加入了握手同步机制，对于握手机制的学习参考了以下几个链接

<https://blog.csdn.net/qq_39814612/article/details/105795252>

<https://blog.csdn.net/weixin_43067657/article/details/89645337>

<https://blog.csdn.net/m0_57113037/article/details/120928334>

自己先做了一个练手

module tx(tclk,reset\_tclk,t\_rdy,data\_avail,transmit\_data,t\_data,r\_ack);

input tclk;

input reset\_tclk;

input data\_avail;

input [31:0]transmit\_data;

input r\_ack;

output t\_rdy;

output t\_data;

localparam IDLE\_T = 2'd0,

ASSERT\_T\_RDY = 2'd1,

DEASSERT\_T\_RDY = 2'd2;

reg [1:0] t\_hndshk\_state,t\_hndshk\_state\_nxt;

reg t\_rdy,t\_rdy\_nxt;

reg [31:0] t\_data,t\_data\_nxt;

reg r\_ack\_tclk;

always@(\*)begin

t\_hndshk\_state\_nxt = t\_hndshk\_state;

t\_rdy\_nxt = 1'b0;

t\_data\_nxt = t\_data;

case(t\_hndshk\_state)

IDLE\_T:begin

if(data\_avail) begin

t\_rdy\_nxt = 1'b1;

t\_hndshk\_state\_nxt = ASSERT\_T\_RDY;

t\_data\_nxt = transmit\_data;

end

end

ASSERT\_T\_RDY:begin

if(r\_ack\_tclk)begin

t\_rdy\_nxt = 1'b0;

t\_hndshk\_state\_nxt = DEASSERT\_T\_RDY;

t\_data\_nxt = 'd0;

end

else begin

t\_rdy\_nxt = 1'b1;

t\_data\_nxt = transmit\_data;

end

end

DEASSERT\_T\_RDY:begin

if(!r\_ack\_tclk)begin

if(data\_avail)begin

t\_rdy\_nxt = 1'b1;

t\_hndshk\_state\_nxt = ASSERT\_T\_RDY;

t\_data\_nxt = transmit\_data;

end

else begin

t\_hndshk\_state\_nxt = IDLE\_T;

end

end

end

endcase

end

always@(posedge tclk or negedge reset\_tclk)begin

if(!reset\_tclk)begin

t\_rdy <= 1'b0;

t\_hndshk\_state <= IDLE\_T;

t\_data <= 32'h00000000;

r\_ack\_tclk <= 1'b0;

end

else begin

t\_rdy <= t\_rdy\_nxt;

t\_hndshk\_state <= t\_hndshk\_state\_nxt;

t\_data <= t\_data\_nxt;

r\_ack\_tclk <= r\_ack;

end

end

endmodule

module rx(rclk,reset\_rclk,t\_rdy,t\_data,r\_ack);

input rclk,reset\_rclk;

input t\_rdy;

input[31:0] t\_data;

output r\_ack;

reg r\_hndshk\_state,r\_hndshk\_state\_nxt;

reg t\_rdy\_rclk;

reg[31:0] t\_data\_rclk,t\_data\_rclk\_nxt;

reg r\_ack,r\_ack\_nxt;

localparam IDLE\_R = 1'b0,

ASSERT\_ACK = 1'b1;

always@(\*)begin

r\_hndshk\_state\_nxt = r\_hndshk\_state;

r\_ack\_nxt = 1'b0;

t\_data\_rclk\_nxt = t\_data\_rclk;

case(r\_hndshk\_state)

IDLE\_R:begin

if(t\_rdy\_rclk)begin

r\_hndshk\_state\_nxt = ASSERT\_ACK;

t\_data\_rclk\_nxt = t\_data;

r\_ack\_nxt = 1'b1;

end

end

ASSERT\_ACK:begin

if(!t\_rdy\_rclk)begin

r\_hndshk\_state\_nxt = IDLE\_R;

r\_ack\_nxt = 1'b0;

end

else begin

r\_ack\_nxt = 1'b1;

end

end

endcase

end

always@(posedge rclk or negedge reset\_rclk)begin

if(!reset\_rclk)begin

r\_hndshk\_state <= IDLE\_R;

t\_data\_rclk <= 1'b0;

t\_rdy\_rclk <= 1'b0;

r\_ack <= 1'b0;

end

else begin

r\_hndshk\_state <= r\_hndshk\_state\_nxt;

t\_data\_rclk <= t\_data\_rclk\_nxt;

t\_rdy\_rclk <= t\_rdy;

r\_ack <= r\_ack\_nxt;

end

end

endmodule

module tb;

reg tclk\_tb,rclk\_tb;

reg [31:0] transmit\_data\_tb;

reg reset\_tclk\_tb,reset\_rclk\_tb;

reg data\_avail\_tb;

wire t\_rdy\_tb;

wire [31:0] t\_data\_tb;

wire r\_ack\_tb;

parameter CLK\_HALF\_PERIOD1 = 5;

parameter CLK\_HALF\_PERIOD2 = 8;

parameter RESET\_DELAY = 100;

initial begin

tclk\_tb = 0;

rclk\_tb = 0;

end

always #CLK\_HALF\_PERIOD1 tclk\_tb = ~tclk\_tb;

always #CLK\_HALF\_PERIOD2 rclk\_tb = ~rclk\_tb;

initial begin

reset\_rclk\_tb = 0;

reset\_tclk\_tb = 0;

#RESET\_DELAY reset\_rclk\_tb = 1;

reset\_tclk\_tb = 1;

end

initial begin

#500;

data\_avail\_tb = 1;

transmit\_data\_tb = 32'h96431346;

end

transmit transmit\_test

(.tclk(tclk\_tb),

.reset\_tclk(reset\_tclk\_tb),

.t\_rdy(t\_rdy\_tb),

.data\_avail(data\_avail\_tb),

.transmit\_data(transmit\_data\_tb),

.t\_data(t\_data\_tb),

.r\_ack(r\_ack\_tb));

receiver receiver\_test

(.rclk(rclk\_tb),

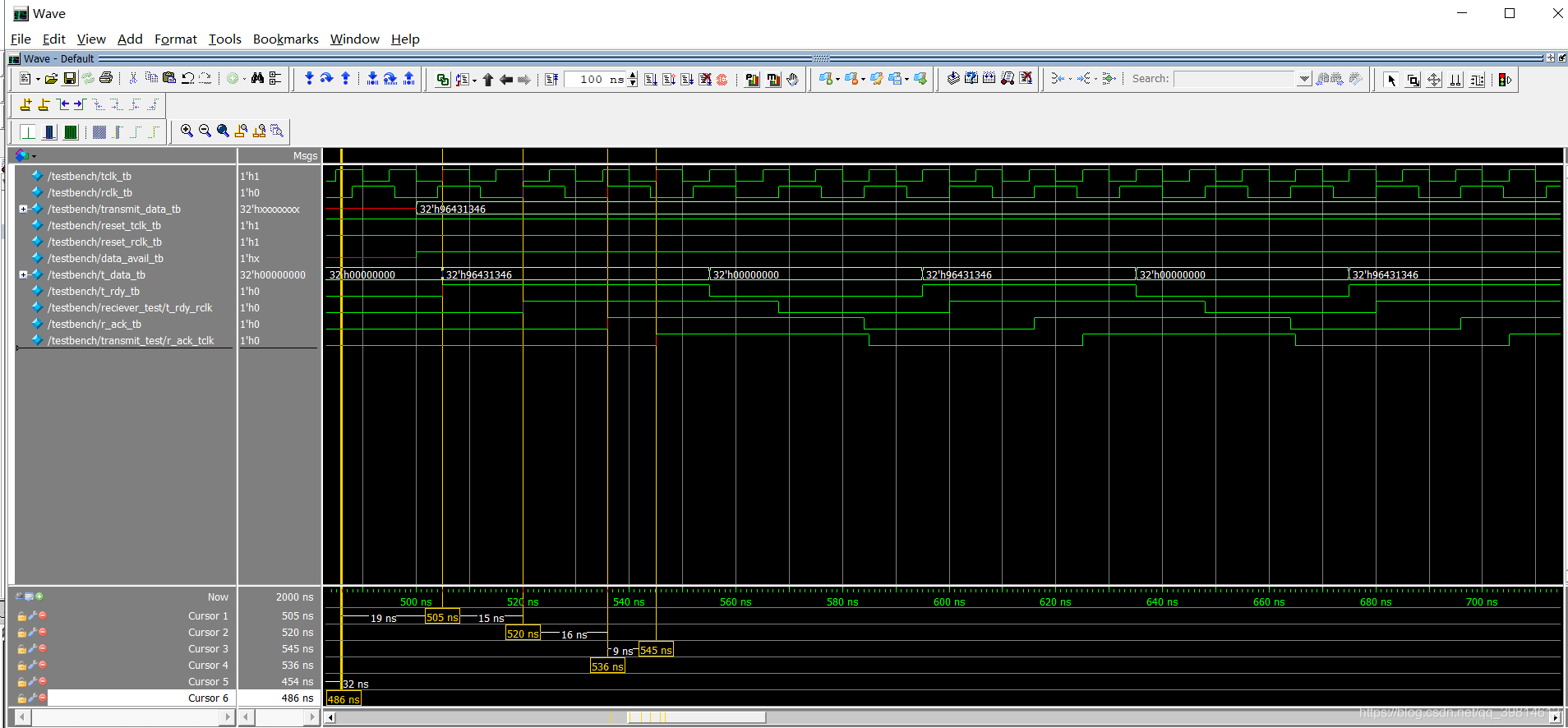
.reset\_rclk(reset\_rclk\_tb),

.t\_rdy(t\_rdy\_tb),

.t\_data(t\_data\_tb),

.r\_ack(r\_ack\_tb));

endmodule



对于这次作业，并不需要特别复杂的握手机制，简单的半握手机制便可以完成

module adv\_hw8\_2

(

input req,

input clk\_2,

input rst,

input [3:0] data,

output [3:0] avg

);

reg [2:0]cnt;//握手次数计数器

reg DFF1,DFF2;//req的同步器

//reg [3:0] data\_DFF1,data\_DFF2;//数据缓冲器

reg [5:0] AU;//算数单元

reg [3:0] current\_state,next\_state;//状态寄存器

//req经同步器同步

always@(posedge clk\_2)

begin

DFF1 <= req;

DFF2 <= DFF1;

end

parameter

IDLE = 4'd0,

REQ\_ACTIVE = 4'd1,

REQ\_RELEASE = 4'd2;

always@(posedge clk\_2 or posedge reset)

begin

if(rst)

current\_state <= IDLE;

else

current\_state <= next\_state;

end

always@(\*)

begin

next\_state = IDLE;

case(current\_state)

IDLE :

begin

cnt = 3'd0;

AU = 6'd0;

next\_state = REQ\_ACTIVE;

end

REQ\_ACTIVE :

begin

if(DFF2)

begin

cnt = cnt + 1'b1;//计数握手一次

AU = AU + data;//存入数值并计算

next\_state = REQ\_RELEASE;//握手成功，等待释放

avg = AU[5:2];//输出当前和/4的结果

end

else

next\_state = REQ\_ACTIVE;//否则继续等待握手

end

REQ\_RELEASE :

begin

if(!DFF2)

begin

next\_state = REQ\_ACTIVE;//释放成功，等待下次握手

if(cnt == 3'd4)//握手四次后清零

begin

cnt = 3'd0;

AU = 6'd0;

end

else;

end

else

next\_state = REQ\_RELEASE;//否则继续等待释放

end

default :

next\_state = IDLE;

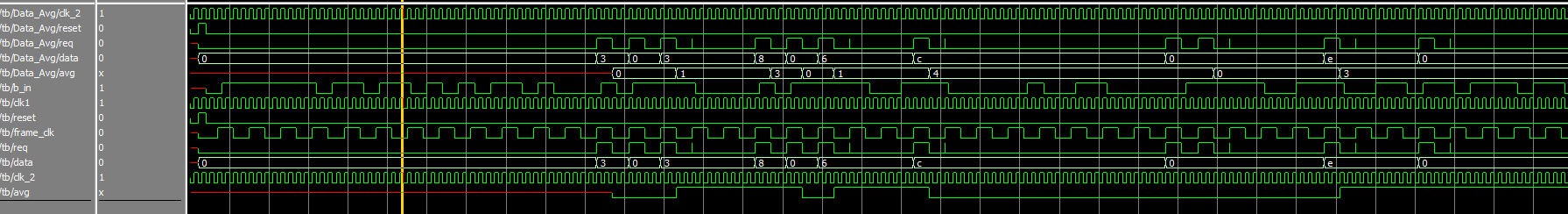
endcase

end

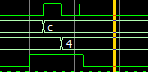
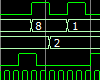
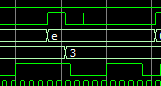
endmodule

仿真结果：

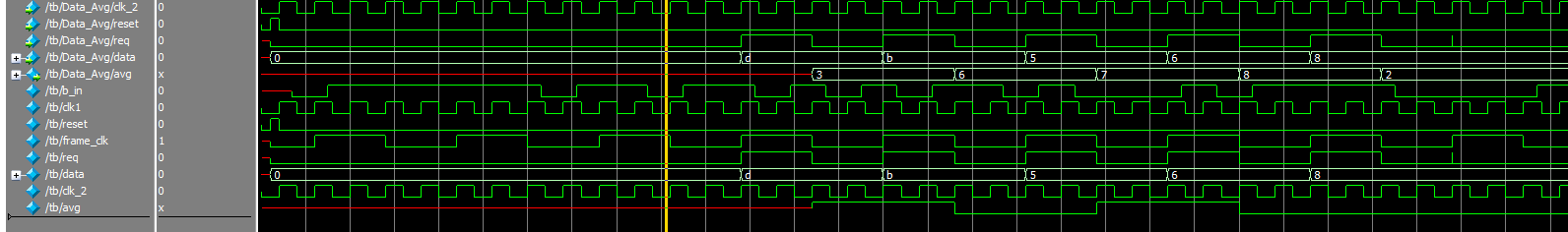
0.05x:



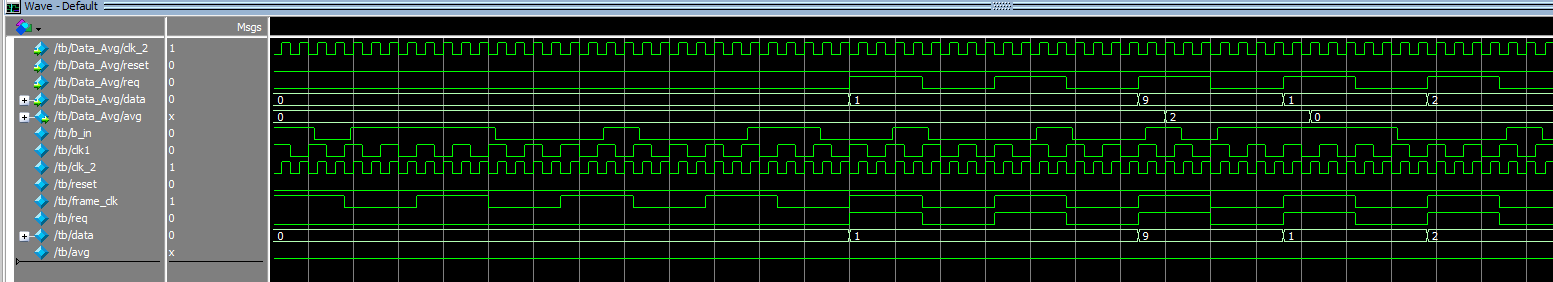
放大局部可以看到求均值的功能实现了

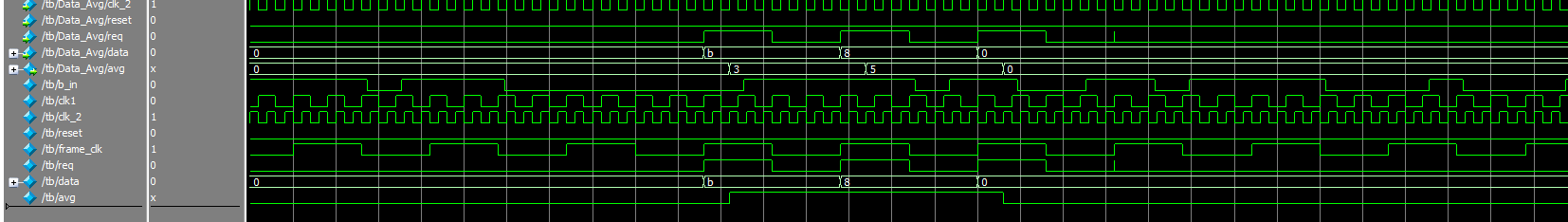
 

1x:

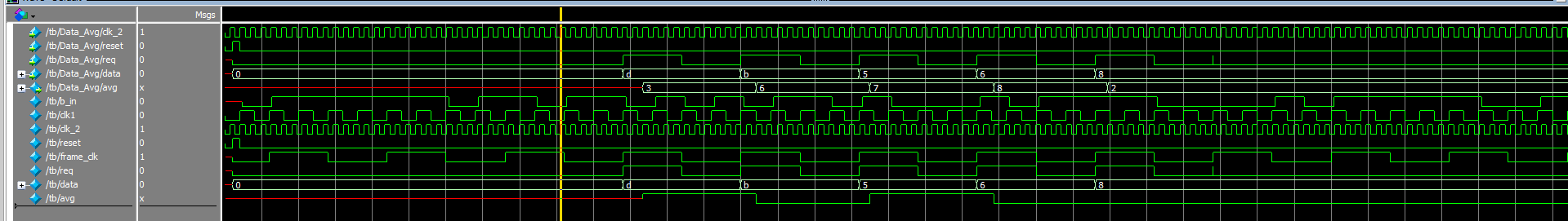


2x:

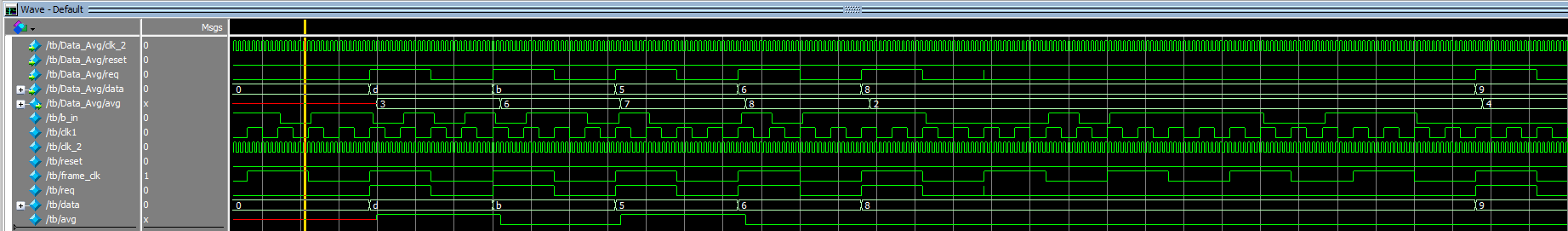




3x:



8x:



主要区别在于上一题需要注意采样信号的频率，clk\_1同时控制着两个模块

本题需要在握手的时候才传输有效数值，要在释放后等待下一个数值的输入，wave会输出一些无效数值