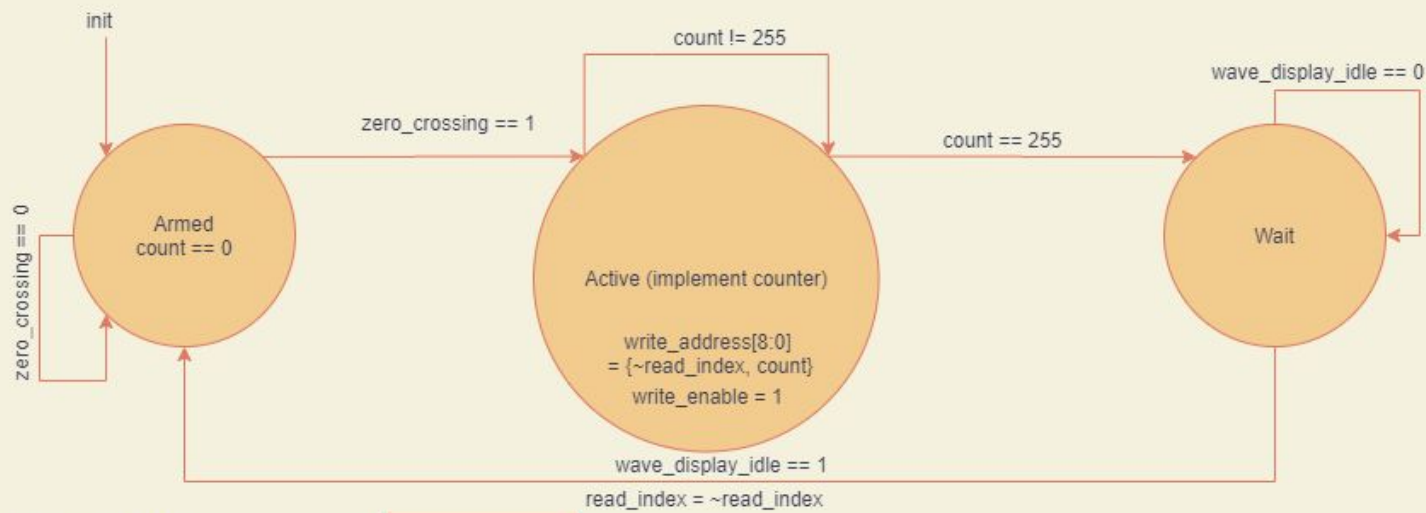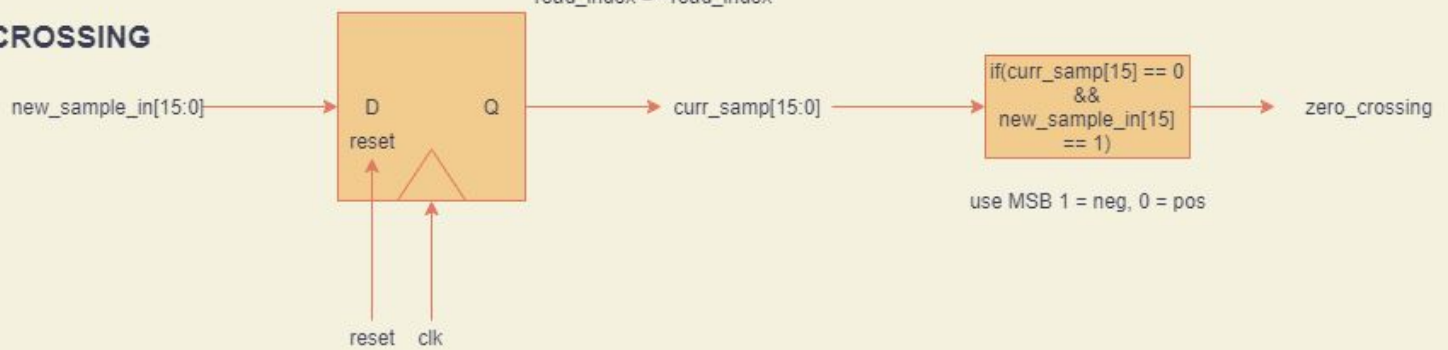# wave_capture
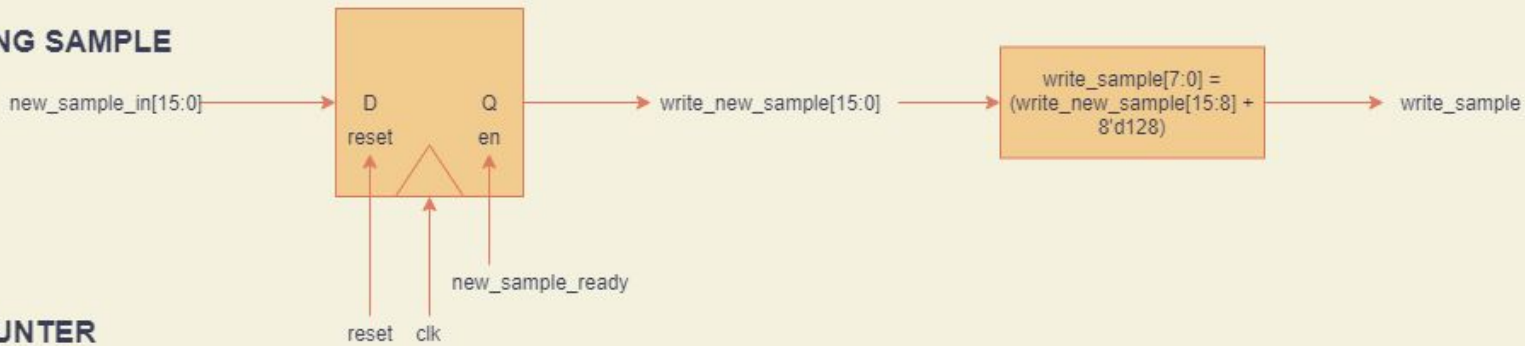
input: clk, reset, new_sample_ready, new_sample_in[15:0], wave_display_idle
output: write_address[8:0], write_enable, write_sample[7:0], read_index

init

count != 255

wave_display_idle == 0

zero_crossing == 1

count == 255

zero_crossing == 0

Armed
count == 0

Active (implement counter)

write_address[8:0]
= {~read_index, count}
write_enable = 1

Wait

wave_display_idle == 1
read_index = ~read_index

ZERO CROSSING

new_sample_in[15:0] → D Q → curr_samp[15:0] →

reset

if(curr_samp[15] == 0
&&
new_sample_in[15]
== 1) → zero_crossing

use MSB 1 = neg, 0 = pos

reset   clk

WRITING SAMPLE

new_sample_in[15:0] → D Q → write_new_sample[15:0] →

reset   en

write_sample[7:0] =
(write_new_sample[15:8] +
8'd128) → write_sample

new_sample_ready

reset   clk

COUNTER

count + 8'b1

8

D   Q → count
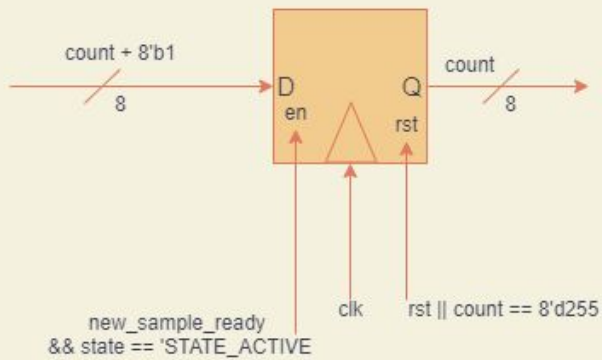en   rst

8

new_sample_ready
&& state == 'STATE_ACTIVE

clk   rst || count == 8'd255

wave_display

inputs: clk, reset, x[10:0], y[9:0], valid, read_index, read_value[7:0]
outputs: read_address[8:0], valid_pixel, r[7:0], b[7:0], g[7:0]

read_value[7:0] → read_value[7:0]

RAM

read_addr[8:0]

D   Q

rst   clk

next_read_address[8:0]

x_8 = {x_10[8], x_10[6:0]}

x_10 = x[10:1]

if x_10[9:8] == 01 || 10
read_addr = {read_index, x_8}

x[10:0]

valid_pixel → valid_pixel

D   Q

rst   clk

pixel_delay2

D   Q

rst   clk

pixel_delay

if (x_10[9:8] == 01 || 10) && (y[9] == 0)
valid_pixel = 1'b1

y[9:0]

if(y_9 >= read_value && y_9 <= curr_read_value && valid)
{r,g,b} = 24'hFFFFFF
if(y_9 <= read_value && y_9 >= curr_read_value && valid)
{r,g,b} = 24'hFFFFFF

{r,g,b}

RAM[x-1] < y < RAM[x]
RAM[x-1] > y > RAM[x]

curr_read_value

D   Q
       en
rst   clk

y_9 = y[9:1]

read_value[7:0]

next_read_addr != read_addr