module Rabbit\_2\_FPGA\_2\_DDS\_FIFO(key\_3\_sweep, tenMHz\_ext, SDIO\_PE\_5\_1, SCLK\_PE\_3\_1,

SDIO, SCLK, IO\_UPDATE, DR\_CTL, OSK, CSB, DR\_HOLD);

193-

input key\_3\_sweep;

input tenMHz\_ext;

input SDIO\_PE\_5\_1;

input SCLK\_PE\_3\_1;

output reg SDIO;

output reg SCLK;

output reg IO\_UPDATE;

output reg DR\_CTL;

output reg OSK;

output reg CSB;

output reg DR\_HOLD;

integer i; //index for the memory as it is being read in

integer N; // index for the memory being read out

integer M; //tells when to send out the SDIO signal

integer L; //tells when to toggle the SCLK signal

integer E;

integer T;

reg sweep\_key\_flag; // this is high during the time that the sweep info is being sent out to the DDS

//it must be reset every time that each signal has been sent

reg \_\_\_\_\_\_\_\_;//this flag is: when it is set sweep\_key\_flag can be set back to zero

reg \_\_\_\_\_\_\_\_; //this flag will occur when the whole memory\_reg has been sent (when N>=i)

//it resets the N and i indices to zero so the thing can be restarted

reg [0:3679] memory\_reg;

//this section of code deals with making the decisions about when to start doing what

//its main job is to make sure the FPGA knows when to send the signal to the DDS

// and when to reset everything

always@ (negedge tenMHz\_ext)

begin

CSB <=0;//must be tied to zero to have DDS recieve

OSK <=0;

DR\_HOLD <=0;//must be tied to zero to let the DDS work smooth

//this sets the sweep\_key\_flag when the key\_3 is depressed and sweep\_key\_flag

// has not been set in 1 sec to "debounce" the keys. \_\_\_\_\_\_\_\_ is

//set when sweep\_key\_flag has been set for 1 sec

if ((key\_3\_sweep!= 1) && (\_\_\_\_\_\_\_\_ == 0))

begin

sweep\_key\_flag <=1;

end

if (\_\_\_\_\_\_\_\_ != 0)

begin

sweep\_key\_flag <=0;

end

//this section will set the \_\_\_\_\_\_\_\_ which will indicate that the whole set of sweep has been run

//this happens when N>=i this will reset everything important including setting N and i back to Zero

if (N>=i)

begin

\_\_\_\_\_\_\_\_ <=1;

end

if (N<i)

begin

\_\_\_\_\_\_\_\_<=0;

end

end

//this section of code deals with sending out the signal to the DDS

always@ (posedge tenMHz\_ext)

begin

if (\_\_\_\_\_\_\_\_ !=0)

begin

N<=0;

end

//when \_\_\_\_\_\_\_\_ is high sweep\_key\_flag can be reset the number mus be so high because the trigger from the keys can be >1sec

if (T>=20000000)

begin

\_\_\_\_\_\_\_\_ <= 1;

T<=0;

E<=E+1;

end

if (T<20000000)

begin

\_\_\_\_\_\_\_\_ <=0;

end

if (sweep\_key\_flag != 0)

begin

//T increments while sweep\_key\_flag is high. when T >=20,000,000 sweep\_key\_flag will go back to zero

T<=T+1;

end

//this is the section of code that toggles the IO\_UPDATE

if ((T>= 422) && (T<427))

begin

IO\_UPDATE <= 1;

end

if ((T<422) || (T>=427))

begin

IO\_UPDATE <=0;

end

//this is the section of code where the message is actually being sent

//when this section is not occuring SCLK and SDIO is low

if ((T>=5) && (T<373))

begin

if(L>=1)

begin

L<=0;

SCLK<=1;

end

if(L<1)

begin

L<=L+1;

SCLK<=0;

end

if (M < 1)

begin

M<=M+1;

//the main difference with this program to have multiple sweeps

//is that N is not reset after every sweep

SDIO <= memory\_reg[N];

end

if (M >= 1)

begin

N<= N+1;

M<=0;

end

end

if ((T<5) || (T>=373))

begin

SCLK<=0;

SDIO<=0;

end

//13

//this section of code toggles SCLK in the appropriate manner

if ((T>=446) && (T<451))

begin

if (E<=18)

begin

if (E<=17)

begin

if (E<=16)

begin

if (E<=15)

begin

if (E<=14)

begin

if (E<=13)

begin

if (E<=12)

begin

if (E<=11)

begin

if (E<=10)

begin

if (E<=9)

begin

if (E<=8)

begin

if (E<=7)

begin

if (E<=6)

begin

if (E<=5)

begin

if (E<=4)

begin

if (E<=5)

begin

if (E<=3)

begin

if (E<=2)

begin

if (E<=1)

begin

if (E<=0)

begin

if (memory\_reg[8:39] > memory\_reg[40:71])

/\*13+22=35 \*/ begin

DR\_CTL<=0;

end

if (memory\_reg[8:39] < memory\_reg[40:71])

begin

DR\_CTL<=1;

end

end

else

begin

if (memory\_reg[184+8:184+39] > memory\_reg[184+40:184+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184+8:184+39] < memory\_reg[184+40:184+71])

begin

DR\_CTL<=1;

end

end

end

end else

begin

if (memory\_reg[184\*2+8:184\*2+39] > memory\_reg[184\*2+40:184\*2+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*2+8:184\*2+39] < memory\_reg[184\*2+40:184\*2+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*3+8:184\*3+39] > memory\_reg[184\*3+40:184\*3+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*3+8:184\*3+39] < memory\_reg[184\*3+40:184\*3+71])

/\*10+35=45\*/ begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*4+8:184\*4+39] > memory\_reg[184\*4+40:184\*4+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*4+8:184\*4+39] < memory\_reg[184\*4+40:184\*4+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*5+8:184\*5+39] > memory\_reg[184\*5+40:184\*5+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*5+8:184\*5+39] < memory\_reg[184\*5+40:184\*5+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*6+8:184\*6+39] > memory\_reg[184\*6+40:184\*6+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*6+8:184\*6+39] < memory\_reg[184\*6+40:184\*6+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*7+8:184\*7+39] > memory\_reg[184\*7+40:184\*7+71])

/\*11+45=56\*/ begin

DR\_CTL<=0;

end

if (memory\_reg[184\*7+8:184\*7+39] < memory\_reg[184\*7+40:184\*7+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*8+8:184\*8+39] > memory\_reg[184\*8+40:184\*8+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*8+8:184\*8+39] < memory\_reg[184\*8+40:184\*8+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*9+8:184\*9+39] > memory\_reg[184\*9+40:184\*9+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*9+8:184\*9+39] < memory\_reg[184\*9+40:184\*9+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*10+8:184\*10+39] > memory\_reg[184\*10+40:184\*10+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*10+8:184\*10+39] < memory\_reg[184\*10+40:184\*10+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*11+8:184\*11+39] > memory\_reg[184\*11+40:184\*11+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*11+8:184\*11+39] < memory\_reg[184\*11+40:184\*11+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*12+8:184\*12+39] > memory\_reg[184\*12+40:184\*12+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*12+8:184\*12+39] < memory\_reg[184\*12+40:184\*12+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*13+8:184\*13+39] > memory\_reg[184\*13+40:184\*13+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*13+8:184\*13+39] < memory\_reg[184\*13+40:184\*13+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*14+8:184\*14+39] > memory\_reg[184\*14+40:184\*14+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*14+8:184\*14+39] < memory\_reg[184\*14+40:184\*14+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*15+8:184\*15+39] > memory\_reg[184\*15+40:184\*15+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*15+8:184\*15+39] < memory\_reg[184\*15+40:184\*15+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*16+8:184\*16+39] > memory\_reg[184\*16+40:184\*16+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*16+8:184\*16+39] < memory\_reg[184\*16+40:184\*16+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*17+8:184\*17+39] > memory\_reg[184\*17+40:184\*17+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*17+8:184\*17+39] < memory\_reg[184\*17+40:184\*17+71])

begin //31+56=87

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*18+8:184\*18+39] > memory\_reg[184\*18+40:184\*18+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*18+8:184\*18+39] < memory\_reg[184\*18+40:184\*18+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*19+8:184\*19+39] > memory\_reg[184\*19+40:184\*19+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184\*19+8:184\*19+39] < memory\_reg[184\*19+40:184\*19+71])

begin

DR\_CTL<=1;

end

end

end

//87+6=93

if ((T<446) || (T>=451))

begin

if (E<=18)

begin

if (E<=17)

begin

if (E<=16)

begin

if (E<=15)

begin

if (E<=14)

begin

if (E<=13)

begin

if (E<=12)

begin

if (E<=11)

begin

if (E<=10)

begin

if (E<=9)

begin

if (E<=8)

begin

if (E<=7)

begin

if (E<=6)

begin

if (E<=5)

begin

if (E<=4)

begin

if (E<=5)

begin

if (E<=3)

begin

if (E<=2)

begin

if (E<=1)

begin

if (E<=0)

begin

if (memory\_reg[8:39] > memory\_reg[40:71])

begin

DR\_CTL<=1;

end

if (memory\_reg[8:39] < memory\_reg[40:71])

begin

DR\_CTL<=0;

end

end

else

begin

if (memory\_reg[184+8:184+39] > memory\_reg[184+40:184+71])

begin

DR\_CTL<=0;

end

if (memory\_reg[184+8:184+39] < memory\_reg[184+40:184+71])

begin

DR\_CTL<=1;

end

end

end

else

begin

if (memory\_reg[184\*2+8:184\*2+39] > memory\_reg[184\*2+40:184\*2+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*2+8:184\*2+39] < memory\_reg[184\*2+40:184\*2+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*3+8:184\*3+39] > memory\_reg[184\*3+40:184\*3+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*3+8:184\*3+39] < memory\_reg[184\*3+40:184\*3+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*4+8:184\*4+39] > memory\_reg[184\*4+40:184\*4+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*4+8:184\*4+39] < memory\_reg[184\*4+40:184\*4+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*5+8:184\*5+39] > memory\_reg[184\*5+40:184\*5+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*5+8:184\*5+39] < memory\_reg[184\*5+40:184\*5+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*6+8:184\*6+39] > memory\_reg[184\*6+40:184\*6+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*6+8:184\*6+39] < memory\_reg[184\*6+40:184\*6+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*7+8:184\*7+39] > memory\_reg[184\*7+40:184\*7+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*7+8:184\*7+39] < memory\_reg[184\*7+40:184\*7+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*8+8:184\*8+39] > memory\_reg[184\*8+40:184\*8+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*8+8:184\*8+39] > memory\_reg[184\*8+40:184\*8+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*9+8:184\*9+39] > memory\_reg[184\*9+40:184\*9+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*9+8:184\*9+39] < memory\_reg[184\*9+40:184\*9+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*10+8:184\*10+39] > memory\_reg[184\*10+40:184\*10+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*10+8:184\*10+39] < memory\_reg[184\*10+40:184\*10+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*11+8:184\*11+39] > memory\_reg[184\*11+40:184\*11+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*11+8:184\*11+39] < memory\_reg[184\*11+40:184\*11+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*12+8:184\*12+39] > memory\_reg[184\*12+40:184\*12+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*12+8:184\*12+39] < memory\_reg[184\*12+40:184\*12+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*13+8:184\*13+39] > memory\_reg[184\*13+40:184\*13+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*13+8:184\*13+39] < memory\_reg[184\*13+40:184\*13+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*14+8:184\*14+39] > memory\_reg[184\*14+40:184\*14+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*14+8:184\*14+39] < memory\_reg[184\*14+40:184\*14+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*15+8:184\*15+39] > memory\_reg[184\*15+40:184\*15+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*15+8:184\*15+39] < memory\_reg[184\*15+40:184\*15+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*16+8:184\*16+39] > memory\_reg[184\*16+40:184\*16+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*16+8:184\*16+39] < memory\_reg[184\*16+40:184\*16+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*17+8:184\*17+39] > memory\_reg[184\*17+40:184\*17+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*17+8:184\*17+39] < memory\_reg[184\*17+40:184\*17+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*18+8:184\*18+39] > memory\_reg[184\*18+40:184\*18+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*18+8:184\*18+39] < memory\_reg[184\*18+40:184\*18+71])

begin

DR\_CTL<=0;

end

end

end

else

begin

if (memory\_reg[184\*19+8:184\*19+39] > memory\_reg[184\*19+40:184\*19+71])

begin

DR\_CTL<=1;

end

if (memory\_reg[184\*19+8:184\*19+39] < memory\_reg[184\*19+40:184\*19+71])

begin

DR\_CTL<=0;

end

end

end

end

//this section of code deals with reading the message from the rabbit

always@ (posedge SCLK\_PE\_3\_1)

begin

if (\_\_\_\_\_\_\_\_ !=0)

begin

i<=0;

end

memory\_reg[i] <= SDIO\_PE\_5\_1;

i<=i+1;

end

endmodule