matrix needs to be displayed in a clear manner, such as in the following figure, where elements are separated by spaces or line breaks. Other display methods can also be used to ensure clarity and user experience; Display multiple matrices in sequence from top to bottom at once.

For example: Output a 2×3 matrix A { {4, 5, 6}, {7, 8, 9} }, the receiving window of the UART communication assistant displays as follows:

```
4 5 6
7 8 9
```

2.4 Matrix Operations

The matrix operation types supported by the system are as follows: Matrix transpose, matrix addition, matrix multiplication, and convolution (convolution is a bonus; the others are required).

During matrix operations, users first select the operation type, and the system display the operation type on the 7-segment tubs of developing board. The system then displays the stored matrix information in the receiving window of the UART software. Users then select the operands, which are displayed in the receiving window of the UART software. The calculation is performed, and the operation result is displayed in the receiving window of the UART software. After each operation, users can use switches to return to the matrix operation mode to start the next operation or return to the matrix input, generation, and storage mode to store new operands.

2.4.1 Selecting the Operation Type

Use toggle switches to select the specific operation type (matrix transpose, matrix addition, scalar multiplication, matrix multiplication, convolution). After setting the toggle switches, press the "confirmation key" (select a suitable button on the development board) to confirm the selection.

After the user makes the selection, the 7-segment tubes show the operation type (Matrix transpose: T, Matrix addition: A, Scalar multiplication: B, Matrix multiplication: C, Convolution: J).

2.4.2 Displaying All Stored Matrix Information

The receiving window of the UART communication software displays the total number of matrices and the matrix specification list (m×n×x, where m and n are the two dimensions of matrix A (m×n), and x is the number of matrices of this specification). If the number of matrices of a certain specification is 0, that matrix specification is not displayed. For example, if the system stores 3 matrices, including 1 matrix of size 2×2 and 2 matrices of size 4×5, the display information is as shown in the table below (output three sets of information, separated by spaces).

3   2*2*1   4*5*2

2.4.3 Selecting Operands

1. **Manual operand selection by the user:**

1) Manual selection of operands from the system-stored matrices:

a. The user specifies the matrix dimensions: The user inputs the operand dimensions m and n sequentially (press the "confirmation key" after each input).

b. Display all matrices of that dimension: After the dimensions are input, the system displays all matrices of that dimension and their numbers in the receiving window of the UART communication assistant (the table below demonstrates the a, b interaction).

Data user input	Information displayed in the receiving window of Uart Communication Assistant
2 3	1 1 2 3 4 5 6 2 1 1 1 7 8 9

c. The user selects the matrix by inputting the corresponding number and presses the "confirmation key" to confirm the operand selection.

d. The receiving window of the UART communication assistant displays the operand (the table below demonstrates the c, d interaction).

Data user input	Information displayed in the receiving window of Uart Communication Assistant
2	1 1 1 7 8 9

e. Three selection modes: The system determines the operand input method based on the operation type:

a) If only one matrix is needed as an operand (e.g., transpose), the user only needs to select the operand once.

b) If two matrices are needed as operands (e.g., addition, multiplication), the above a–d operations are performed twice.

c) If one matrix and one constant are needed (e.g., scalar multiplication), first select the matrix (a–d operations once) and then input the scalar. The scalar value is input using the development board's switches, and the "confirmation key" is pressed to confirm the scalar input.

2) **System validation of operand compliance with operation requirements**

Matrix addition and matrix multiplication have requirements for the two operands (e.g., matrix addition requires the two operands to have the same dimensions; matrix multiplication A(m×n) and B(n×p) requires n and m to be the same, etc.). If the operands do not meet the requirements, the system returns to the operand selection phase for re-selection. Based on the validation result of operand legality:

a. If the operands meet the calculation requirements, the system enters the calculation mode.

b. If the operands do not meet the calculation requirements, the error LED is illuminated, and an input countdown is initiated (the countdown is synchronously displayed on the 7-segment tubes):

a) The countdown time is in seconds, dynamically configurable (i.e., the user can specify the countdown during system operation without regenerating the bit file). The default countdown value is 10 seconds, and the configurable time range is 5–15 seconds.

b) If new operands are specified within the countdown period (all operand numbers are input and the "confirmation key" is pressed), the new operands are checked:

   i. If meet the requirements, the system enters the calculation mode.

   ii. If do not meet the requirements, the system follows step b.

c) If the input is not completed by the time the countdown expires, the current input is invalid, and the user must re-enter

2. **System random selection of operands based on operation requirements:**

The system automatically selects operands that meet the requirements from the stored matrices. For scalar multiplication (x × A (matrix dimensions m×n)), x is a random number between 0 and 9. After the system completes the selection, the operands are displayed sequentially on the output device. For example, for A × B, A is displayed first, followed by B.

2.4.4 Matrix Calculation

1. **Matrix transpose**: Implements the transpose operation of stored matrices. For example: Transposing matrix A (1×3) = {1, 2, 3}, the transposed output B should be displayed in the receiving window of the UART communication software as:

```
1
2
3
```

2. **Matrix addition**: Implements the addition of two matrices of the same dimension. For example, adding matrix A (2×3) = { {1, 2, 3}, {3, 4, 5} } and matrix B (2×3) = { {3, 3, 3}, {2, 2, 2} }, A + B = C, the operation result C (2×3) = { {4, 5, 6}, {5, 6, 7} }, should be displayed in the receiving window of the UART communication software as:

```
4    5    6
5    6    7
```

3. **Scalar multiplication**: Implements the multiplication of a matrix by a scalar. For example, performing scalar multiplication on matrix A (2×3) = { {1, 2, 3}, {3, 4, 5} } with a multiplier of 3, A × 3 = C, the operation result C (2×3) = { {3, 6, 9}, {9, 12, 15} }, should be displayed in the receiving window of the UART communication software as:

3	6	9
9	12	15

4. **Matrix multiplication**: For matrix A with dimensions m×n and matrix B with dimensions n×p, the resulting matrix C has dimensions m×p, where the element C[i][j] = Σ(A[i][k] × B[k][j]), k from 1 to n (m, n, p range from 1 to 5). For example, multiplying matrix A (2×3) = { {1, 2, 3}, {3, 4, 5} } and matrix B (3×2) = { {1, 0}, {2, 1}, {3, 2} }, A × B = C, the operation result C (2×2) = { {14, 8}, {26, 14} }, should be displayed in the receiving window of the UART communication software as:

14	8
26	14

III. Additional Functions

3.1 Display matrix output with column elements aligned to the left

For example, in the receiving window display of Uart communication software (the following table is only a left aligned example)

1	12	3	-5	8
251	9	2045	26	-4

3.2 Configuration Function

System matrix-related parameters are dynamically configurable (i.e., the user can specify the parameters during system operation without regenerating the bit file for it to take effect):

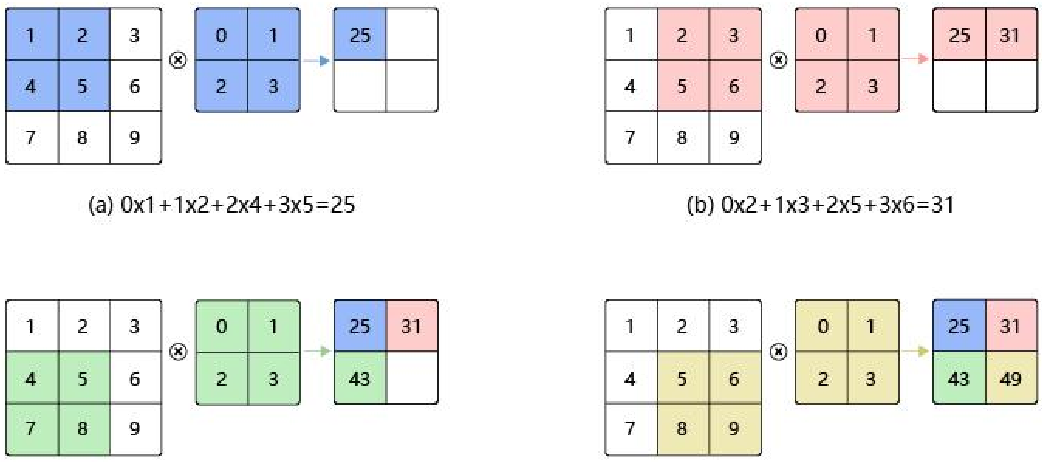
These parameters are synchronized and effective in matrix input, generation, detection, and storage functions.

(1) Maximum number x of matrices of each specification: x defaults to 2. If set to 5, it means a maximum of 5 matrices of each specification can be generated.

(2) Matrix element value range: The default matrix element value range is 0–9. If set to -3, 20, it means integers in the range [-3, 20] are legal matrix elements.

3.3 Convolution Function

The specific calculation process of convolution is illustrated in the figure below. The left image represents the input image, which is a two-dimensional array of dimensions m × n; the middle image represents the convolution kernel, which is a two-dimensional array of dimensions a × b. By sliding the convolution kernel (assuming a stride of 1) and performing matrix multiplication with the corresponding submatrix of the input, the final output matrix is obtained. The example in the figure below requires 4 matrix multiplications, and the output matrix is 2×2.



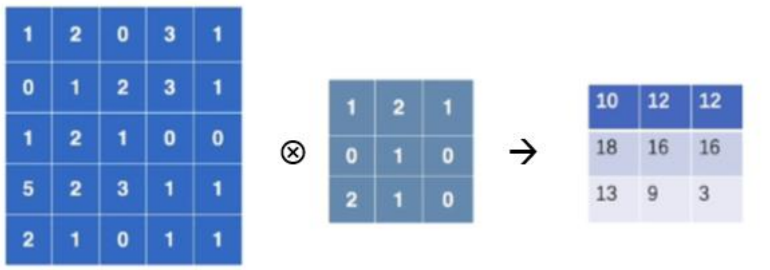
(a) 0x1+1x2+2x4+3x5=25

(b) 0x2+1x3+2x5+3x6=31

(c) 0x4+1x5+2x7+3x8=43

(d) 0x5+1x6+2x8+3x9=49

In the following example, the input image is a 5×5 matrix, the convolution kernel is a 3×3 matrix, and the convolution result, as shown in figure c), is a 3×3 matrix. This process requires 9 matrix multiplications.



a) input (5x5)   b) kernel (3x3)   c) output (3x3)

Bonus implementation requirements:

Implement convolution with a stride of 1. The input image is a given 10×12 matrix (element values 0–9, as shown below). Implement using the hard-coded method in Verilog.

Input Image (10 rows × 12 cols): 3 7 2 9 0 5 1 8 4 6 3 2 8 1 6 4 7 3 9 0 5 2 8 1

```
4 9 0 2 6 8 3 5 7 1 4 9
7 3 8 5 1 4 9 2 0 6 7 3
2 6 4 0 8 7 5 3 1 9 2 4
9 0 7 3 5 2 8 6 4 1 9 0
5 8 1 6 4 9 2 7 3 0 5 8
1 4 9 2 7 0 6 8 5 3 1 4
6 2 5 8 3 1 7 4 9 0 6 2
0 7 3 9 5 6 4 1 8 2 0 7
```

```
/ * input image Implemented using hard code, the following code is for reference only (the code is synthesizable and has been verified on board) * /
module input_image_rom (
    input clk,
    input [3:0] x,          //Row address
    input [3:0] y,           // Column address
    output reg [3:0] data_out // output
);

// ROM  Memory definition -10 rows x 12 columns=120 elements
reg [3:0] rom [0:119];

// Initialize ROM
initial begin
    // Line 1: 3 7 2 9 0 5 1 8 4 6 3 2
    rom[0] = 4'd3;  rom[1] = 4'd7;  rom[2] = 4'd2;  rom[3] = 4'd9;
    rom[4] = 4'd0;  rom[5] = 4'd5;  rom[6] = 4'd1;  rom[7] = 4'd8;
    rom[8] = 4'd4;  rom[9] = 4'd6;  rom[10] = 4'd3; rom[11] = 4'd2;

    // Line 2: 8 1 6 4 7 3 9 0 5 2 8 1
    rom[12] = 4'd8; rom[13] = 4'd1; rom[14] = 4'd6; rom[15] = 4'd4;
    rom[16] = 4'd7; rom[17] = 4'd3; rom[18] = 4'd9; rom[19] = 4'd0;
    rom[20] = 4'd5; rom[21] = 4'd2; rom[22] = 4'd8; rom[23] = 4'd1;

    // Line 3: 4 9 0 2 6 8 3 5 7 1 4 9
    rom[24] = 4'd4; rom[25] = 4'd9; rom[26] = 4'd0; rom[27] = 4'd2;
    rom[28] = 4'd6; rom[29] = 4'd8; rom[30] = 4'd3; rom[31] = 4'd5;
    rom[32] = 4'd7; rom[33] = 4'd1; rom[34] = 4'd4; rom[35] = 4'd9;

    // Line 4: 7 3 8 5 1 4 9 2 0 6 7 3
    rom[36] = 4'd7; rom[37] = 4'd3; rom[38] = 4'd8; rom[39] = 4'd5;
    rom[40] = 4'd1; rom[41] = 4'd4; rom[42] = 4'd9; rom[43] = 4'd2;
    rom[44] = 4'd0; rom[45] = 4'd6; rom[46] = 4'd7; rom[47] = 4'd3;

    // Line 5: 2 6 4 0 8 7 5 3 1 9 2 4
    rom[48] = 4'd2; rom[49] = 4'd6; rom[50] = 4'd4; rom[51] = 4'd0;
    rom[52] = 4'd8; rom[53] = 4'd7; rom[54] = 4'd5; rom[55] = 4'd3;
    rom[56] = 4'd1; rom[57] = 4'd9; rom[58] = 4'd2; rom[59] = 4'd4;

    // Line 6: 9 0 7 3 5 2 8 6 4 1 9 0
    rom[60] = 4'd9; rom[61] = 4'd0; rom[62] = 4'd7; rom[63] = 4'd3;
```

```
	rom[64] = 4'd5; rom[65] = 4'd2; rom[66] = 4'd8; rom[67] = 4'd6;
	rom[68] = 4'd4; rom[69] = 4'd1; rom[70] = 4'd9; rom[71] = 4'd0;

	// Line 7: 5 8 1 6 4 9 2 7 3 0 5 8
	rom[72] = 4'd5; rom[73] = 4'd8; rom[74] = 4'd1; rom[75] = 4'd6;
	rom[76] = 4'd4; rom[77] = 4'd9; rom[78] = 4'd2; rom[79] = 4'd7;
	rom[80] = 4'd3; rom[81] = 4'd0; rom[82] = 4'd5; rom[83] = 4'd8;

	// Line 8: 1 4 9 2 7 0 6 8 5 3 1 4
	rom[84] = 4'd1; rom[85] = 4'd4; rom[86] = 4'd9; rom[87] = 4'd2;
	rom[88] = 4'd7; rom[89] = 4'd0; rom[90] = 4'd6; rom[91] = 4'd8;
	rom[92] = 4'd5; rom[93] = 4'd3; rom[94] = 4'd1; rom[95] = 4'd4;

	// Line 9: 6 2 5 8 3 1 7 4 9 0 6 2
	rom[96] = 4'd6;    rom[97] = 4'd2;    rom[98] = 4'd5;    rom[99] = 4'd8;
	rom[100] = 4'd3; rom[101] = 4'd1; rom[102] = 4'd7; rom[103] = 4'd4;
	rom[104] = 4'd9; rom[105] = 4'd0; rom[106] = 4'd6; rom[107] = 4'd2;

	// Line 10: 0 7 3 9 5 6 4 1 8 2 0 7
	rom[108] = 4'd0; rom[109] = 4'd7; rom[110] = 4'd3; rom[111] = 4'd9;
	rom[112] = 4'd5; rom[113] = 4'd6; rom[114] = 4'd4; rom[115] = 4'd1;
	rom[116] = 4'd8; rom[117] = 4'd2; rom[118] = 4'd0; rom[119] = 4'd7;
end

// Synchronized reading
wire [6:0] addr;

assign addr = x*12+y;

always @(posedge clk) begin
	data_out <= rom[addr];
end

endmodule
```

The convolution kernel is a 3×3 matrix input by the user on-site.

On-site testing:

1. After inputting the convolution kernel, start the calculation immediately and count the number of clock cycles.
2. After the calculation is completed:

a) The serial port displays the output matrix.

b) The 7 segment tubes show the total number of clock cycles.

3. Output matrix dimensions: 8×10.

PPT presentation video: Provide a PPT with no more than 10 pages, rich in graphics and text (prohibiting large paragraphs of text and code pasting). Use Tencent Meeting to record a screen-based explanation of the PPT and upload it to BB.

The PPT and explanation must include the following:

1. Simplified block diagram of the system architecture, the position of the convolution module and I/O ports (cannot use Vivado synthesize schematic or RTL analysis schematic screenshots).
2. Design schematic of the convolution module (cannot use Vivado synthesize schematic or RTL analysis schematic screenshots).
3. Data flow control solution.
4. Performance optimization strategies (if any).
5. Analysis of the optimization effects achieved with LLM assistance (if any).
6. Theoretical cycle count analysis.
7. Comparison of actual test cycle counts and the actual operating clock frequency.

8. Resource usage report of the whole project and the convolution module.

9. Problems encountered and solutions.

TIPS: Method for viewing the resource usage report of the current design under Vivado Confirm the circuit module to be evaluated in the Vivado project, set it as the top-level module (right-click on the module name and select set as top), implement the circuit module (select “Run Implement” under IMPLEMATION), and after implementation, select “Report Ultilization” under IMPLEMATION (as shown in the bottom left figure) Resource statistics can be conducted immediately. Select “Summary” on the left side of the Ultilization window to view the resource usage report (as shown in the bottom right image).