新的Data\_Avg模块，通过此模块完成输入的值选取有效值并且对数值进行计算的功能。是在先前的模块中加入了平均值计算的功能，采用的思路是先写入寄存器的所有数据，然后写计数器比读计数器慢一拍，形成加一个数的同时减一个数的效果，并且对所有数据进行平均值计算。

//计算前一个模块输出四个有效值的平均值

module Data\_Avg(

#(parameter N = 5)

input req,

input frame\_clk,

input clk\_2,

input rst,

input [3:0] data,

output [3:0] avg

);

reg [3:0] cnt;

wire wr\_req; //Write Request

wire rd\_req; //Read Request

wire [3:0] data\_cal;

reg clk;

reg rst\_n;

reg [3:0] wr\_cnt; //Count write times

reg [3:0] wr\_cnt\_r;

reg [3:0] rd\_cnt; //Count read times

reg [4:0] data\_sum; //Result of all data adding up

reg [4:0] data\_average; //Average

reg [3:0] DFF[3:0]; //Register reuse

assign clk = clk\_2;

assign rst\_n = !rst;

assign wr\_req = 1'b1; //Always enable writing request

////// \*\*\*\*\*\* MUX of [3:0] data and '0' \*\*\*\*\*\* //////

always @(posedge clk) begin

if(!req)

data\_cal <= 4'b0000;

else

data\_cal <= data;

end

////// \*\*\*\*\*\* Algorithm Unit \*\*\*\*\*\* //////

//Counter control

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

cnt <= 4'b0000;

else if(cnt == N)

cnt <= 4'b0000;

else

cnt <= cnt + 1'b1;

end

//Begin reading when all data are in

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

rd\_req <= 1'b0;

else if(cnt == N-1)

rd\_req <= 1'b1;

else

rd\_req <= rd\_req;

end

//Addition of numbers

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

data\_sum <= 'd0;

else if(cnt == N)

data\_sum <= data\_sum + DFF[wr\_cnt] - DFF[rd\_cnt];

else

data\_sum <= data\_sum + DFF[wr\_cnt];

end

//Division of all data

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

data\_average <= 'd0;

else if(cnt == N)

data\_average <= data\_sum/(N-1);

else

data\_average <= 'd0;

end

//Output

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

avg <= 'd0;

else

avg <= data\_average;

end

//Control write counter

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

wr\_cnt\_r <= 'd0;

else if(wr\_cnt\_r == 4)

wr\_cnt\_r <= 'd0;

else

wr\_cnt\_r <= wr\_cnt\_r + 1'b1;

end

//Write counter

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

wr\_cnt <= 'd0;

else

wr\_cnt <= wr\_cnt\_r;

end

//Read counter

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n)

rd\_cnt <= 'd0;

else if(rd\_cnt == 4)

rd\_cnt <= 'd0;

else

rd\_cnt <= rd\_cnt +1'b1;

end

//Initialize

integer i;

always @(posedge clk or negedge rst\_n) begin

if(!rst\_n) begin

for(i=0;i<4;i=i+1'b1)

DFF[i] <= 'd0;

end

else if(wr\_req == 1'b1)

DFF[wr\_cnt\_r] <= data\_cal;

end

////// \*\*\*\*\*\* End of Algorithm Unit \*\*\*\*\*\* //////

Endmodule

仿真结果：

