



2 x 2 卷积运算设计方案

Hollow Man

2*2 卷积运算设计方案

目录

版本信息	3
卷积过程	3
实现功能	3
设计思路	3
设计方案	4
版本 1.0	4
程序文件.....	4
仿真文件.....	6
仿真电路.....	7
运行结果.....	7
优点	8
缺点	8
版本 2.0	8
程序文件.....	8
仿真文件.....	11
仿真电路.....	12
运行结果.....	12
优点	13
版本 3.0	13
程序文件.....	14
运行结果.....	17
反思	18

版本信息

版本	完成日期	名称
1.0	2019/11/9	2*2 卷积运算
2.0	2019/11/16	
3.0	2019/11/23	

卷积过程

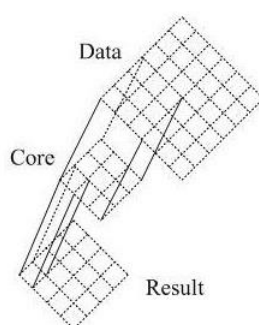


图 1 卷积过程示意图

图 1 为一般卷积过程的示意图，其过程为：

1. 对矩阵 A 补零，第一行之前和最后一行之后都补 $mb-1$ 行，第一列之前和最后一列之后都补 $nb-1$ 列
2. 将卷积核对准数据矩阵的左上角。然后，每个单元格内的数据对应相乘，然后将所有得到的数据相加，得到结果矩阵的对应单元格值。
3. 将卷积核向右移动一格单元格，重复步骤 2。
4. 若卷积核无法继续向右移动，此时再次将卷积核对准数据矩阵左端，并且将卷积核向下移动一格单元格，重复步骤 2、3，直到运算完成。

实现功能

2*2 卷积运算

设计思路

如图 2，我们选取了一个 2*2 数据矩阵和 2*2 的卷积核。首先对数据矩阵进行补零，然后再和卷积核进行卷积运算，得到结果后再将结果矩阵中的所有数据相加，得到结果。

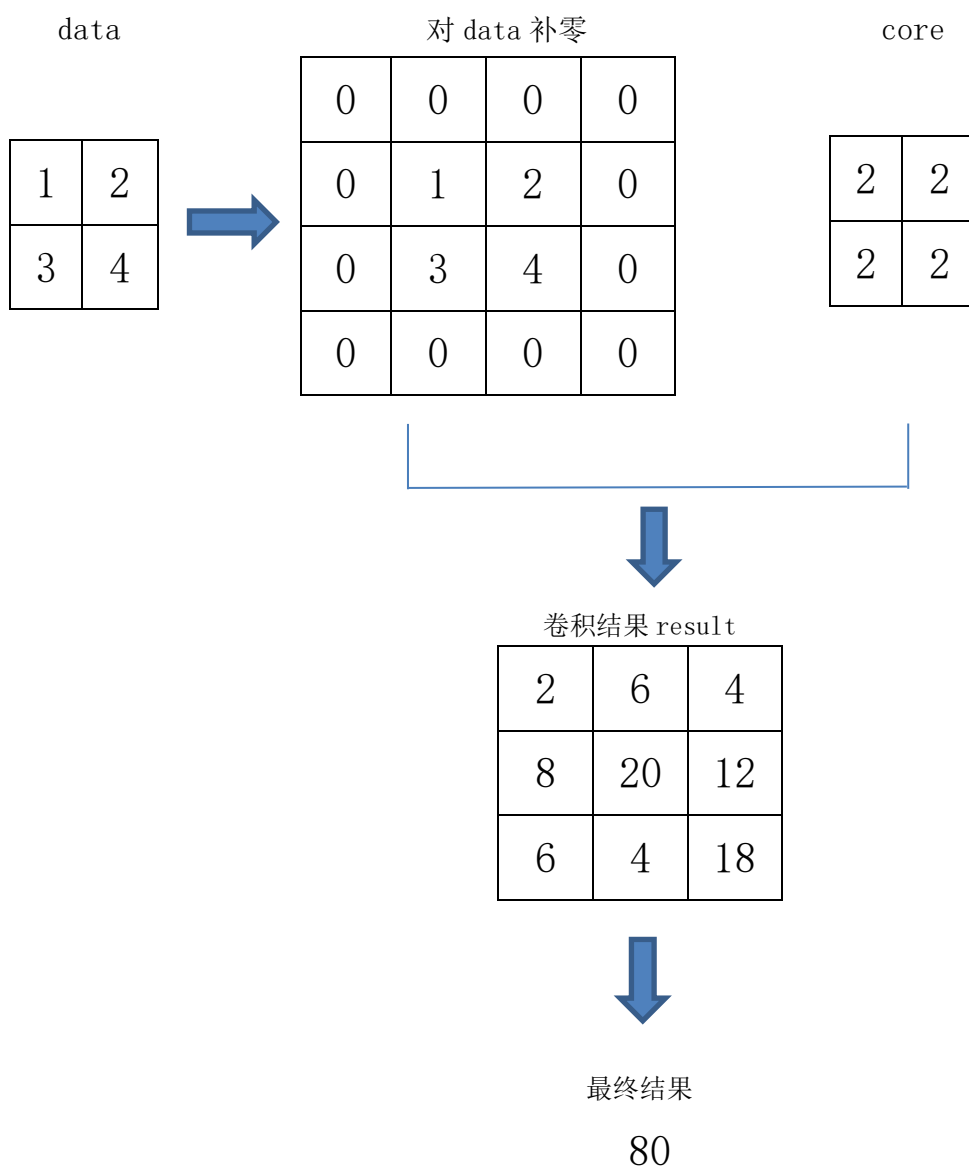


图 2 实现的 2*2 卷积运算

设计方案

版本 1.0 采用较为简单地加法器和乘法器组合实现卷积运算。

版本 2.0 采用按行循环求取卷积和的形式，电路图较为简洁。

版本 3.0 加入自己写的乘法器进行乘法运算。

下面是对各版本的详细介绍：

版本 1.0

程序文件

```
`timescale 1ns / 1ps
////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/11/08 15:24:15
```

```
// Design Name:
// Module Name: test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////

module test(
    a1,
    b1,
    c1,
    d1,
    a2,
    b2,
    c2,
    d2,
    sum
);
//输入分别赋值
    input [31:0] a1;
    input [31:0] b1;
    input [31:0] c1;
    input [31:0] d1;
    input [31:0] a2;
    input [31:0] b2;
    input [31:0] c2;
    input [31:0] d2;
    output sum;
    wire [31:0]a1;
    wire [31:0]b1;
    wire [31:0]c1;
    wire [31:0]d1;
    wire [31:0]a2;
    wire [31:0]b2;
    wire [31:0]c2;
    wire [31:0]d2;
```

```

reg [31:0] sum;

always @ (a1 or b1 or c1 or d1 or a2 or b2 or c2 or d2)
begin
    sum= a1*a2+a1*b2+a1*c2+a1*d2+b1*a2+b1*b2+b1*c2+b1*d2+c1*a2+c1*b2+c1*c2+
c1*d2+d1*a2+d1*b2+d1*c2+d1*d2 ;//直接暴力计算输出
end
endmodule

```

仿真文件

```

`timescale 1ns / 1ps
/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/11/08 15:24:43
// Design Name:
// Module Name: simu
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module simu;
reg [31:0] a1,b1,c1,d1,a2,b2,c2,d2;
wire [31:0] sum;
//reg clk;
//parameter STEP=100.0000;

//always#(STEP/2) begin
//    clk <= ~clk;
//end

initial
begin

```

```

#0      a1= 32'd2;
        b1= 32'd2;
        c1= 32'd2;
        d1= 32'd2;
        a2= 32'd2;
        b2= 32'd2;
        c2= 32'd2;
        d2= 32'd2;

#10     $stop;

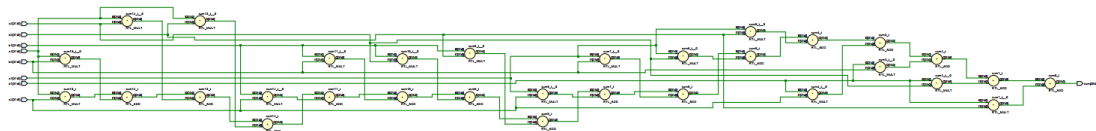
end

test regfile(
    .a1(a1),
    .b1(b1),
    .c1(c1),
    .d1(d1),
    .a2(a2),
    .b2(b2),
    .c2(c2),
    .d2(d2),
    .sum(sum)
//      .clk(clk)
);
endmodule

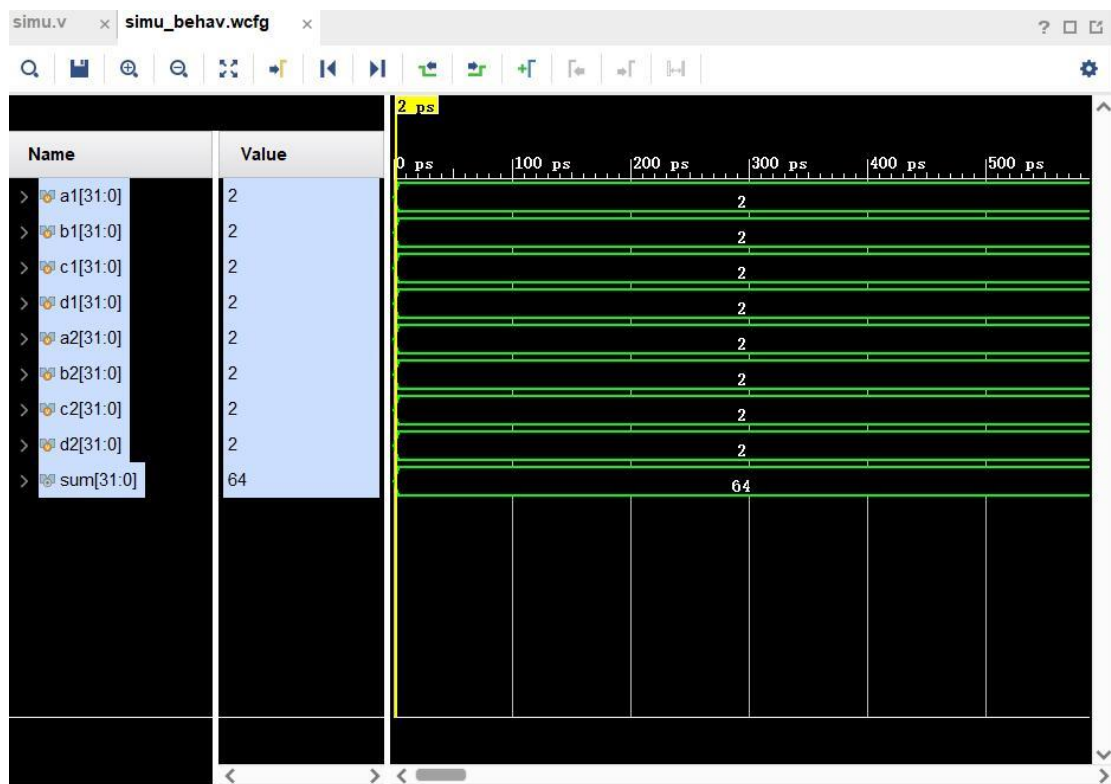
```

此版本直接使用加法器和乘法器进行运算。

仿真电路



运行结果



优点

简单明了，便于观察和修改。

缺点

过于复杂，可拓展性差。

版本 2.0

程序文件

```

////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/11/15 15:17:47
// Design Name:
// Module Name: test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//

```



```
////////////////////////////////////
```

```
module test#(
    parameter CORE2X2_0 = 4'b0010, //初始化 2*2 的卷积核为 2, 2, 2, 2
    parameter CORE2X2_1 = 4'b0010,
    parameter CORE2X2_2 = 4'b0010,
    parameter CORE2X2_3 = 4'b0010)
(
    //定义时钟信号, 复位信号
    input CLK,
    input RSTn,
    input read,
    output [7:0]sumout, //输出最终加和后的卷积结果
    output [7:0] CONV_OUT //输出卷积得到的 3*3 矩阵每一位的值
);

reg [3:0] ROM [15:0]; //存储补零后的 4*4 数据矩阵
reg [5:0] rd; //辅助卷积核进行移动的 flag
reg [7:0] CONV_OUT_REG; //卷积得到的 3*3 矩阵的每一项输出
reg [3:0] WIN0; //在数据矩阵中取出的 2*2 子阵
reg [3:0] WIN1;
reg [3:0] WIN2;
reg [3:0] WIN3;
reg [7:0] sum;

initial //被卷积矩阵的初始化（外面加一圈 0）
begin
    ROM[ 0] = 4'b0000;
    ROM[ 1] = 4'b0000;
    ROM[ 2] = 4'b0000;
    ROM[ 3] = 4'b0000;
    ROM[ 4] = 4'b0000;
    ROM[ 5] = 4'b0001;
    ROM[ 6] = 4'b0010;
    ROM[ 7] = 4'b0000;
    ROM[ 8] = 4'b0000;
    ROM[ 9] = 4'b0011;
    ROM[10] = 4'b0100;
    ROM[11] = 4'b0000;
    ROM[12] = 4'b0000;
    ROM[13] = 4'b0000;
    ROM[14] = 4'b0000;
    ROM[15] = 4'b0000;
end
```

```

always @ (posedge CLK or negedge RSTn)
begin
    if (!RSTn)
        begin
            WIN0 <= 2'bx;
            WIN1 <= 2'bx;
            WIN2 <= 2'bx;
            WIN3 <= 2'bx;
            rd <= 0;
        end
    else if (read)
        begin
            WIN0 = ROM[rd]; //在被卷积矩阵中读取 2*2 子阵, 初始选择 0, 1, 4, 5
            WIN1 = ROM[rd + 4'b0001];
            WIN2 = ROM[rd + 4'b0100];
            WIN3 = ROM[rd + 4'b0101];
            rd <= rd + 1'b1; //选择完成后向后滑动一格
        end
end

always @ ( posedge CLK or negedge RSTn )
begin//生成电路图
    if(!RSTn)
        begin
        end
    else
        begin
            case (rd)//辅助卷积核在卷积运算中换行
                6'd2:rd <= rd + 2'b10;
                6'd6:rd <= rd + 2'b10;
                6'd10:rd <= rd + 2'b10;
            endcase
        end
end

always @ ( posedge CLK or negedge RSTn )
begin
    if (!RSTn)
        CONV_OUT_REG <= 0;
        sum<=0;
    else
        //计算卷积得到的 3*3 矩阵的每一位

```

```

        CONV_OUT_REG <= WIN0 * CORE2X2_3 + WIN1 * CORE2X2_2
            + WIN2 * CORE2X2_1 + WIN3 * CORE2X2_0 ;
        sum<=sum+CONV_OUT_REG;//求和
    end

    assign CONV_OUT = CONV_OUT_REG;//卷积得到的 3×3 矩阵结果输出
    assign sumout=sum; //求和后的卷积结果输出
endmodule

```

仿真文件

```

/////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/11/15 15:18:09
// Design Name:
// Module Name: sim
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////

module sim();
    reg CLK;
    reg RSTn;
    reg read;

    wire [7:0] CONV_OUT;
    wire [7:0] sumout;

    initial
    begin
        CLK = 0;
        forever #100 CLK = ~CLK;
    end
end

```

```

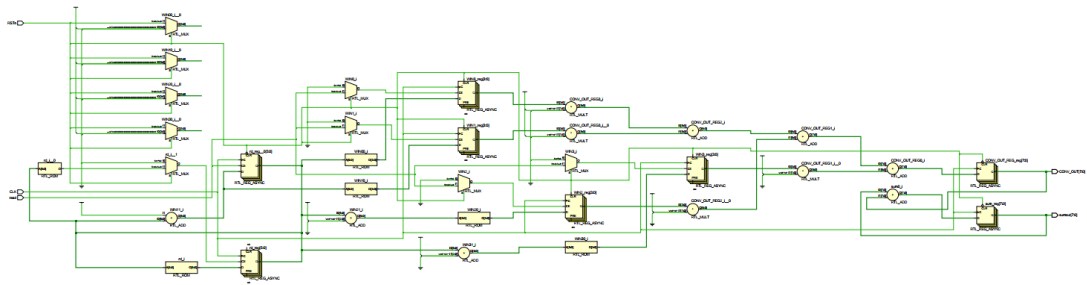
initial
begin
    RSTn = 0;
    #50 RSTn = 1;
    read = 1;
end

test test(.CLK(CLK),
    .RSTn(RSTn),
    .read(read),
    .CONV_OUT(CONV_OUT),
    .sumout(sumout)
);

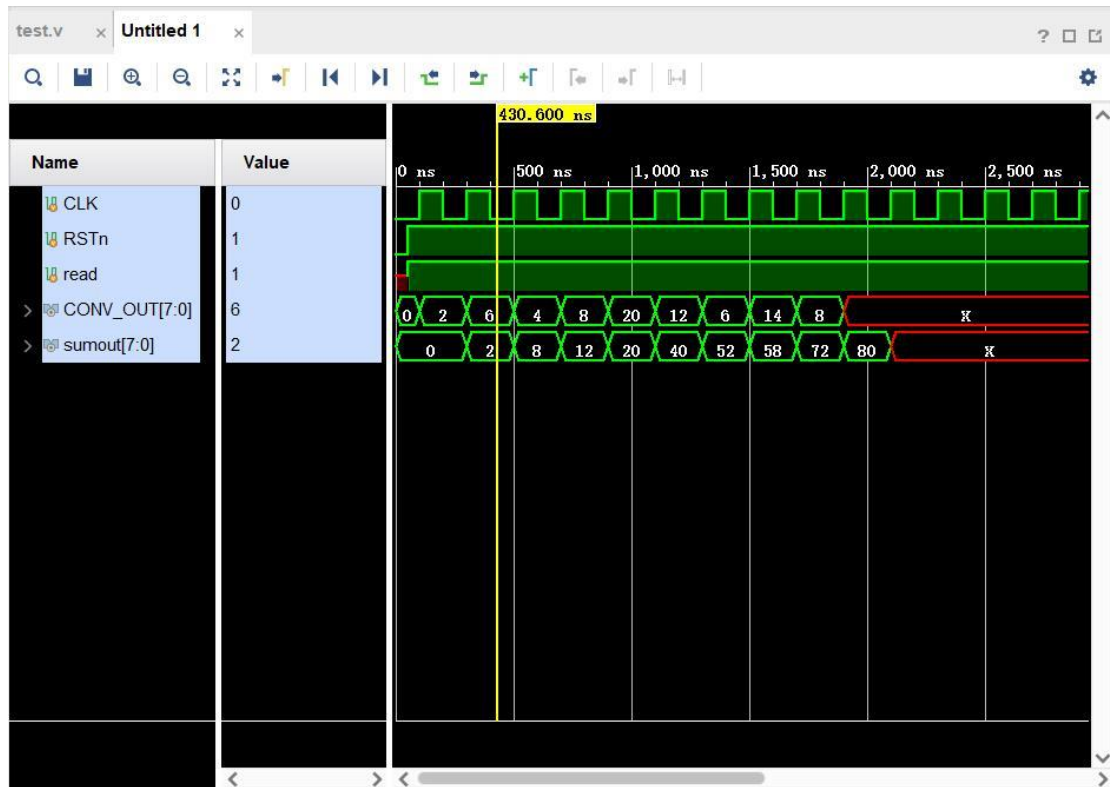
endmodule

```

仿真电路



运行结果



优点

电路图简单，清晰，模拟了卷积的过程。

版本 3.0

在版本 1.0 的基础上加入了乘法器 mult:

```
function [7:0] mult;
    input[3:0] ina,inb;
    reg[7:0] temp,ci;
    integer i,j;
begin
    mult=8'h00;
    for(i=0;i<4;i=i+1)
    begin
        if(inb&(1<<i))
        begin
            temp=ina<<i;
        end
        else
        begin
            temp=8'h00;
        end
    end
    begin
        for(j=0;j<8;j=j+1)
        begin
```

```

        if(j==0)
            begin
                ci[j]=mult[j]&temp[j];
                mult[j]=mult[j]^temp[j];
            end
        else
            begin

                ci[j]=(mult[j]&temp[j])|(mult[j]&ci[j-1])|(temp[j]&ci[j
-1]);

                mult[j]=mult[j]^temp[j]^ci[j-1];
            end
        end
    end
end
endfunction

```

并在之后的卷积运算中调用 mult 进行计算:

```

CONV_OUT_REG <= mult(WIN0 , CORE2X2_3) + mult(WIN1 , CORE2X2_2) + mult(WIN2 , C
ORE2X2_1) + mult(WIN3 , CORE2X2_0) ;

```

程序文件

```

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 2019/11/21 18:24:47
// Design Name:
// Module Name: test
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

module test#(
    parameter CORE2X2_0 = 4'b0010,

```

```

parameter CORE2X2_1 = 4'b0010,
parameter CORE2X2_2 = 4'b0010,
parameter CORE2X2_3 = 4'b0010
)
(
    input CLK,
    input RSTn,
    input read,
    output [7:0]sumout,
    output [7:0] CONV_OUT
);

```

```

reg [3:0] ROM [15:0];
reg [5:0] rd;
reg [7:0] CONV_OUT_REG;
reg [3:0] WIN0;
reg [3:0] WIN1;
reg [3:0] WIN2;
reg [3:0] WIN3;
reg [7:0] sum;

```

```
initial
```

```
begin
```

```

    ROM[ 0] = 4'b0000;
    ROM[ 1] = 4'b0000;
    ROM[ 2] = 4'b0000;
    ROM[ 3] = 4'b0000;
    ROM[ 4] = 4'b0000;
    ROM[ 5] = 4'b0001;
    ROM[ 6] = 4'b0010;
    ROM[ 7] = 4'b0000;
    ROM[ 8] = 4'b0000;
    ROM[ 9] = 4'b0011;
    ROM[10] = 4'b0100;
    ROM[11] = 4'b0000;
    ROM[12] = 4'b0000;
    ROM[13] = 4'b0000;
    ROM[14] = 4'b0000;
    ROM[15] = 4'b0000;

```

```
end
```

```

function [7:0] mult;
    input[3:0] ina,inb;
    reg[7:0] temp,ci;

```

```

    integer i,j;
begin
    mult=8'h00;
    for(i=0;i<4;i=i+1)
    begin
        if(inb&(1<<i))
            begin
                temp=ina<<i;
            end
        else
            begin
                temp=8'h00;
            end
        begin
            for(j=0;j<8;j=j+1)
            begin
                if(j==0)
                    begin
                        ci[j]=mult[j]&temp[j];
                        mult[j]=mult[j]^temp[j];
                    end
                else
                    begin
                        ci[j]=(mult[j]&temp[j])|(mult[j]&ci[j-1])|(temp[j]&ci[j-1])
;
                        mult[j]=mult[j]^temp[j]^ci[j-1];
                    end
                end
            end
        end
    end
endfunction

always @ (posedge CLK or negedge RSTn)
begin
    if (!RSTn)
        begin
            WIN0 <= 2'bx;
            WIN1 <= 2'bx;
            WIN2 <= 2'bx;
            WIN3 <= 2'bx;
            rd <= 0;

        end

```



```

else if (read)
begin
    WIN0 = ROM[rd];
    WIN1 = ROM[rd + 4'b0001];
    WIN2 = ROM[rd + 4'b0100];
    WIN3 = ROM[rd + 4'b0101];
    rd <= rd + 1'b1;
end
end

always @ ( posedge CLK or negedge RSTn )
begin
    if(!RSTn)
        begin
            end
        else
            begin
                case (rd)
                    6'd2:rd <= rd + 2'b10;
                    6'd6:rd <= rd + 2'b10;
                    6'd10:rd <= rd + 2'b10;
                endcase
            end
        end

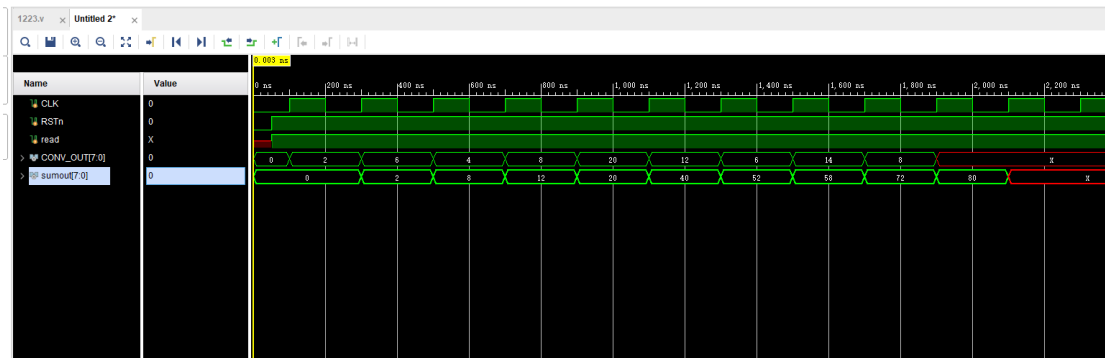
always @ ( posedge CLK or negedge RSTn )
begin
    if (!RSTn)
        CONV_OUT_REG <= 0;
        sum<=0;
    else
        CONV_OUT_REG <= WIN0 * CORE2X2_3 + WIN1 * CORE2X2_2
            + WIN2 * CORE2X2_1 + WIN3 * CORE2X2_0 ;
        sum<=sum+CONV_OUT_REG;
    end

assign CONV_OUT = CONV_OUT_REG;
assign sumout=sum;

endmodule

```

运行结果



反思

通过本次实验，我们初步掌握了 vivado 仿真软件的使用，学习了二维卷积运算的过程及原理，并用 vivado 软件初步实现了二维卷积运算的电路实现，并且通过三种方法的对比，深刻体会到了时序逻辑的重要性，并且激发了我们对电子线路设计的好奇和热情。