SUSTECH-CS207-PROJECT:基于 FPGA 的矩阵计算器

一、项目概述

本项目旨在设计并实现一个基于 FPGA 的矩阵计算器电路。该电路需支持基于矩阵的生成、展示、矩阵运算等功能（矩阵维度默认上限为 5×5），通过开发板上的基础输入输出设备及 UART 串口实现与用户及系统的数据交互。

二、核心功能

系统开机后进入主菜单供用户选择具体功能：1. 矩阵输入及存储，2. 矩阵生成及存储，3. 矩阵展示，4. 矩阵运算。每种模式结束后可以选择继续停留在该模式下，也可以选择返回到主菜单。

“矩阵运算”功能下可以选择具体的运算类型，一种运算结束后可以选择继续当前运算类型，或者切换到其他运算类型，或者切换到主菜单。

系统采用开发板的拨码开关、按键开关以及 Uart 串口通信软件做输入，采用开发板的 LED、7 段数码显示管以及 Uart 串口通信软件按做输出。

说明：本系统中涉及到的矩阵，如未做特别说明，以维度为 m×n 的矩阵 A 为例，m 和 n 为 1~5 之间的整数；矩阵元素的值为 0~9 之间的整数。

2.1 输入输出处理

2.1.1 输入

1. 功能选择用开发板的拨码开关和按键: 比如拨码开关上设置系统的工作模式，按键用作”确认键 “。

2. 用户在 uart 传输软件上输入矩阵相关信息，比如矩阵维度、矩阵元素，点击软件上的“发送” 将数据送給系统。如下图红色方框所示。

说明：如 uart 功能未实现，使用键盘实现相同方式的输入则视为等价，不扣分。

2.1.2 输出

1. 开发板的 LED 灯及数码管：对用户输入的矩阵、用户选择的运算数进行合法性检测，发现错误点亮一盏 LED 灯，如果没有错误则熄灭该 LED 灯，倒计时信息同步显示在 7 段数码显示管。

2. Uart 传输软件的接收窗口：矩阵运算的运算数、运算结果，都通过 uart 输出到 uart 传输软件，在该软件上进行展示，如下图红色方框所示。

说明：如 uart 功能未实现，使用 VGA 实现相同格式的输出则视为等价，不扣分。

2.2 矩阵输入、生成及存储

说明：系统支持每种规格的矩阵最多存储 x 个（x 可配置，默认为 2）。系统支持的矩阵 A（m*n），m 和 n 的值可配置，默认为 1~5 之间的整数。

2.2.1 支持用户输入矩阵并存储矩阵

1. 用户输入：用户在输入矩阵时应先输入矩阵的维度，再按先行后列的方式遍历输入每个矩阵元素的值。以维度为 m×n 的矩阵 A 为例，应先输入 m，再输入 n，最后输入 Aij (i 从 0 到 m-1，i 值确定时 j 的值从 0 到 n-1). 例如，用户输入一个 2*3 的矩阵，发送窗口种的数据以及对应矩阵如下表所示（无需在接收窗口显示该矩阵）。

发送窗口的输入数据	生成的矩阵
2 3 4 5 6 7 8 9	4 5 6 7 8 9

2. 系统检测：

(1) 维度范围：如果用户输入的维度超出限制（1~5 之间），则点亮检测报错 LED 灯，需要用户重新输入。

(2) 矩阵元素的数值范围：如果用户输入的数组元素超出 0~9 的范围，则点亮检测报错 LED 灯，需要用户重新输入。

(3) 矩阵元素的个数：如果用户输入的数组元素的数量不足，比如矩阵规模是 2*3，但用户只输入了 4 个数据，则这 4 个数据被依次当作矩阵元素 A00, A01,A02,A10, 将 A11, A12 赋值为 0； 如果用户输入的数组元素的个数超出指定范围，比如矩阵规模是 2*3，用户输入了 8 个矩阵元素，则只取前 6 个矩阵元素的值，第 7、8 个输入的矩阵元素不作数。

2.2.2 支持根据用户的指定生成矩阵并存储

用户在输入矩阵时应先输入矩阵的维度（以维度为 m×n 的矩阵 A 为例，先输入 m 再输入 n）、矩阵的个数（不超过 2 个），再按先行后列的方式遍历生成每个矩阵元素的值。

例如，用户输入 1 3 2(如下表左侧所示)，表示需要产生 2 个 1*3 的矩阵 A 和 B. 电路因此生成 6 个随机数 8 2 6 5 7 9，则 A、B 如下表右侧所示。

请注意：使用递增或递减的方式产生的数字不是随机数，不得分。

用户输入	矩阵 A	矩阵 B
1 3 2	8 2 6	5 7 9

2.2.3 系统存储矩阵

系统可以存储用户输入的矩阵、系统根据用户指定生成的矩阵以及矩阵的编号（编号方式由设计者自行设计并在系统里实现，不由用户输入指定）。每种规格的矩阵最多存储 x(x 的值默认为 2)个，如果同种类型的矩阵已存储了 x 个，再输入相同规格的矩阵，则新输入的矩阵覆盖原来的矩阵。应保证原存储的矩阵可能被新输入的相同规格的矩阵全部覆盖，比如原来有 A,B，新输入 C 时覆盖了 A，再新输入 D 时覆盖 B。

2.3 矩阵展示

需以清晰方式展示矩阵，例如下图中，元素之间以空格、换行间隔，也可采用其他展示方式，需清晰并保证用户体验；一次展示多个矩阵时从上到下依次展示。

比如：输出一个 2*3 的矩阵 A{ {4，5，6}， {7，8，9} }，Uart 通信助手的接收窗口显示如下所示。

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

2.4 矩阵运算

系统支持的矩阵运算类型如下：矩阵转置、矩阵加法、标量和矩阵相乘、矩阵乘法、卷积，其中卷积运算为附加功能（bonus），其他 4 类运算为必选。

在进行矩阵运算时用户先选择运算类型，系统在 7 段数码管显示选中的运算类型，系统将存储的矩阵信息显示在 Uart 软件的接收窗口，用户再选择运算数，系统将运算数显示在 Uart 软件的接收窗口，并进行计算，将运算结果显示在 Uart 软件的接收窗口。每个运算结束后用户可以通过开关返回到矩阵运算模式开始下一次的运算，或者返回到矩阵输入、生成及存储模式，存入新的运算数。

2.4.1 选择运算类型

用拨码开关来选择具体的运算类型（矩阵转置、矩阵加法、标量乘法、矩阵乘法、卷积（具体实现将在 bonus 部分说明，不做卷积 bonus 则无需支持该运算类型））。拨码开关设置好后按“确认键”（开发板上选择一个合适的按键）表示选择完毕。

用户选择完毕后，7 段数码管上显示运算类型（矩阵转置 T、矩阵加法 A、标量乘法 B、矩阵乘法 C、卷积 J）

2.4.2 显示存储矩阵的全部信息

Uart 通信软件的接收窗口中显示矩阵总数、矩阵规格列表（m*n*x，其中 m，n 为矩阵 A（m*n）的两个维度，x 是这种规格矩阵的个数）。某规格的矩阵个数为 0，则不显示该矩阵的规格列表。比如系统存储了 3 个矩阵，其中 1 个是 2*2 的矩阵，2 个 4*5 的矩阵。则显示信息如下表所示（输出三组信息，每组信息之间以空格分隔）。

3　2*2*1　4*5*2

2.4.3 选择运算数

1. 用户手动选择运算数

1）用户从系统存储的矩阵中手动选择运算数

a. 用户指定矩阵维度：用户依次输入运算数的维度 m 和 n（每次输入一个维度后按“确认键”）

b. 展示该维度下的所有矩阵：维度输入完毕后，系统在 Uart 通信助手的接收窗口中显示该维度的所有矩阵及其编号。（下表示范 a,b 互动）

用户输入	Uart 通信助手的接收窗口显示
2 3	1 1 2 3 4 5 6 2 1 1 1 7 8 9

c. 用户选择矩阵输入对应编号，按“确认键”，表示选定该运算数。

d. Uart 通信助手的接收窗口中显示该运算数（下表示范 c,d 互动）。

用户输入	Uart 通信助手的接收窗口显示
2	1 1 1 7 8 9

e. 三种选择模式：系统根据运算类型来确定运算数的输入方式：

a) 如果只需要一个矩阵作为运算数（比如 转置、卷积（矩阵作为卷积核）），则用户只需要选择一次运算数；

b) 如需要两个矩阵作为运算数（比如加、乘积）则上述 a-d 操作要进行 2 次；

c) 如需要一个矩阵和一个常量（比如标量乘法）则先选择矩阵(a-d 操作 1 次)，再输入标量。标量值由开发板的拨码开关进行输入，按“确认键”表示标量输入完毕。

2）系统判定运算数是否符合运算要求

矩阵加法、矩阵乘法运算对两个运算数都有要求（比如矩阵加法要求两个运算数的维度相同，A(Am*An)和 B(Bm*Bn)矩阵乘法要求 An 与 Bm 相同等），如果运算数不符合要求，则应回到运算数的选择起始阶段重新选择运算数。根据运算数合法性的判定结果：

a. 如运算数符合计算要求则进入计算模式。

b. 如运算数不符合计算要求，则点亮报错 LED，并开启输入倒计时（倒计时在数码管上同步显示）；

a) 倒计时的时间以秒为单位，动态可配置（即系统运行过程中用户输入指定倒计时，不需要重新生成 bit 文件），倒计时的默认值为 10 秒，可配置时间为 5~15 秒。

b) 倒计时内完成新的运算数指定（所有运算数的编号都完成输入并按“确认键”）则对新的运算数进行检查。

i. 如果符合要求则进入计算模式

ii. 如果不符合要求则按 b 进行操作。

c) 倒计时到期没输入完则本次输入无效，需要再次回到运算数的选择起始阶段重新选择运算数。

2. 系统按照运算要求随机选择运算数

由系统自动从存储的矩阵中随机选择符合要求的运算数。涉及到标量乘法（x*A(矩阵维度 m*n)）时，x 取 0~9 之间的随机数。系统选择完毕后，将运算数依次显示到输出设备上，比如 A*B，先显示 A 再显示 B。

2.4.4 矩阵计算

1. **矩阵转置：**实现存储矩阵的转置运算

比如：对矩阵 A（1*3）={1，2，3}进行转置，转置后的输出 B 在 Uart 通信软件的接收窗口显示应为：

```
1
2
3
```

2. **矩阵加法：**实现两个同维度矩阵的加法运算。

比如对矩阵 A(2*3)={ {1,2,3}, {3,4,5} } 和矩阵 B(2*3)={ {3,3,3}, {2,2,2} }进行加法运算，A+B=C，运算结果 C (2*3)={ {4,5,6}, {5,6,7} }，在 Uart 通信软件的接收窗口显示应为：

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

3. **矩阵标量乘法：**实现矩阵与一个标量的乘法运算

比如对矩阵 A(2*3)={ {1,2,3}, {3,4,5} } 做标量乘法，乘数为 3，A*3=C，运算结果 C(2*3)={ {3,6,9}, {9,12,15} }， 在 Uart 通信软件的接收窗口显示应为：

```
3  6  9
9  12 15
```

4. **矩阵乘法：**按照矩阵 A 维度为 m×n，矩阵 B 维度为 n×p，则结果矩阵 C 维度为 m×p，其中元素 C[i][j] = Σ(A[i][k] * B[k][j]), k 从 1 到 n。（m,n,p 的范围都是 1~5）

比如对矩阵 A(2*3)={ {1,2,3}, {3,4,5} } 和矩阵 B(3*2)={ {1,0}, {2,1}，{3,2} }进行矩阵乘法，A*B=C，运算结果 C(2*2)={ {14,8}, {26,14} }， 在 Uart 通信软件的接收窗口显示应为：

```
14 8
26 14
```

三、 附加功能

3.1 输出矩阵时按列元素靠左对齐展示

比如在 Uart 通信软件的接收窗口显示（下表仅做左对齐示例）：

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

3.2 设置功能

系统矩阵的相关参数动态可配置（即系统运行过程中用户输入指定参数，不需要重新生成 bit 文件即可生效）：

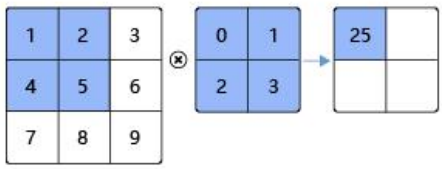
这些参数在矩阵输入、生成、检测和存储功能中都同步有效。

(1) 每种规格的矩阵最大个数 x，x 的值默认为 2，如果设置为 5，表示每种规格的矩阵最多可生成 5 个。

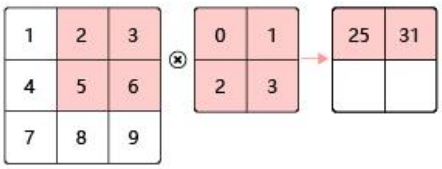
(2) 矩阵元素的数值范围：默认矩阵元素的数值范围是 0~9，如果设置成-3，20，则表示可以生成[-3,20] 之间的整数都是合法矩阵元素。

3.3 卷积功能

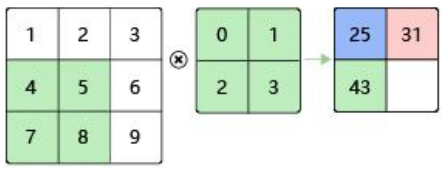
卷积的具体计算过程可由下图所示，每张图的左图表示输入图像 input image，是一个维度为 m x n 的二维数组；中间的图表示卷积核 kernel，是一个维度为 a x b 的二维数组，通过滑动卷积核（假设步长 stride 为 1），与对应位置的 input 子矩阵进行矩阵乘法，可得到最终的输出矩阵 output，下图所示例子需做 4 次矩阵乘法，输出矩阵为 2x2。



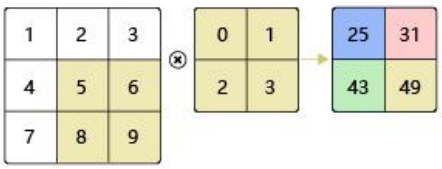
(a) 0x1+1x2+2x4+3x5=25



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

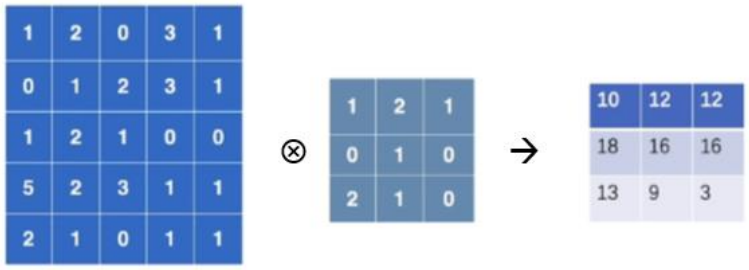


(c) 0x4+1x5+2x7+3x8=43



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

在接下来的例子中，输入图像为 5x5 矩阵，卷积核为 3x3 矩阵，卷积结果如图 c)所示为 3x3 矩阵，此过程需做 9 次矩阵乘法。



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

Bonus 实现要求:

实现步长为 1 的卷积运算。Input image 为给定 10x12 矩阵（元素值 0-9，如下所示）。使用 hard code 方式在 verilog 中实现。

```
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  使用 hard code 方式实现，以下代码供参考（已综合并上板验证过）
module input_image_rom (
    input clk,
    input [3:0] x,          //  行地址
    input [3:0] y,          //  列地址
    output reg [3:0] data_out //  输出
);

// ROM 存储器定义  - 10 行×12 列  = 120 个元素
reg [3:0] rom [0:119];

//  初始化 ROM 内容
initial begin
    //  第 1 行: 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;

    //  第 2 行: 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;

    //  第 3 行: 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;

        // 第 4 行: 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;

        // 第 5 行: 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;

        // 第 6 行: 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;

        // 第 7 行: 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;

        // 第 8 行: 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;

        // 第 9 行: 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;

        // 第 10 行: 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

    // 同步读取
    wire [6:0] addr;

    assign addr = x*12+y;

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

    endmodule
```

卷积核为用户现场输入的 3x3 矩阵。

现场测试：

1. 输入卷积核后立即开始计算并统计时钟周期数
2. 计算完成后：

a) 串口显示输出矩阵
b) 数码管显示总时钟周期数

3. 输出矩阵尺寸：8x10

PPT 讲解视频：提供 10 页以内图文并茂 ppt（禁大段文字和代码粘贴），使用腾讯会议以录屏方式制作 ppt 讲解上传至 BB。

PPT 及讲解须包含以下内容：

1. 系统架构简易框图以及卷积模块所在的位置和接口示意图（不可使用 vivado synthesize schematic 或 RTL analysys schematic 截图）
2. 卷积模块设计示意图（不可使用 vivado synthesize schematic 或 RTL analysys schematic 截图）
3. 数据流控制方案
4. 性能优化策略（如有）
5. LLM 辅助优化效果分析（如有）
6. 理论周期数分析
7. 实际测试周期数对比以及实际运行时钟频率
8. 工程以及卷积子模块资源使用报告
9. 遇到的问题与解决方案

附：vivado 下查看当前设计的资源使用报告的方法

Vivado 工程中确认要评估的电路模块，将其设置为顶层模块（模块名上单击右键，选择 set as top），对该电路模块进行实现（IMPLEMENTATION 下选择 Run Implement），实现完成后在 IMPLEMENTATION 下选择 Report Ultilization（如左下图所示）．即可进行资源统计。在 Ultilization 窗口左侧选择 Summary，即可查看资源使用报告（如右下图所示）。