Systolic\_array

module systolic\_array(

input logic [2:0][7:0] A\_out, B\_out,

input logic [2:0][2:0] newmatrix,

input logic clk, rst,

output logic [2:0][2:0][7:0] x, y,

output logic [2:0][2:0][15:0] result

);

logic [2:0][2:0][15:0] c, z;

genvar i;

generate for (j=0; j<3;j++)

begin

assign z[0][j]= (newmatrix[0][j] == 0) ?

x[0][j] \* y[0][j] + c[0][j] :

x[0][j] \* y[0][j] ;

assign z[1][j]= (newmatrix[1][j] == 0) ?

x[1][j] \* y[1][j] + c[1][j] :

x[1][j] \* y[1][j] ;

assign z[2][j]= (newmatrix[0][j] == 0) ?

x[2][j] \* y[2][j] + c[2][j] :

x[2][j] \* y[2][j] ;

end

endgenerate

always\_ff @(posedge clk or posedge rst)

begin

if (rst)

begin

c <= 144'b0;

end

else

begin

c <= z;

x[0][0] <= A\_out[0];

x[1][0] <= A\_out[1];

x[2][0] <= A\_out[2];

x[0][1] <= x[0][0];

x[1][1] <= x[1][0];

x[2][1] <= x[2][0];

x[0][2] <= x[0][1];

x[1][2] <= x[1][1];

x[2][2] <= x[2][1];

y[0][0] <= B\_out[0];

y[0][1] <= B\_out[1];

y[0][2] <= B\_out[2];

y[1][0] <= y[0][0];

y[1][1] <= y[0][1];

y[1][2] <= y[0][2];

y[2][0] <= y[1][0];

y[2][1] <= y[1][1];

y[2][2] <= y[1][2];

end

end

assign result = c;

endmodule

assign result = c;

endmodule

mat\_to\_stream

1. בשלב ראשון מגדירים את הinterface של המודול.

כניסות לאברי המטריצות a,b.

יציאות לווקטורי השורה/עמודה A\_out ו B\_out.

סיגנלי newdata,clk,rst.

הגדרת סיגנלים איתם ממשים את המערכים באמצעותם נממש shift\_register.

מימוש מערך של shift\_registers עבור r כפי שרואים בתמונה.

כניסה לתהליך בעליית שעון או בעליית ריסט. בריסט כל הרגיסטרים מאופסים. אם אין ריסט ויש עליית שעון כאשר newdata = 0 מבצעים הזזה. כאשר אין ריסט ו newdata = 1 מבצעים השמה של ערכי מטריצה a למקומות הנכונים. בr14,r24,r23 מבצעים הזזה וr00,r01,r10 מאופסים.

מימוש מערך של shift\_registers עבור s כפי שרואים בתמונה.

ההסבר דומה מאוד להסבר שרשמנו עבור r.

השמה קבועה של תוצאות ההזזה ליציאות.

module mat\_to\_stream (

input [2:0][2:0][7:0] a, b

input newdata,

input clk, rst

output [2:0][7:0] A\_out,B\_out );

// RTL part

logic [2:0][4:0][7:0] r;

logic [4:0][2:0][7:0] s;

// Shift Register For Matrix a

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

always\_ff @(posedge clk or posedge rst)

if (rst)

begin

r <= 120'h0;

end

else if (newdata==1'b0)

begin

r[0][4] <= r[0][3];

r[1][4] <= r[1][3];

r[2][4] <= r[2][3];

r[0][3] <= r[0][2];

r[1][3] <= r[1][2];

r[2][3] <= r[2][2];

r[0][2] <= r[0][1];

r[1][2] <= r[1][1];

r[2][2] <= r[2][1];

r[0][1] <= r[0][0];

r[1][1] <= r[1][0];

r[2][1] <= r[2][0];

r[0][0] <= 8'h0;

r[1][0] <= 8'h0;

r[2][0] <= 8'h0;

end

else // new data

begin

r[0][4] <= a[0][0];

r[1][4] <= r[1][3];

r[2][4] <= r[2][3];

r[0][3] <= a[0][1];

r[1][3] <= a[1][0];

r[2][3] <= r[2][2];

r[0][2] <= a[0][2];

r[1][2] <= a[1][1];

r[2][2] <= a[2][0];

r[0][1] <= 8'h0;

r[1][1] <= a[1][2];

r[2][1] <= a[2][1];

r[0][0] <= 8'h0;

r[1][0] <= 8'h0;

r[2][0] <= a[2][2];

end

end

always\_ff @(posedge clk or posedge rst)

if (rst)

begin

s <= 120'h0;

end

else if (newdata==1'b0)

begin

s[4][0] <= s[3][0];

s[4][1] <= s[3][1];

s[4][2] <= s[3][2];

s[3][0] <= s[2][0];

s[3][1] <= s[2][1];

s[3][2] <= s[2][2];

s[2][0] <= s[1][0];

s[2][1] <= s[1][1];

s[2][2] <= s[1][2];

s[1][0] <= s[0][0];

s[1][1] <= s[0][1];

s[1][2] <= s[0][2];

s[0][0] <= 8'h0;

s[0][1] <= 8'h0;

s[0][2] <= 8'h0;

end

else

begin

s[4][0] <= b[0][0];

s[4][1] <= s[3][1];

s[4][2] <= s[3][2];

s[3][0] <= b[1][0];

s[3][1] <= b[0][1];

s[3][2] <= s[2][2];

s[2][0] <= b[2][0];

s[2][1] <= b[1][1];

s[2][2] <= b[0][2];

s[1][0] <= s[0][0];

s[1][1] <= b[2][1];

s[1][2] <= b[1][2];

s[0][0] <= 8'h0;

s[0][1] <= 8'h0;

s[0][2] <= b[2][2];

end

end

assign A\_out[0]=r[0][4];

assign A\_out[1]=r[1][4];

assign A\_out[2]=r[2][4];

assign B\_out[0]=S[4][0];

assign B\_out[1]=S[4][1];

assign B\_out[2]=S[4][2];

endmodule

מימוש מכונת המצבים:

|  |  |  |
| --- | --- | --- |
| V | [0:2][0:2]newmatrix |  |
| **0** |  | **LOAD\_XY** |
| **0** |  | **S1** |
| **0** |  | **S2** |
| **0** |  | **S3** |

הקוד:

module systolic\_array\_fsm (

input clk, rst, newdata, finish,

output logic V,

output logic [2:0][2:0] newmatrix

);

typedef enum bit[2:0] {IDLE=3'b000, LOAD\_XY=3'b001, S1=3'b010,

S2=3'b011, S3=3'b100, VIOLATION=3'b101} STATE;

STATE CUR\_ST;

STATE NEXT\_ST;

always\_ff @(posedge clk or posedge rst)

begin

if (rst==1)

CUR\_ST=IDLE;

else

CUR\_ST=NEXT\_ST;

end

always\_comb

begin

V=0;

case(CUR\_ST)

IDLE:

begin

V=0;

newmatrix={9{1'b0}};

if (newdata==1'b1)

NEXT\_ST=LOAD\_XY;

else

NEXT\_ST=CUR\_ST;

end

LOAD\_XY:

begin

newmatrix={9{1'b0}};

if ((newdata==1'b1)||(finish==1'b1))

NEXT\_ST=VIOLATION;

else

NEXT\_ST=S1;

end

S1:

begin

newmatrix={9{1'b0}};

newmatrix[0][0]=1'b1;

newmatrix[2][1]=1'b1;

newmatrix[1][2]=1'b1;

if (newdata==1'b1)

NEXT\_ST=VIOLATION;

else if (finish==1'b1)

NEXT\_ST=IDLE;

else

NEXT\_ST=S2;

end

S2:

begin

newmatrix={9{1'b0}};

newmatrix[0][1]=1'b1;

newmatrix[1][0]=1'b1;

newmatrix[2][2]=1'b1;

if (finish==1'b1)

NEXT\_ST=VIOLATION;

else

NEXT\_ST=S3;

end

S3:

begin

newmatrix={9{1'b0}};

newmatrix[0][2]=1'b1;

newmatrix[2][0]=1'b1;

newmatrix[1][1]=1'b1;

if ((newdata==1'b1)||(finish==1'b1))

NEXT\_ST=VIOLATION;

else

NEXT\_ST=S1;

end

VIOLATION:

begin

newmatrix={9{1'b0}};

V=1;

if (rst==1'b1)

NEXT\_ST=IDLE;

else

NEXT\_ST=CUR\_ST;

end

endcase

end

endmodule

**mat\_mul\_test:**

הסטים הנדרשים לבדיקה:



הקוד:

module mat\_mul\_test;

logic rst,clk,newdata;

logic [2:0][2:0]newmatrix;

logic [2:0][2:0][7:0] a,b; // input matrices

logic V,finish;

logic [2:0][2:0][7:0] x,y;

logic [2:0][2:0][15:0] result; // testbench

results

logic [2:0][7:0] A\_out, B\_out; // output

vectors

initial begin //setting initial values

rst = 1'b1;

finish = 1'b0;

// matrix a,b - first pair

a[0]={8'd0,8'd1,8'd0};

a[1]={8'd3,8'd0,8'd0};

a[2]={8'd0,8'd1,8'd2};

b[0]={8'd2,8'd2,8'd0};

b[1]={8'd0,8'd0,8'd0};

b[2]={8'd1,8'd5,8'd0};

#1;

clk = 1'b0; // reset the clk signal at #1

newdata = 1'b0;

#10;

rst = 1'b0;

newdata = 1'b1;

#10;

newdata = 1'b0;

#20; //second pair

a[0]={8'd0,8'd0,8'd3};// 1st row

a[1]={8'd8,8'd7,8'd3};

a[2]={8'd3,8'd6,8'd1};

b[0]={8'd6,8'd3,8'd0};// 1st row

b[1]={8'd1,8'd1,8'd8};

b[2]={8'd0,8'd8,8'd1};

#10;

newdata = 1'b0;

#70;

finish = 1'b1;

#150;

$finish;

end

always begin

#5 clk = ~clk;

end

mat\_to\_stream mat\_mul\_tb1 (.\*);

systolic\_array\_fsm mat\_mul\_tb2 (.\*);

systolic\_array mat\_mul\_tb3 (.\*);

endmodule

assertions:

assertions:

sequence rose\_nd;

$rose(newdata);

endsequence

sequence fell\_nd;

$fell(newdata);

endsequence

sequence rose\_f;

$rose(finish);

endsequence

sequence fell\_rst;

$fell(rst);

endsequence

test\_en1:assert property (@ (posedge clk iff rst==1'b0) fell\_rst|->##1

rose\_nd))

else $warning ("Warning! newdata rise before reset fall");

test\_en2:assert property (@ (posedge clk iff rst==1'b0) (fell\_nd|

rise\_nd)|->##5 rose\_f))

else $warning ("Warning! newdata rise before reset fall");

שאלות

1. מכונת Mealy היא מכונה בה המוצא תלוי בכניסות ובמצב הנוכחי, ואילו במכונת Moore המוצא תלוי אך ורק במצב הנוכחי.
2. המכונה בתרגיל היא מכונת Moore.
3. S1.
4. S3.
5. S1.
6. S2.
7. S1.
8. ארבעה מחזורי שעון.
9. שלושה מחזורי שעון.
10. שישה מחזורי שעון.
11. שני מחזורי שעון.
12. מחזור שעון בודד.
13. שבעה מחזורי שעון.
14. החישוב הראשון של המטריצה השנייה.
15. 1/3
16. 8 מחזורי שעון.
17. על פניו נראה כי הקוד שכתבנו תקין ואין צורך בשינויים.