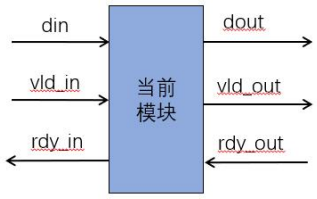
数字 1：基于vld-rdy握手协议的8x8矩阵运算模块

题目说明：本题目要求完成一个基于 vld-rdy 握手协议的 8x8 矩阵运算模块。

（1）vld-rdy 握手协议说明：



信号名	信号描述
din	前级送来的数据
vld_in	表示前级送来的数据是否有效，高电平有效
rdy_in	表示当前模块是否已经准备好接收数据，高电平有效
dout	当前模块输出的数据
vld_out	表示当前模块的输出数据是否有效，高电平有效
rdy_out	表示后级模块是否已经准备好接收数据，高电平有效

图 1.1 vld-rdy 握手端口定义

以最简单情况（假设模块中仅有一级寄存器打拍）进行说明，如图 1.2 所示：

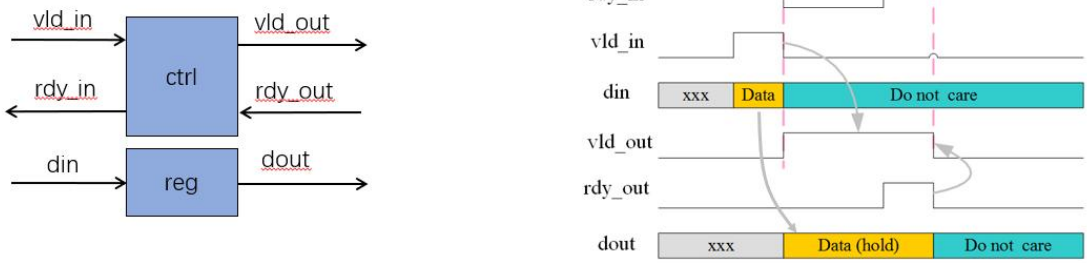


图 1.2 vld-rdy 握手时序说明

- 在 T0 时刻，当 reg 准备好接收（rdy_in==1），且前级数据有效（vld_in==1）时， din 被更新进入 reg，因此，T0 时刻以后 dout 有效（vld_out==1）
- 在 T1 时刻，后级可以接收，即（vld_out & rdy_out）==1 时，dout 才被取走，因此 T1 时刻后 vld_out 拉低
- 当 reg 中数据无效时，本级总是能够向前级接收数据
- 当 reg 中数据有效，且后级不能接收数据时，本级不能从前级接收数据，否则 reg 内当前数据被覆盖
- 当后级能够接收数据时，本级总是能够向前级接收数据
- 本级能否接收数据不依赖于前级数据是否有效

- 本级数据是否有效不依赖于后级能否接收数据

（2）矩阵运算要求：

某视频编码协议使用了图 1.3 所定义的矩阵运算$tmp = f(src)$，其中输入矩阵为 src[8][8]，其元素均为 8bit 有符号数，输出矩阵为 tmp[8][8]。

```
for (int y = 0; y < 8; ++y) {
    p[0] = src[y][0] + src[y][7];
    p[1] = src[y][3] - src[y][4];
    p[2] = src[y][1] + src[y][6];
    p[3] = src[y][2] - src[y][5];
    p[4] = src[y][2] + src[y][5];
    p[5] = src[y][1] - src[y][6];
    p[6] = src[y][3] + src[y][4];
    p[7] = src[y][0] - src[y][7];

    q[0] = p[0] + p[6];
    q[1] = p[1] - (p[7] >> 2);
    q[2] = p[2] + p[4];
    q[3] = p[3] + (p[5] >> 2);
    q[4] = p[2] - p[4];
    q[5] = (p[3] >> 2) - p[5];
    q[6] = p[0] - p[6];
    q[7] = (p[1] >> 2) + p[7];

    tmp[y][0] = q[0] + q[2];
    tmp[y][1] = q[3] - q[5] + q[7] + (q[7] >> 1);
    tmp[y][2] = (q[4] >> 1) + q[6];
    tmp[y][3] = -q[1] - q[3] - (q[3] >> 1) + q[7];
    tmp[y][4] = q[0] - q[2];
    tmp[y][5] = q[1] + q[5] + (q[5] >> 1) + q[7];
    tmp[y][6] = -q[4] + (q[6] >> 1);
    tmp[y][7] = -q[1] - (q[1] >> 1) + q[3] + q[5];
}
```

图 1.3 矩阵运算关系说明

（3）请按以上说明完成以下硬件设计：

- 输入为 vld-rdy 握手，每次握手成功输入两行，顺序为第 0-1 行、第 2-3 行、第 4-5 行、第 6-7 行；
- 输出为 vld-rdy 握手，每次握手成功输出**两列**，顺序为第 0-1 列、第 2-3 列、第 4-5 列、第 6-7 列；
- 中间变量 p、q、tmp 行及输出 tmp 列数据均为寄存器输出信号，且每级寄存器均为 vld-rdy 握手控制，后级可以通过拉低 rdy 信号反压前级
- **使用一组 8x8 数据缓存寄存器完成行列转换**

- 顶层端口说明：

端口名	方向	位宽	端口说明
clk	input	1	系统时钟，上升沿有效
rst_n	input	1	异步复位，低电平有效
src_row_vld	input	1	输入 src 数据有效标志，高有效
src_row_rdy	output	1	运算模块能够接收 src 数据的标志，高有效
src_row_data	input	8*8*2	输入 src 行数据，奇数行在高位，行内靠右数据在高位，以 01 行为例，以“_行索引_列索引”为后缀，排列顺序为： {src_row_data_1_7[7:0], src_row_data_1_6[7:0], ……, src_row_data_1_0[7:0], src_row_data_0_7[7:0], src_row_data_0_6[7:0], ……, src_row_data_0_0[7:0]}
tmp_col_vld	output	1	输出 tmp 数据有效标志，高有效
tmp_col_rdy	input	1	后级模块能够接收 tmp 数据的标志，高有效
tmp_col_data	output	12*8*2	输出 tmp **列数据**，奇数列在高位，列内靠下数据在高位，以 01 列为例，以“_列索引_行索引”为后缀，排列顺序为： {tmp_col_data_1_7[11:0], tmp_col_data_1_6[11:0], … … … … …, tmp_col_data_1_0[11:0], tmp_col_data_0_7[11:0], tmp_col_data_0_6[11:0], … … … … …, tmp_col_data_0_0[11:0]}

- 当 src_row_vld 及 tmp_col_rdy 均为 1 时，处理吞吐量应为每周期输出两列
- **建议步骤：**（1）设计如图 1.2 左侧所示输入输出为 vld-rdy 的纯寄存器控制模块；（2）设计 src->p,p->q,q->tmp 运算所对应的纯组合逻辑模块，注意位宽和符号问题；（3）设计行进列出的转换 buffer，注意吞吐量问题；（4）连接纯组合逻辑运算模块、寄存器控制模块及转换 buffer 并测试，注意前后级输入的 vld、rdy 信号可以随时反压；

（4）测试环境说明：

将**设计顶层模块命名为 matrix_cal_top，并将 check 点引用到的信号按表 1 命名**后即可直接仿真。除在./sim/file.f 中添加必要.v 文件外，**请勿改动仿真环境**。请在./sim 目录下分别执行 make test1、make test2、make test3、make test4；

如图 1.4 所示，测试环境提供 4 个 check 点，分别是中间数据 p、q、tmp 行及最终输出 tmp 列数据。每个 check 点同时比对 16 个数据元素。Check 点输入信号如表 1 所示。

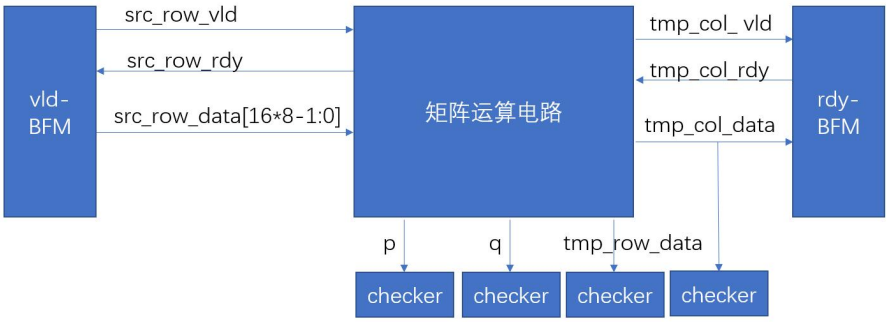


图 1.4 测试环境

表 1 check 点的信号引用

check 点	比对使能	比对数据	说明
p	u_matrix_cal_top.p_vld && u_matrix_cal_top.p_rdy	u_matrix_cal_top.p_data	包含 16 个数据，bit 排列方式与 src_row_data 一致
q	u_matrix_cal_top.q_vld && u_matrix_cal_top.q_rdy	u_matrix_cal_top.q_data	包含 16 个数据，bit 排列方式与 src_row_data 一致
tmp_row_data	u_matrix_cal_top.tmp_row_vld && u_matrix_cal_top.tmp_row_rdy	u_matrix_cal_top. tmp_row_data	包含 16 个数据，bit 排列方式与 src_row_data 一致

tmp_col_data	u_matrix_cal_top.tmp_col_vld && u_matrix_cal_top.tmp_col_rdy	u_matrix_cal_top. tmp_col_data	包含 16 个数据，bit 排列方式见端口列表

（5）评分标准

- p、q、tmp 行数据 check 通过各获 20 分，其中每个 run test 分值为 5 分（共 60 分）；
- 若使用一组 8x8 数据缓存寄存器完成 tmp 列数据比对获得 40 分，其中每种 run test 情况分值为 10 分；
- 若使用两组 8x8 数据缓存寄存器完成 tmp 列数据比对获得 10 分，其中每种 run test 情况分值为 2.5 分；