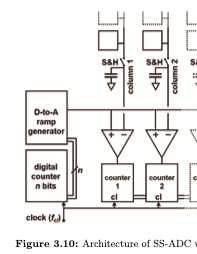
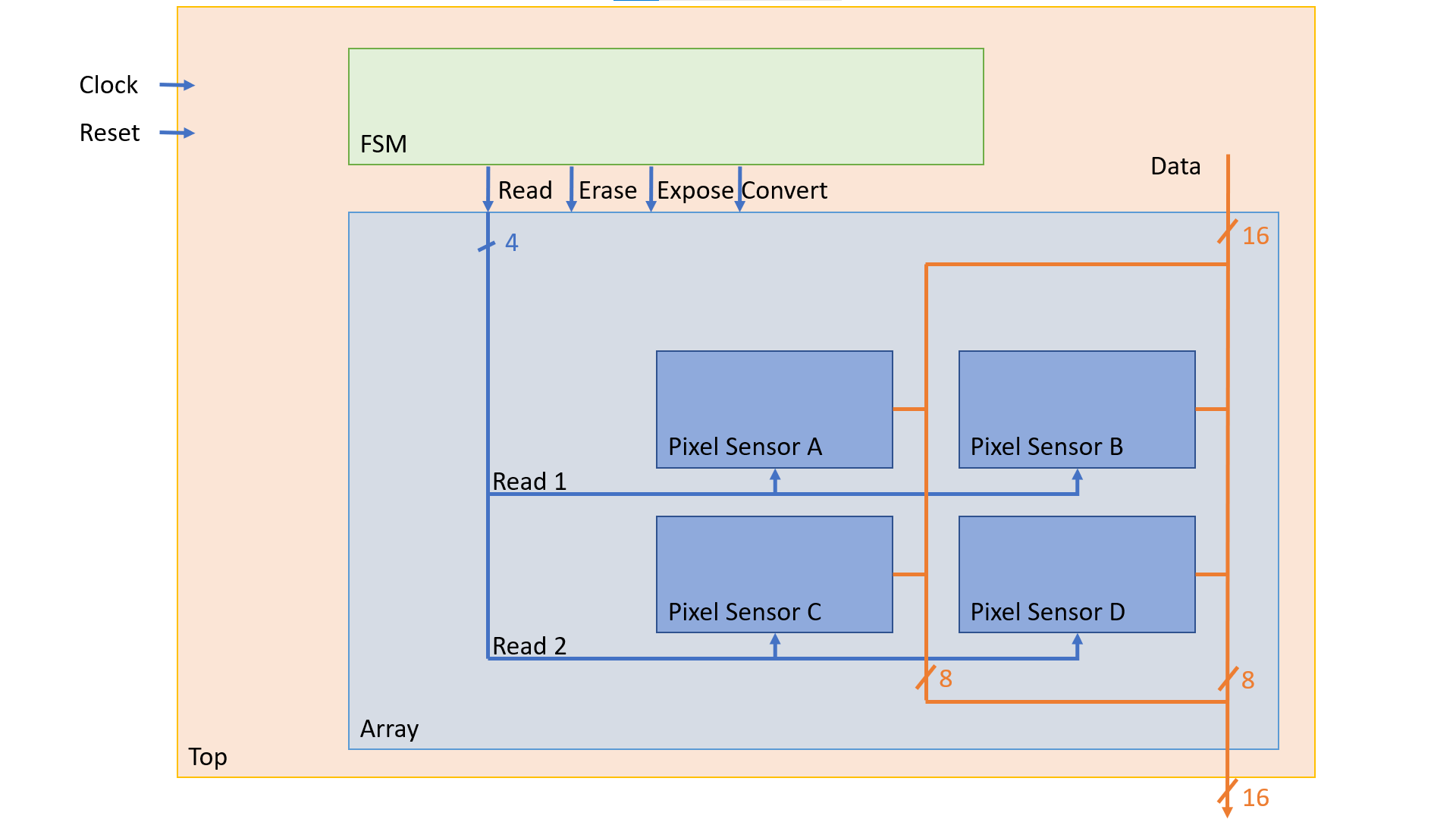
**IMAGE SENSOR SIMULATION FOR DASHCAM**  
Researcher: Lê Quang Nhật Nam

1. **Overview**





1. Simulation on Vivado

1/ Counter 8 bits

A 8-bit counter that generates the ramp code for the ADC conversion process.

* SystemVerilog code:

**module counter(input logic clk,**

**input logic reset,**

**output wire[width-1:0] out);**

**parameter width = 8;**

**logic[width-1:0] count = 0;**

**always\_ff @(posedge clk or posedge reset) begin**

**if (reset) begin**

**count <= 0;**

**end else begin**

**count <= count + 1;**

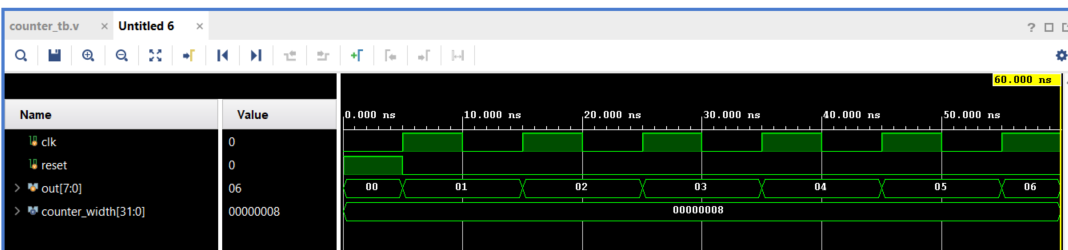
**end**

**end**

**assign out = count;**

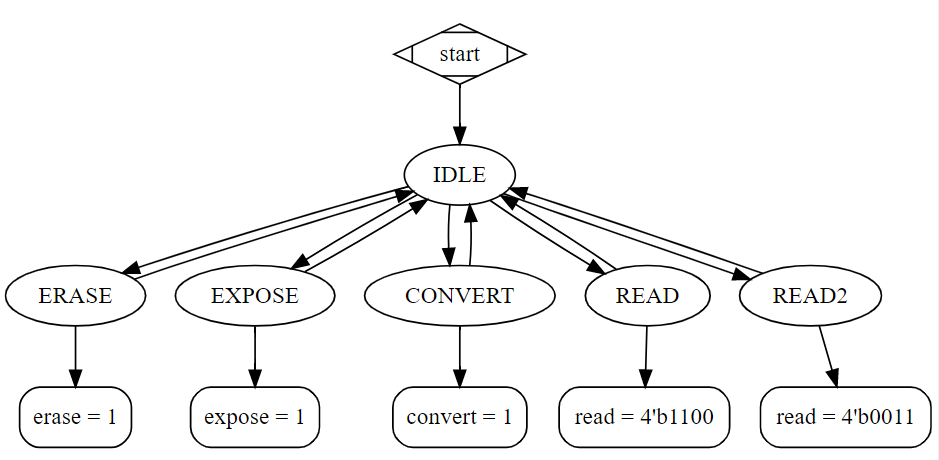
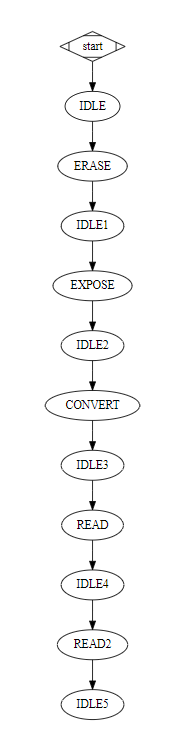
**endmodule**

* Waveform:



When the reset signal is at 0 (not activated), the counter starts to operate on the clk pulse. Each rising edge of clk increases the value of out[7:0] by 1, starting from 00, 01, 02… up to 255 because this is an 8-bit counter. The clock period in this waveform is about 10 ns, so each count value lasts for 10 ns before moving to the next value. This confirms that the counter works correctly: incrementing the count value steadily on the clock pulse when the reset is not activated.

3/ FSM



* SystemVerilog code:

**module pixelState(**

**input logic clk,**

**input logic reset,**

**output logic erase,**

**output logic expose,**

**output logic[3:0] read,**

**output logic convert**

**);**

**//State duration in clock cycles**

**parameter integer c\_erase = 5;**

**parameter integer c\_expose = 255;**

**parameter integer c\_convert = 255;**

**parameter integer c\_read = 5;**

**//------------------------------------------------------------**

**// State Machine**

**//------------------------------------------------------------**

**parameter ERASE=0, EXPOSE=1, CONVERT=2, READ=3, READ2=4, IDLE=5;**

**logic convert\_stop;**

**logic [2:0] state,next\_state; //States**

**integer counter; //Delay counter in state machine**

**// Control the output signals**

**always\_ff @(negedge clk ) begin**

**case(state)**

**ERASE: begin**

**erase <= 1;**

**read <= 4'b0;**

**expose <= 0;**

**convert <= 0;**

**end**

**EXPOSE: begin**

**erase <= 0;**

**read <= 4'b0;**

**expose <= 1;**

**convert <= 0;**

**end**

**CONVERT: begin**

**erase <= 0;**

**read <= 4'b0;**

**expose <= 0;**

**convert = 1;**

**end**

**READ: begin**

**erase <= 0;**

**read <= 4'b1100;**

**expose <= 0;**

**convert <= 0;**

**end**

**READ2: begin**

**erase <= 0;**

**read <= 4'b0011;**

**expose <= 0;**

**convert <= 0;**

**end**

**IDLE: begin**

**erase <= 0;**

**read <= 4'b0;**

**expose <= 0;**

**convert <= 0;**

**end**

**endcase // case (state)**

**end // always @ (state)**

**// Control the state transitions.**

**always\_ff @(posedge clk or posedge reset) begin**

**if(reset) begin**

**state = IDLE;**

**next\_state = ERASE;**

**counter = 0;**

**convert = 0;**

**end**

**else begin**

**case (state)**

**ERASE: begin**

**if(counter == c\_erase) begin**

**next\_state <= EXPOSE;**

**state <= IDLE;**

**end**

**end**

**EXPOSE: begin**

**if(counter == c\_expose) begin**

**next\_state <= CONVERT;**

**state <= IDLE;**

**end**

**end**

**CONVERT: begin**

**if(counter == c\_convert) begin**

**next\_state <= READ;**

**state <= IDLE;**

**end**

**end**

**READ:**

**if(counter == c\_read) begin**

**state <= IDLE;**

**next\_state <= READ2;**

**end**

**READ2:**

**if(counter == c\_read) begin**

**state <= IDLE;**

**next\_state <= ERASE;**

**end**

**IDLE:**

**state <= next\_state;**

**endcase // case (state)**

**if(state == IDLE)**

**counter = 0;**

**else**

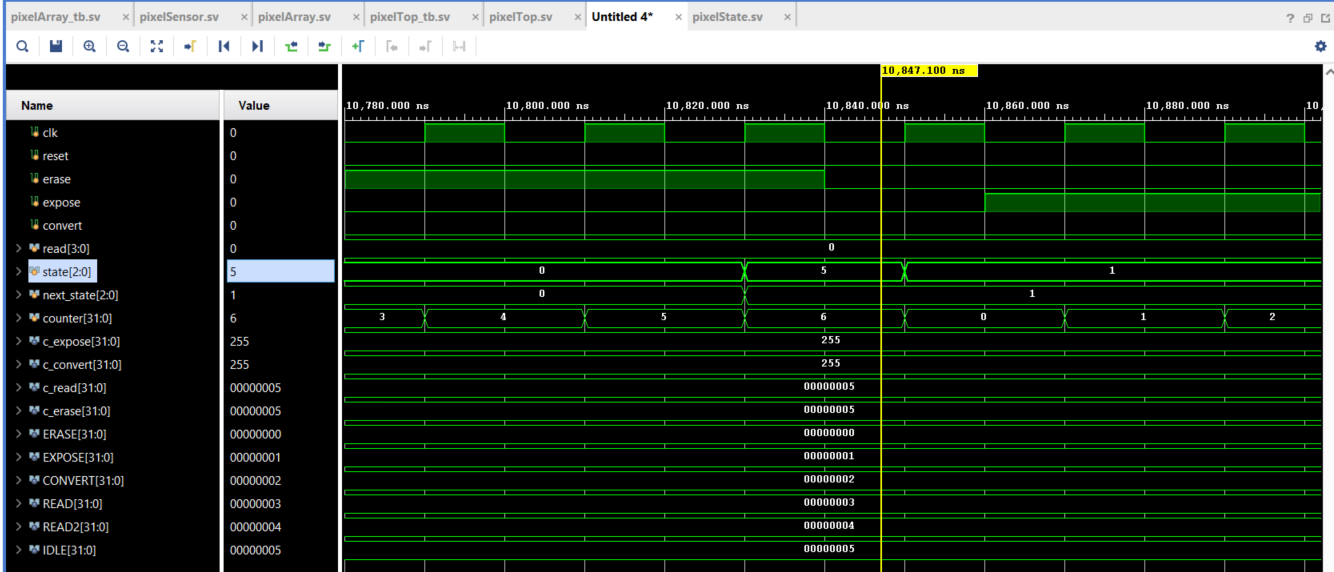
**counter = counter + 1;**

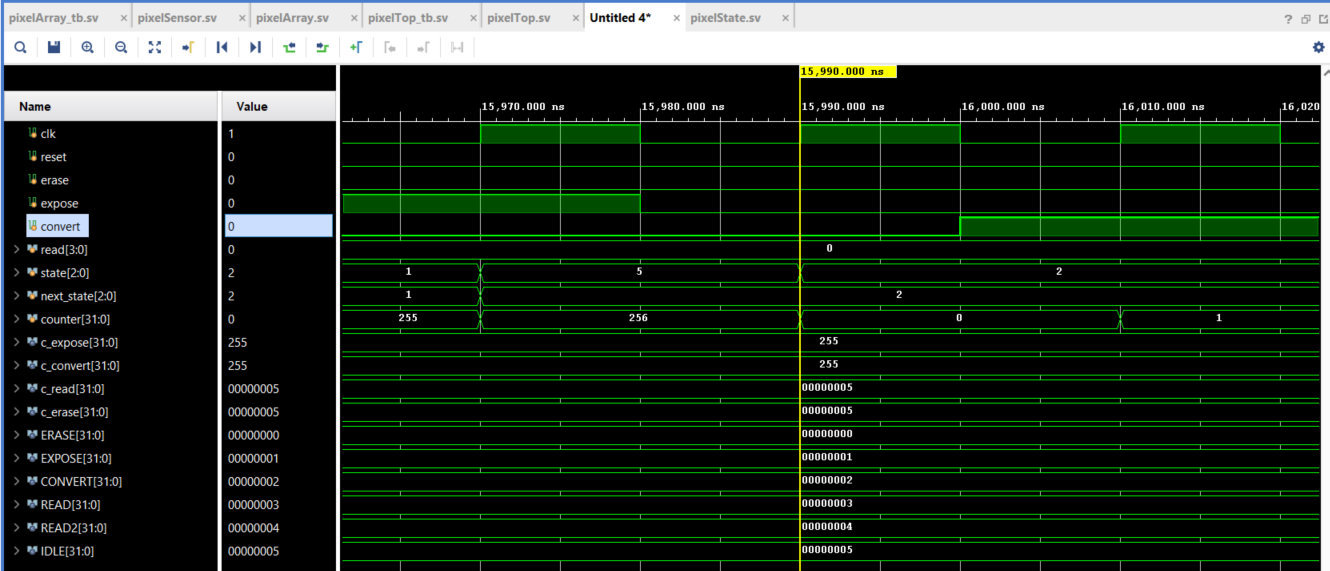
**end**

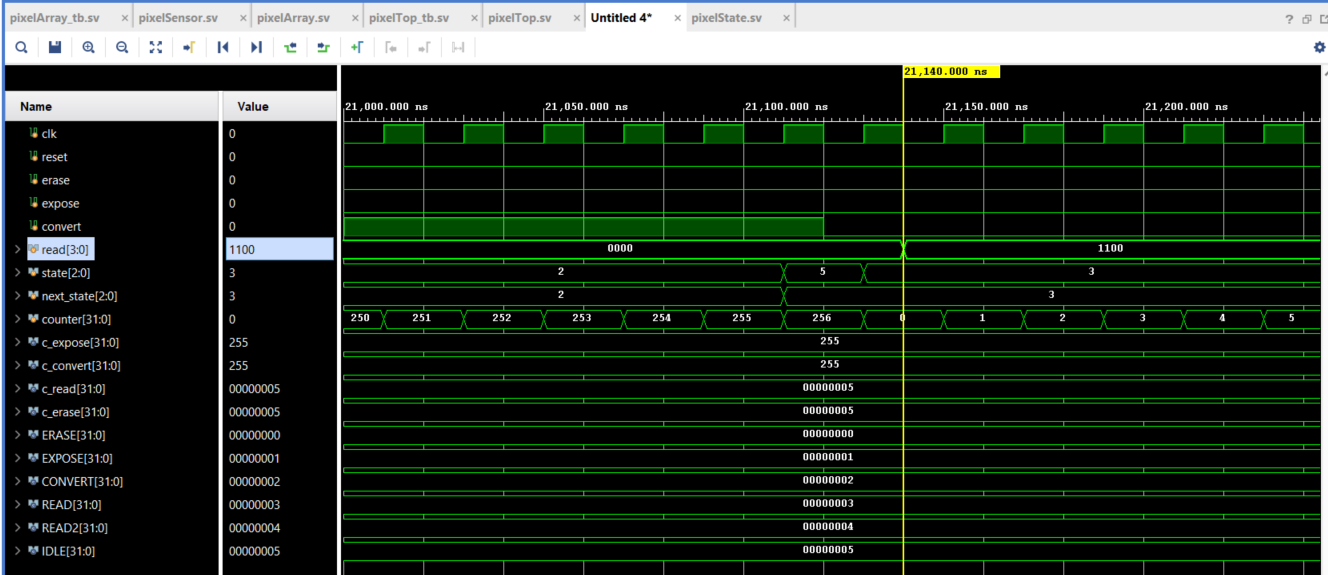
**end // always @ (posedge clk or posedge reset)**

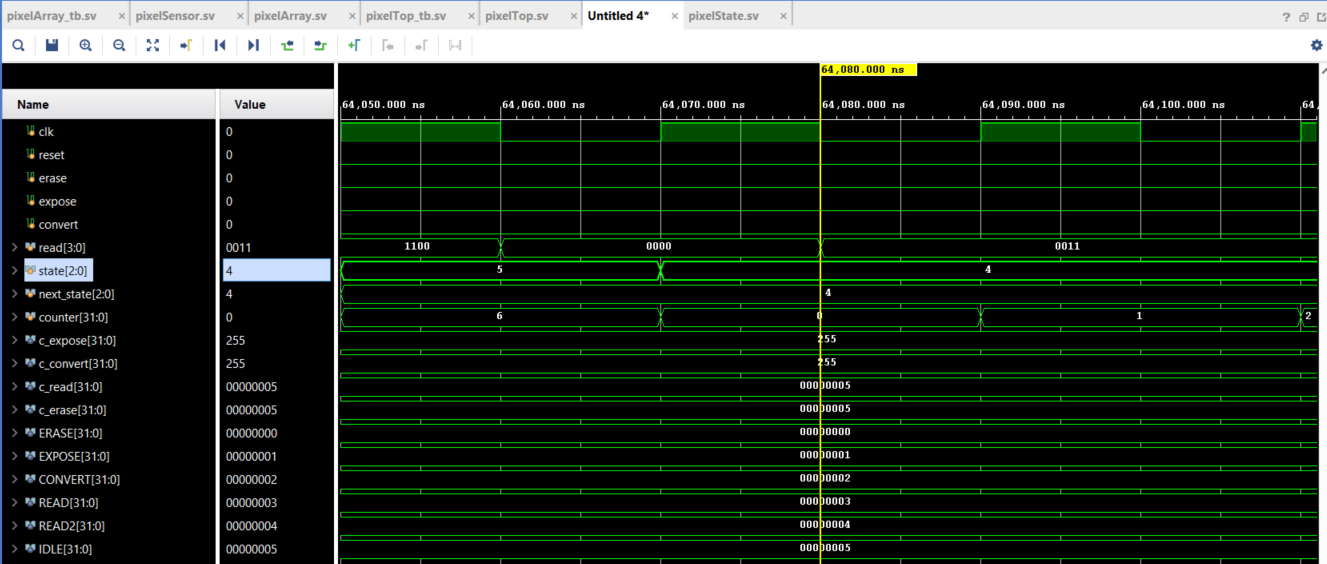
**endmodule**

- Simulation









4/ PIXEL\_SENSOR

* SystemVerilog code:

**module PIXEL\_SENSOR**

**(**

**input logic VBN1,**

**input logic RAMP,**

**input logic RESET,**

**input logic ERASE,**

**input logic EXPOSE,**

**input logic READ,**

**inout [7:0] DATA**

**);**

**// ---- analog-like params ----**

**real v\_erase = 1.2;**

**real lsb = v\_erase/255.0;**

**parameter real dv\_pixel = 0.5;**

**// ---- pixel state ----**

**real tmp; // ?i?n áp pixel (mô ph?ng)**

**real adc; // ramp analog (mô ph?ng)**

**logic cmp; // comparator tripped**

**logic [7:0] p\_data, next\_p\_data; // latch mã ??m**

**// ---- ERASE + CONVERT (??ng b?) ----**

**always\_ff @(posedge RAMP or posedge ERASE) begin**

**if (ERASE) begin**

**tmp <= v\_erase;**

**adc <= 0.0;**

**cmp <= 1'b0;**

**p\_data <= 8'h00;**

**end**

**else begin**

**adc <= adc + lsb;**

**if (adc > tmp) cmp <= 1'b1;**

**p\_data <= next\_p\_data; // c?p nh?t latch theo next-state**

**end**

**end**

**// ---- EXPOSE (tích l?y ánh sáng) ----**

**always\_ff @(posedge VBN1) begin**

**if (EXPOSE)**

**tmp <= tmp - (dv\_pixel\*lsb \* ({$random}%2));**

**end**

**// ---- next-state cho p\_data ----**

**always\_comb begin**

**next\_p\_data = p\_data; // m?c ??nh gi? nguyên**

**if (!cmp) // tr??c khi comparator trip => bám DATA**

**next\_p\_data = DATA;**

**end**

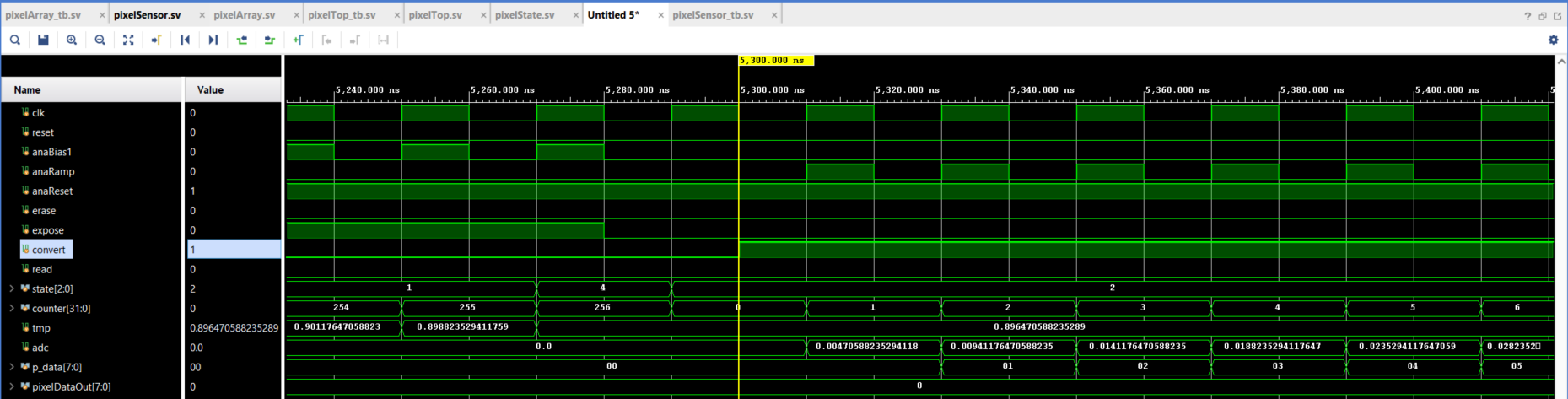
**// ---- READOUT ----**

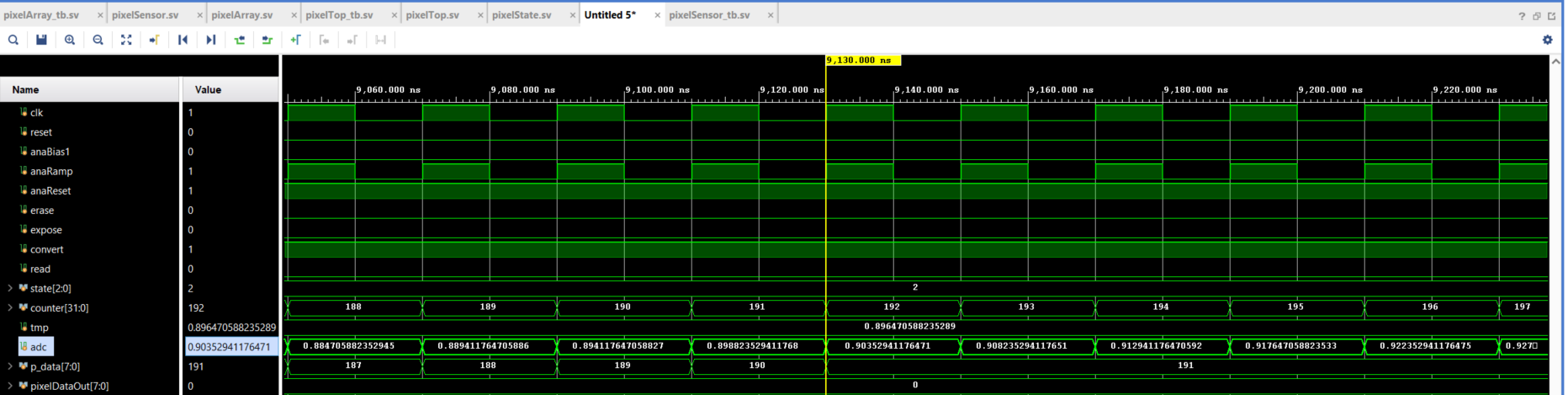
**// READ=1 -> pixel lái bus b?ng p\_data; READ=0 -> nh? bus**

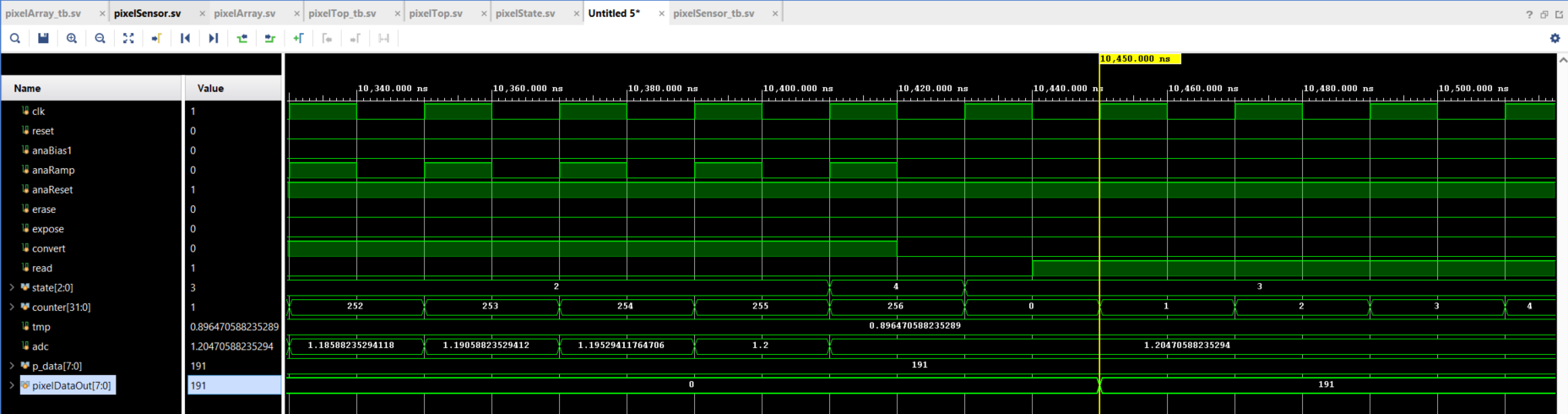
**assign DATA = READ ? p\_data : 8'bZ;**

**endmodule**

- Simulation:







5/ PIXEL\_ARRAY 2x2

* SystemVerilog code:

**module pixelArray(**

**input logic VBN1,**

**input logic RAMP,**

**input logic RESET,**

**input logic ERASE,**

**input logic EXPOSE,**

**input [3:0] READ,**

**inout [(N\*8-1):0] DATA**

**);**

**parameter real dv\_pixel = 0.5; //Set the expected**

**parameter integer N = 2;**

**genvar i;**

**genvar j;**

**generate**

**for(j=0; j<N; j++) begin**

**for(i=0; i<N; i++) begin**

**PIXEL\_SENSOR #(.dv\_pixel(dv\_pixel)) ps1(VBN1,**

**RAMP,**

**RESET,**

**ERASE,**

**EXPOSE,**

**READ[i+(j\*N)],**

**DATA[(1+i)\*8-1:i\*8]);**

**end**

**end**

**endgenerate**

**endmodule**

- Simulation:

