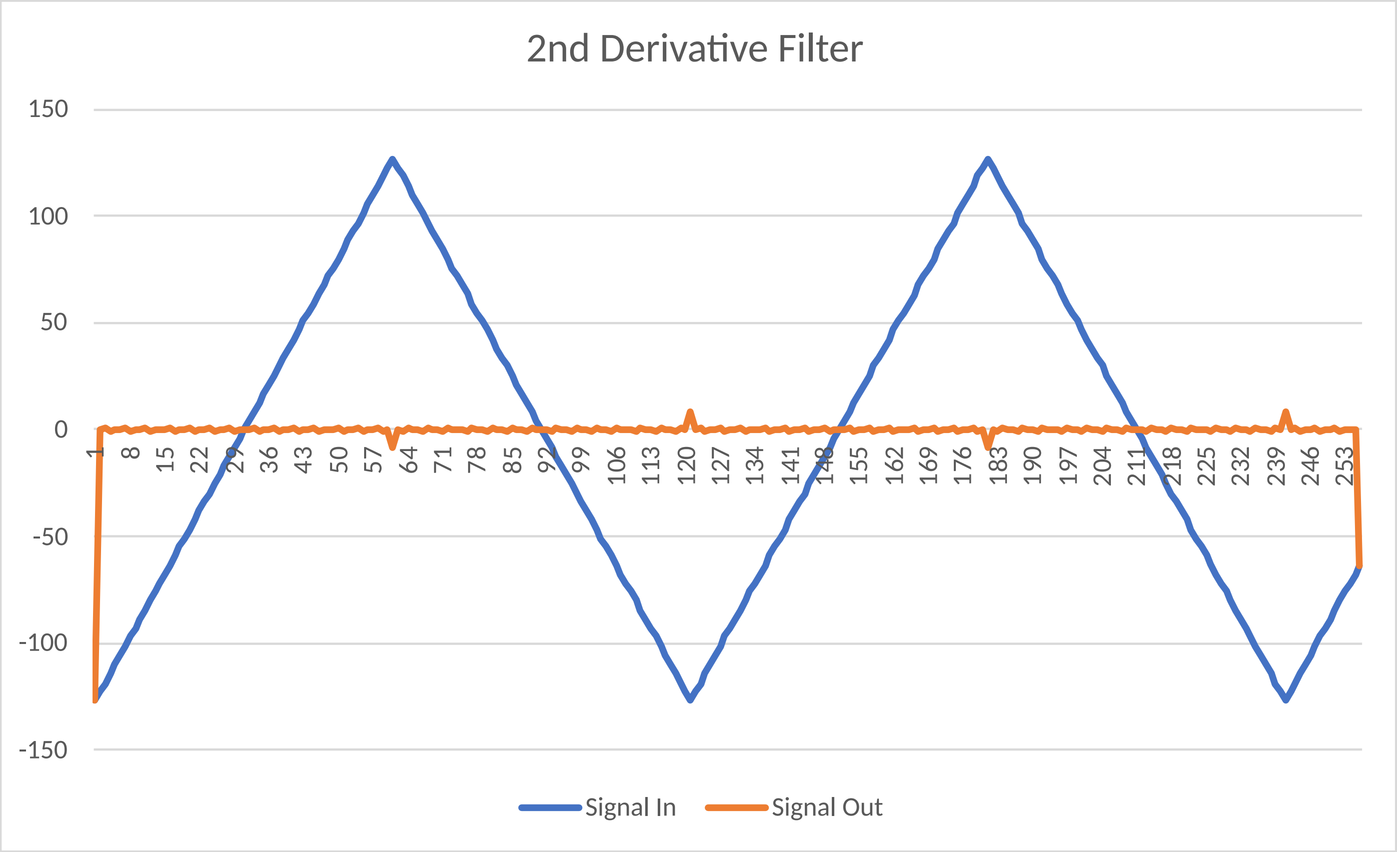
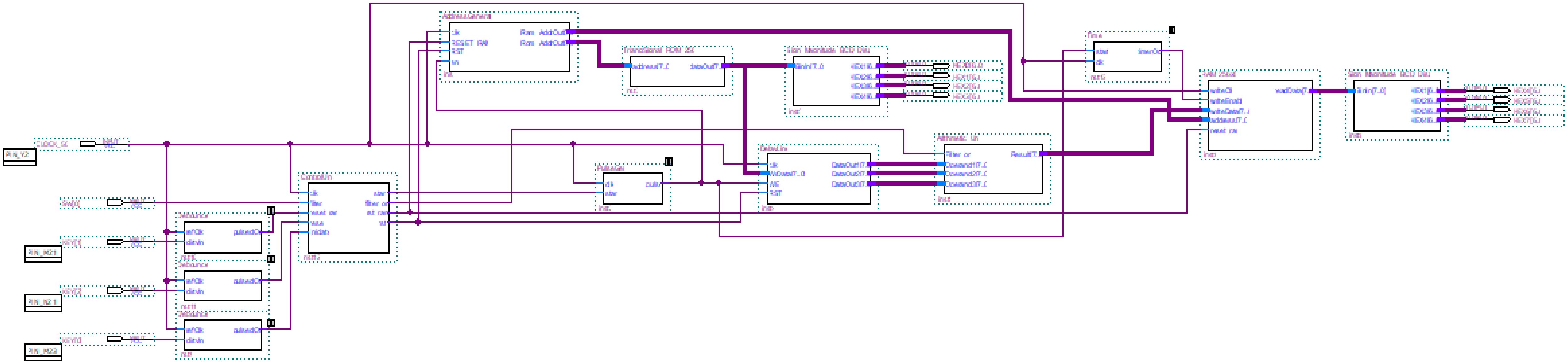
SECOND DERIVATIVE FILTER



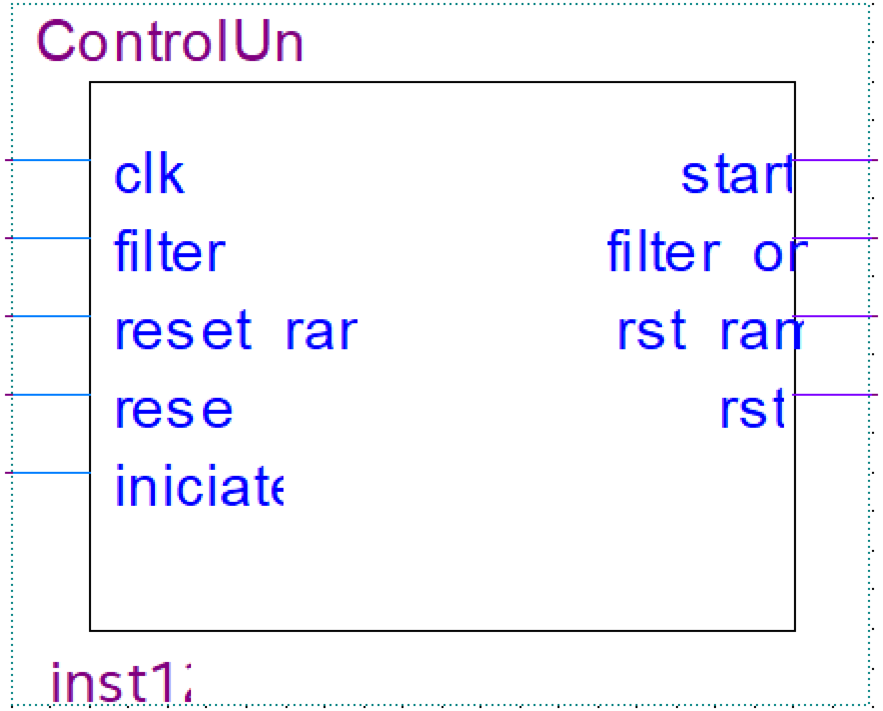
CIRCUIT STRUCTURE [DIAGRAM]



THE CIRCUIT HAS 13 PARTS

- 1 Pulse generator.
- 1 Address generator.
- 1 Rom.
- 1 Ram.
- 1 Delay line.
- 1 Timer.
- 3 Debouncers.
- 2 Binary converters for three 7 segment displays.

CONTROL UNIT



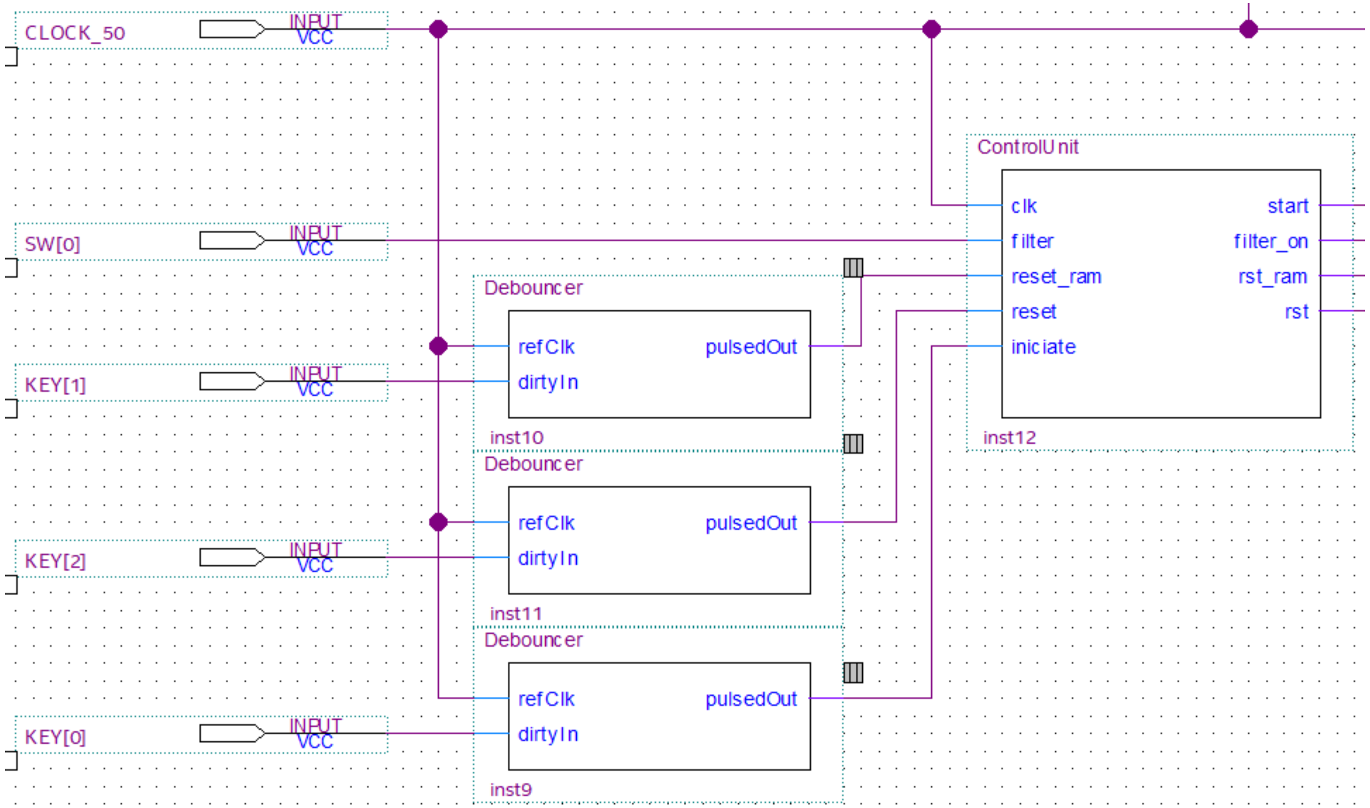
5 INPUTS:

- **Clk :** Controlled by CLOCK_50;
- **Filter :** Controlled by SW[0];
- **Reset_ram :** Controlled by KEY[1]
- **Reset :** Controlled by KEY[2];
- **Iniciate :** Controlled by KEY[0];

4 OUTPUTS:

- **Start :** output that will start the count;
- **Filter_on :** output that will start the filtering;
- **Rst_ram :** output that will clean the RAM and stop the count;
- **Rst :** output that resets the whole circuit;

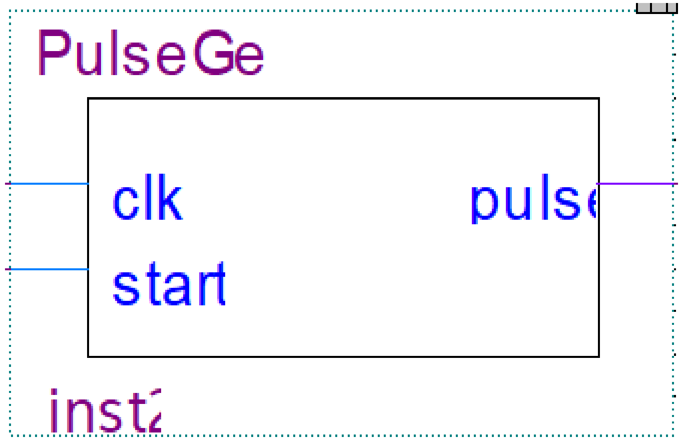
DEBOUNCER



<u>Three DEBOUNCERES are needed!</u>

→ 1 for KEY[0];
→ 1 for KEY[1];
→ 1 for KEY[2];

PULSE GENERATOR



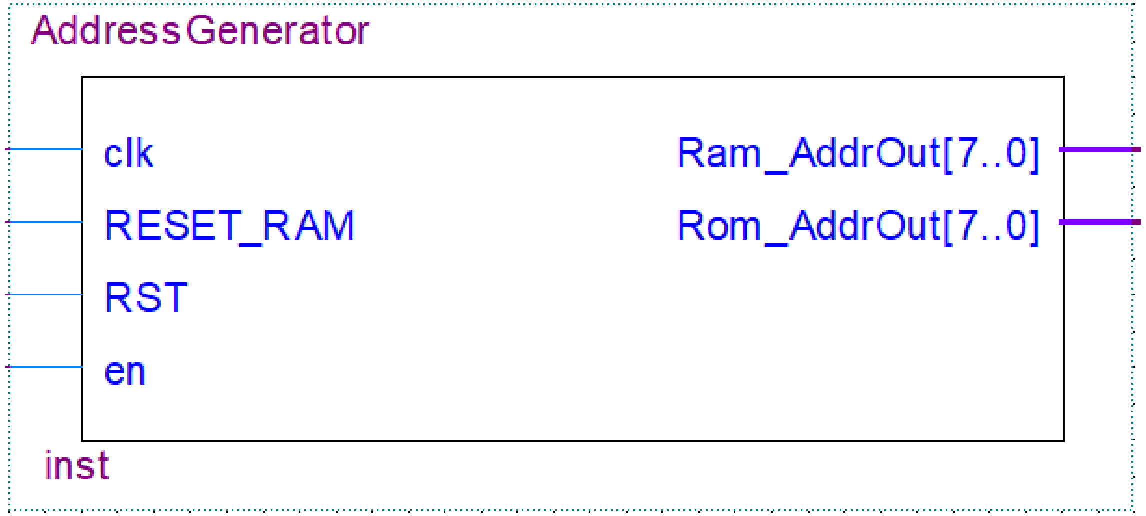
2 INPUTS:

- **Clk :** Controlled by CLOCK_50;
- **Start :** Controlled by the initiation of the CONTROL UNIT;

1 OUTPUT:

- **Pulse :** Generated Pulse every 0.5 seconds;

ADDRESS GENERATOR



4 INPUTS:

- **Clk :** Controlled by CLOCK_50;
- **RESET_RAM :** Controlled by rst_ram from CONTROL UNIT;
- **RST :** Controlled by rst from CONTROL UNIT;
- **En :** Controlled by the pulse from PULSE GENERATOR;

2 OUTPUTS:

- **Ram_AddrOut :** RAM address;
- **Rom_AddrOut :** ROM address;

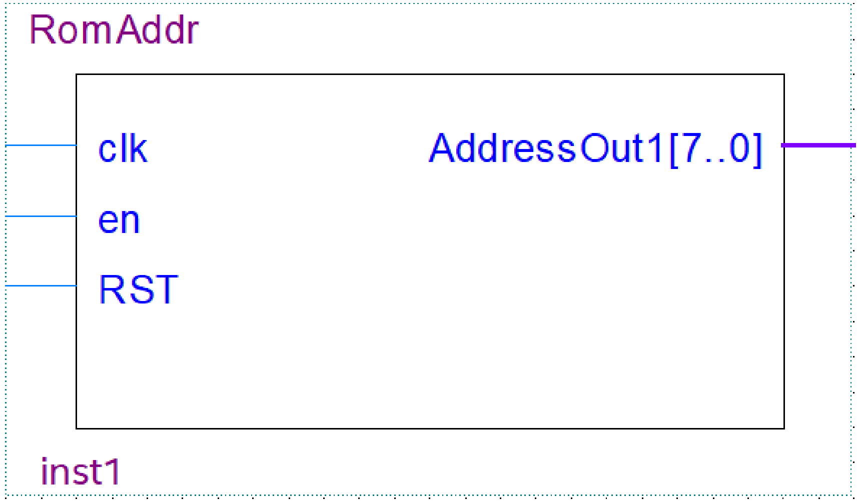
CAN BE DEVIDED INTO 2 PARTS!

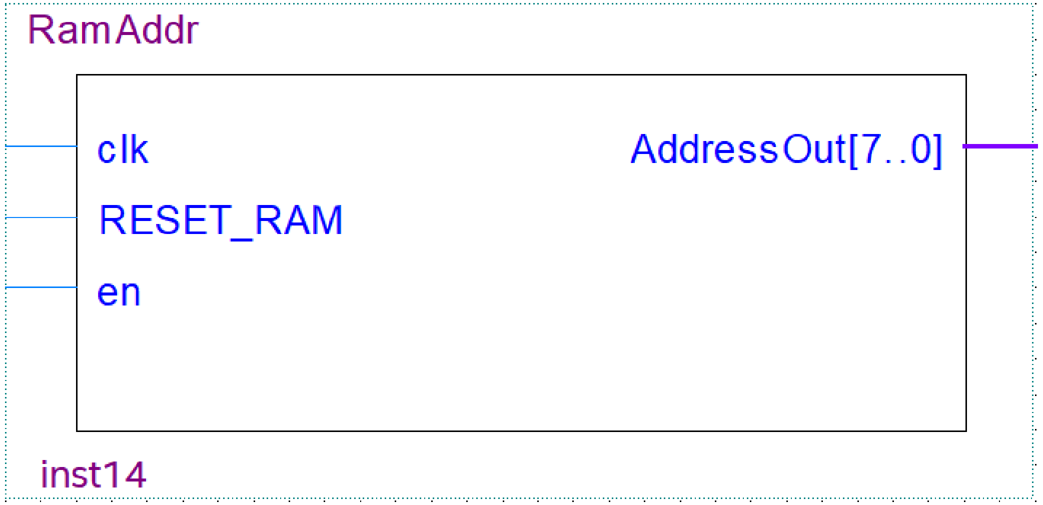
1) Rom address generator:

- Only has a global reset.

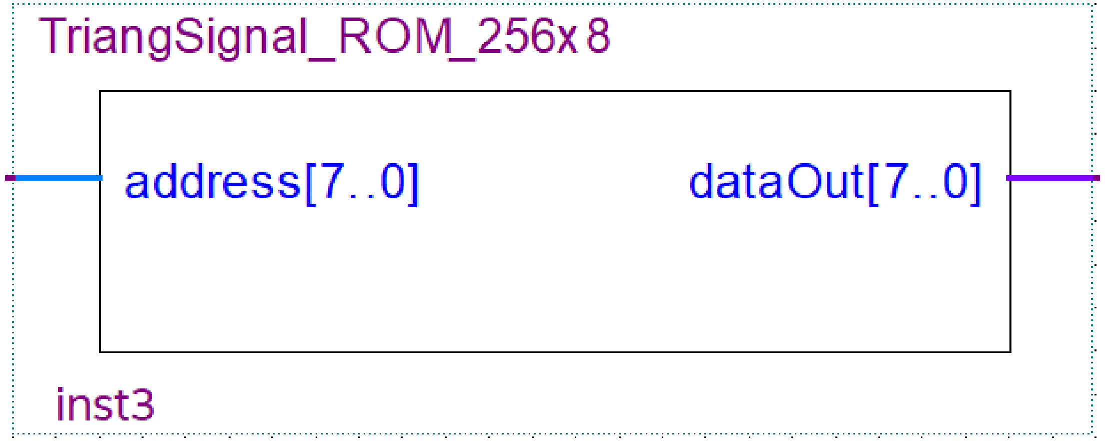
2) Ram address generator:

- Has both a global reset and a ram only reset.





256X8 ROM



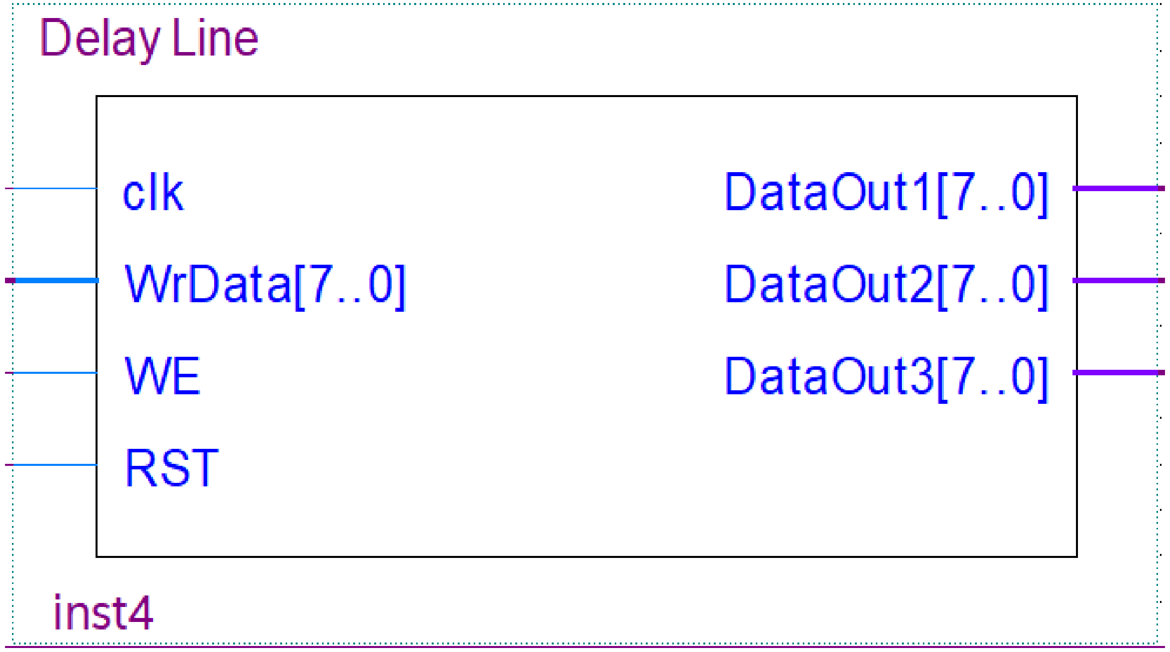
1 INPUT:

- **Address :** Controlled by ADDRESS GENERATOR;

1 OUTPUT:

- **DataOut :** Value from ROM for each address (predefined);

DELAY LINE



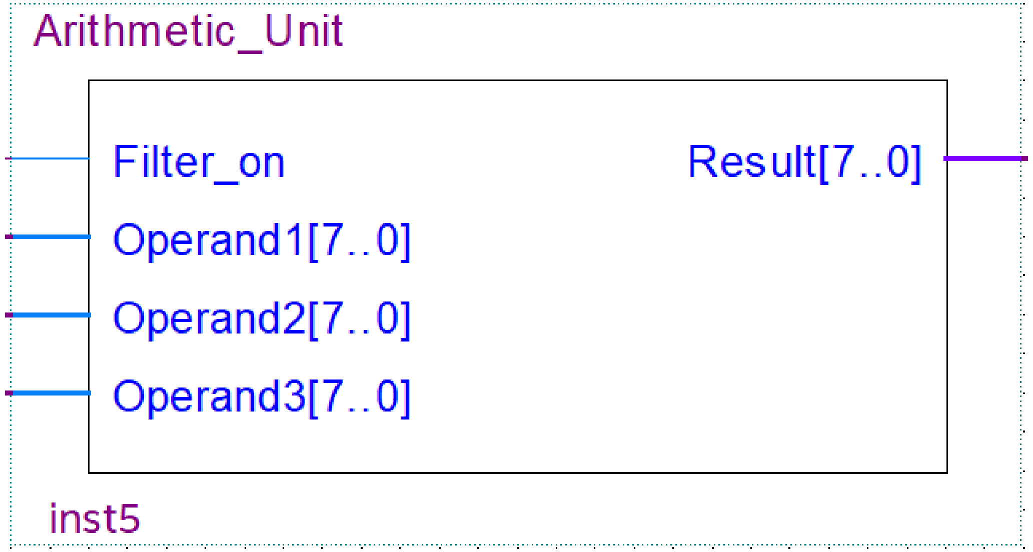
4 INPUTS:

- **Clk :** Controlled by CLOCK_50;
- **WrData :** ROM values;
- **WE :** Activated by the PULSE GENERATOR;
- **RST :** Global reset, controlled by CONTROL UNIT

3 OUTPUTS:

- **DataOut1 :** “k+1” value;
- **DataOut2 :** “k” value;
- **DataOut3 :** “k-1” value;

ARITHMETIC UNIT



Filter operation

$$y_k = x_{k+1} - 2x_k + x_{k-1} \quad k = 1, \dots 254,$$
$$y_0 = x_0, y_{255} = x_{255}$$

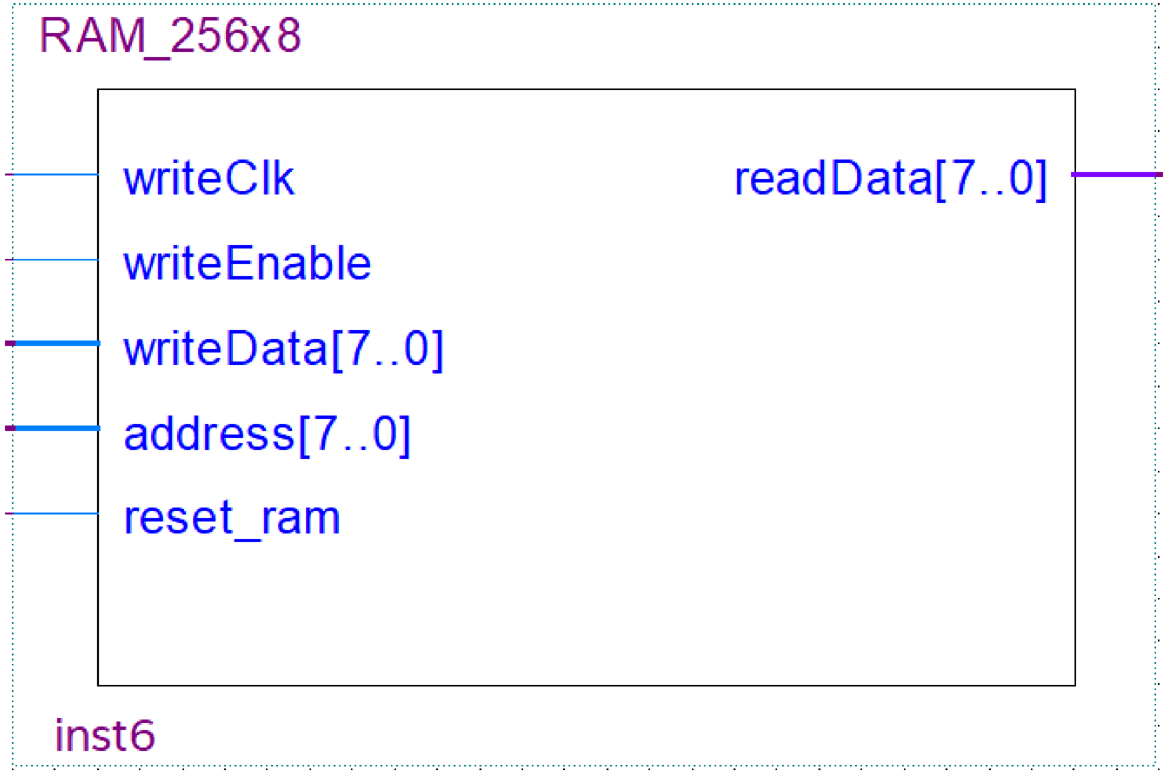
4 INPUTS:

- **Filter_on :** Controlled by CONTROL UNIT;
- **Operand1 :** “K+1” from DELAY LINE;
- **Operand2 :** “K” from DELAY LINE;
- **Operand3 :** “K-1” from DELAY LINE;

1 OUTPUT:

- **Result :** Filter operation result.

256X8 RAM



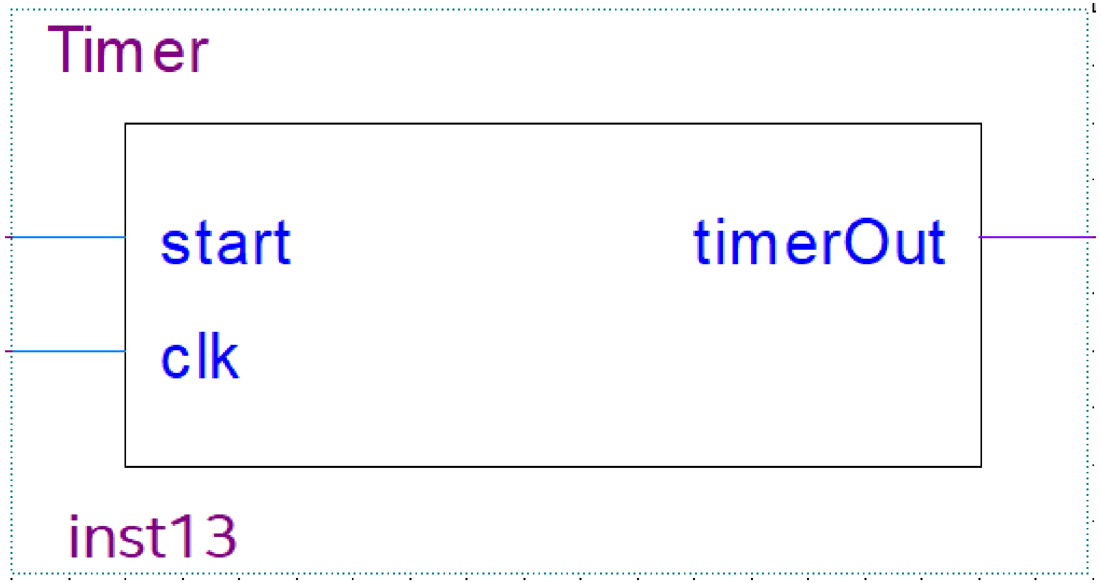
5 INPUTS:

- **WriteClk :** Controlled by CLOCK_50;
- **WriteEnable :** Activated by TIMER;
- **WriteData :** Result from ARITHMETIC UNIT;
- **Address :** Controlled by ADDRESS GENERATOR;
- **Reset_ram :** Controlled by rst_ram from CONTROL UNIT;

1 OUTPUT:

- **ReadData :** Value of RAM in the current address;

TIMER



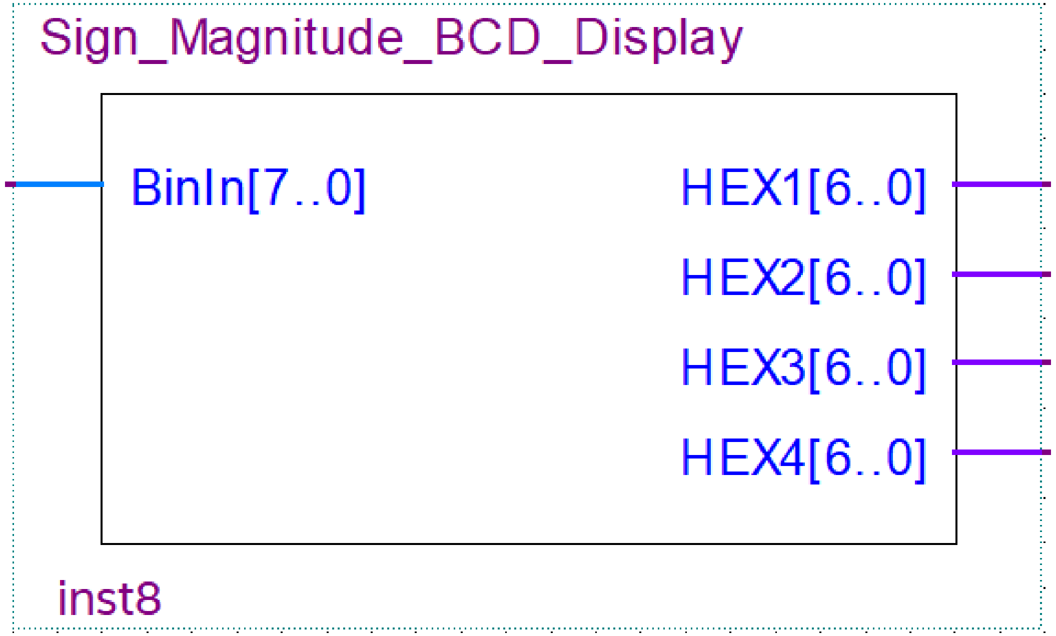
2 INPUTS:

- **Clk :** Controlled by CLOCK_50;
- **Start :** Activated by the pulse from PULSE GENERATOR;

1 OUTPUT:

- **TimerOut :** Output sent after the count;

SIGN MAGNITUDE BCD DISPLAY



1 INPUTS:

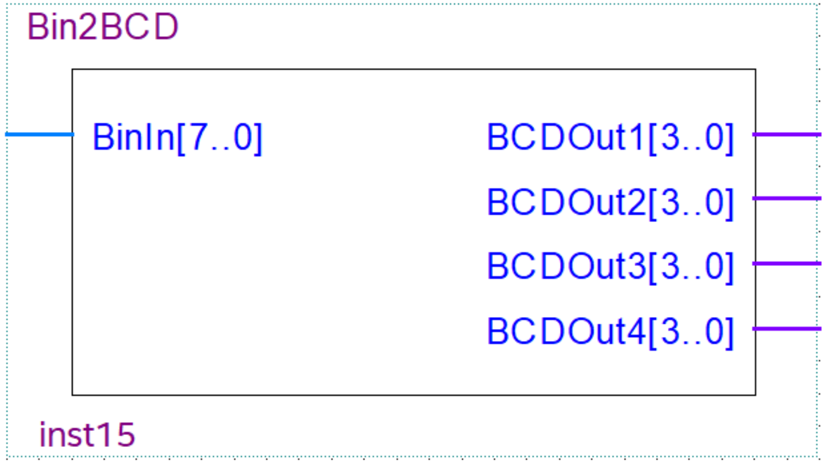
- **BinIn :** Binary value to be displayed;

4 OUTPUT:

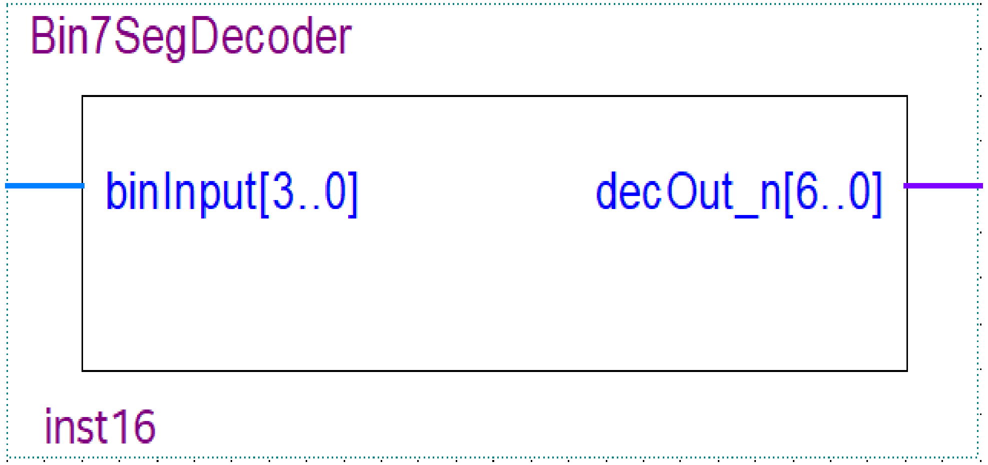
- **HEX1 :** Value sent to the first display;
- **HEX2 :** Value sent to the second display;
- **HEX3 :** Value sent to the third display;
- **HEX4 :** Value sent to the fourth display;

CAN ALSO BE DIVIDED INTO 2 PARTS!

1) 8 bit binary number to 4 BCD numbers of 4 bits each converter.



2) 4 bit BCD number to 7 segment display converter.



PUTTING EVERY PIECE TOGETHER…

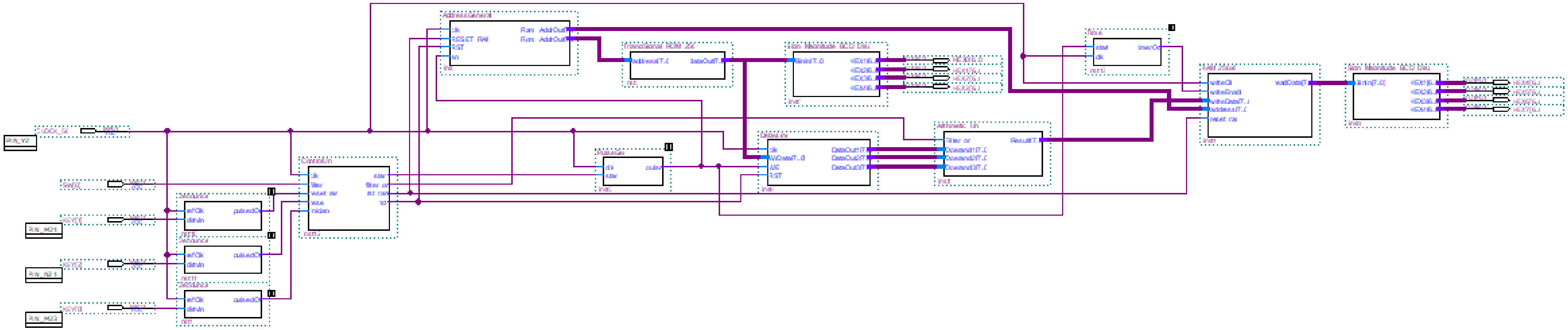
4 INPUTS:

- KEY[0] : Start;
- KEY[1] : Reset Ram;
- KEY[2] : Reset;
- SW[0] : Filter On;

8 OUTPUTS:

- HEX0 to HEX3: ROM values;
- HEX4 to HEX7: RAM values;

And the following circuit:



PIN_Y2
CLOCK_50
SW[0]
KEY[1]
PIN_M21
KEY[2]
PIN_N21
KEY[0]
PIN_M23
Debounce
Debounce
Debounce
ControlUn
PulseGen
Timer
AddressGenerat
TransitionalROM 256
Sign_Magnitude_BCD Dis
DelayLine
Arithmetic_Un
RAM 256x8
Sign_Magnitude_BCD Dis
HEX0[6..0]
HEX1[6..0]
HEX2[6..0]
HEX3[6..0]
HEX4[6..0]
HEX5[6..0]
HEX6[6..0]
HEX7[6..0]