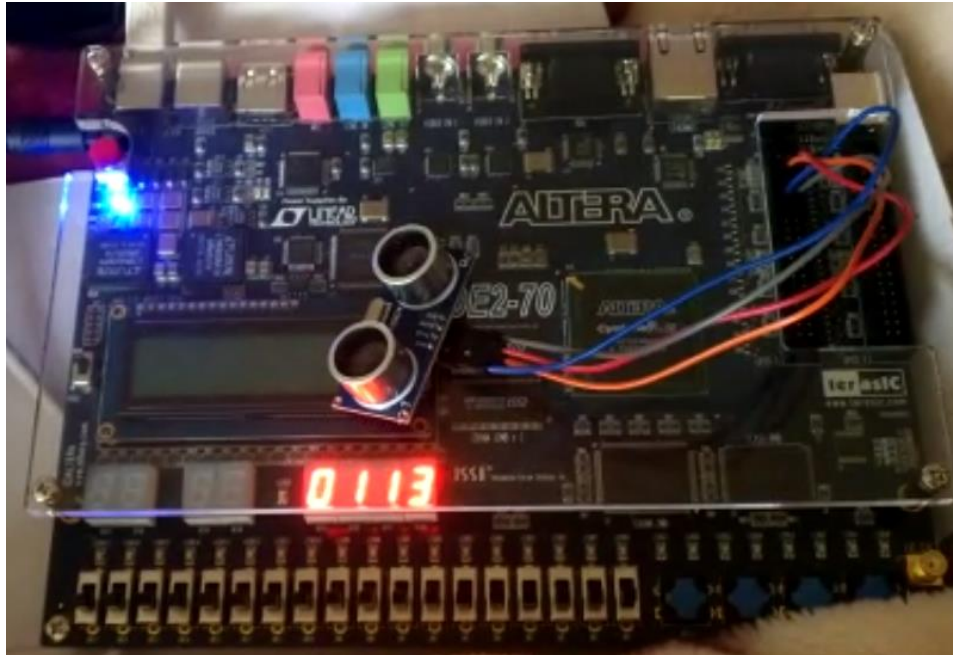


# Range sensor using FPGA



**By: Yuvraj Patel**

**ECE 641**

**Prof. Ali Akansu**

# Introduction

## What is Range Sensor?

- Range sensor captures the three-dimensional (3-D) representation of the object, mainly used for measuring depth of the nearest surface.

## Working Mechanism

- Ultrasonic range sensor uses SONAR for detecting objects and the distance. A similar technique is used by ships and bats to detect objects underwater and in the absence of light.

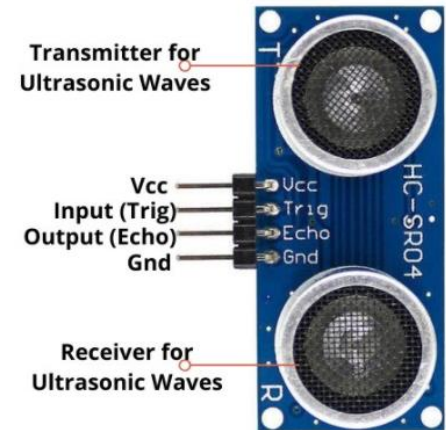
## Uses and implementation

- Autonomous vehicles
- Robotic Obstacle detection
- Manufacturing industry

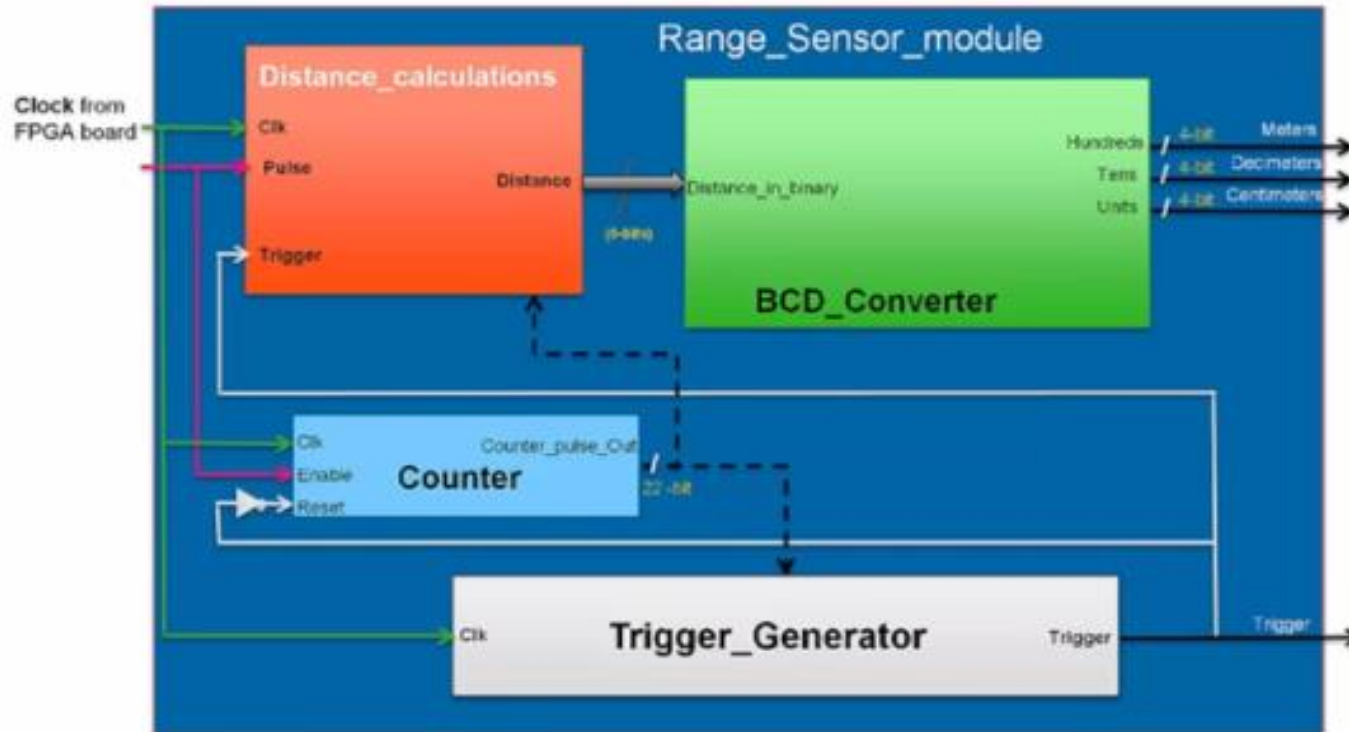
# Introduction

## Materials used for project

- HC-SR04 Ultrasonic sensor
  - This economical sensor provides 2cm to 400cm of non-contact measurement functionality with a ranging accuracy that can reach up to 3mm.
  - A 10 $\mu$ S TTL trigger pulse needs to be instantiate for time calculation of the received signal.
  - Operating voltage is DC 5V and operating current is 15mA.
- Altera DE2-70 FPGA board and jumper cables



# Block diagram of the VHDL code structure



<https://www.pantechsolutions.net/fpga-implementation-of-distance-measurement-with-ultrasonic-sensor>

# Block diagram of the VHDL code structure

```
entity binary_counter is
  generic
    (COUNT : Positive := 10);
  port
    (
      clk      : in std_logic;
      enable   : in std_logic;
      reset    : in std_logic;
      q        : out std_logic_vector (COUNT -1 downto 0)
    );
end entity;
```

```
architecture rtl of binary_counter is
  signal cnt      : std_logic_vector (COUNT -1 downto 0);
begin
  process (clk,reset)
  begin
    if reset = '0' then -- Reset signal
      cnt <= (others=>'0');
    elsif clk'event and clk='1' then
      if enable = '1' then -- counter incremented on enabled
        cnt <= cnt + 1;
      end if;
    end if;
  end process;
  q <= cnt; -- Output
end rtl;
```

- The main counter is initialized in the first VHDL file.

# Block diagram of the VHDL code structure

```
entity Distance_Calculation is
  port
  (
    clk      : in std_logic;
    calculation_reset : in std_logic;
    pulse     : in std_logic;
    distance  : out std_logic_vector (8 downto 0)
  );
end entity;
```

```
signal pulse_width : std_logic_vector (21 downto 0);

begin
  CounterPulse : binary_counter generic map (22) port map (clk, pulse, not calculation_reset, pulse_width);

  Distance_Calculation: process(pulse)
    variable result: integer;
    variable multiplier: std_logic_vector (23 downto 0);
    begin
      if pulse = '0' then
        multiplier := pulse_width * "11"; --multiply the pulse width with 3 and divide it by 58 to calculate
        result := to_integer (unsigned (multiplier (23 downto 13))); -- multiply by 3 and shift the expres
        if (result = 458) then
          distance <= "11111111";
        else
          distance <= std_logic_vector (to_unsigned (result,9));
        end if ;
      end if ;
    end if ;
  end process Distance_Calculation;
end behavioral;
```

- Distance is calculated in centimeters, by doing calculation on the pulse width.

# Block diagram of the VHDL code structure

```
entity TriggerGenerator is
    port
    (
        clk      : in std_logic;
        trigger   : out std_logic
    );
end entity;
```

```
signal reset_counter : std_logic ;
signal outputcounter : std_logic_vector (23 downto 0);
begin
    trigg : binary_counter generic map (24) port map (clk=> clk,enable => '1', reset =>reset_counter,q=>outputcounter);
    process (clk)
        constant ms250 :std_logic_vector (23 downto 0):= "101111101011110000100000";
        constant ms250add100us :std_logic_vector (23 downto 0):= "101111101100111110101000";
        begin
            if (outputcounter > ms250 and outputcounter < ms250add100us) then
                trigger<= '1';
            else
                trigger<= '0';
            end if ;
            if (outputcounter =ms250add100us or outputcounter ="XXXXXXXXXXXXXXXXXXXX" ) then
                reset_counter <= '0';
            else
                reset_counter <='1';
            end if;
        end process;
end behavioral;
```

- Distance is calculated in centimeters, by doing calculation on the pulse width.

# Block diagram of the VHDL code structure

```
entity BCDconverter is
  port
  (
    DistanceInput  : in std_logic_vector(8 downto 0);
    hundreds       : out std_logic_vector(3 downto 0);
    tens           : out std_logic_vector(3 downto 0);
    unit           : out std_logic_vector(3 downto 0)
  );
end entity;
```

```
architecture behavioral of BCDconverter is
begin
  process(DistanceInput)
    variable i : integer := 0;
    variable bcd: std_logic_vector (20 downto 0);
  begin
    bcd := (others => '0');
    bcd(8 downto 0) := DistanceInput;
    --double-dabble algorithm to convert the binary number
    for i in 0 to 8 loop
      bcd(19 downto 0) := bcd(18 downto 0) & '0';
      if (i<8 and bcd(12 downto 9) > "0100") then
        bcd(12 downto 9) := bcd(12 downto 9) + "0011";
      end if;
      if (i<8 and bcd(16 downto 13) > "0100") then
        bcd(16 downto 13) := bcd(16 downto 13) + "0011";
      end if;
      if (i<8 and bcd(20 downto 17) > "0100") then
        bcd(20 downto 17) := bcd(20 downto 17) + "0011";
      end if;
    end loop;
    hundreds <= bcd(20 downto 17);
    tens <= bcd(16 downto 13);
    unit <= bcd(12 downto 9);
  end process;
end behavioral;
```

- Binary value of the distance is converted to BCD format for the seven-segment display module.



# Block diagram of the VHDL code structure

```
entity RangeSensor is
  port
  (
    fpgaclk      : in std_logic;
    pulse        : in std_logic;
    triggerOut   : out std_logic;
    meters       : out std_logic_vector(3 downto 0);
    decimeter    : out std_logic_vector(3 downto 0);
    centimeter   : out std_logic_vector(3 downto 0)
  );
end entity;
```

```
signal distanceout: std_logic_vector (8 downto 0);
signal triggout : std_logic;

begin

trigger_gen : TriggerGenerator port map (clk =>fpgaclk,trigger=>triggout);

pulsewidth : Distance_Calcualtion port map (clk =>fpgaclk,calculation_reset => triggout,pulse=>pulse,distance=>distanceout);

BCDCov : BCDconverter port map (DistanceInput=>distanceout,hundreds=>meters,tens=>decimeter,unit=>centimeter);

triggerOut<= triggout;

end behavioral;
```

- Port mapping is executed in the range sensor VHDL file to connect different components.

# Block diagram of the VHDL code structure

```
entity UltraSonic is
  port
  (
    clk           : in std_logic;
    pulse_pin     : in std_logic;
    trigger_pin   : out std_logic;
    op0,op1,op2,op3 : out std_logic_vector(6 downto 0)
  );
end entity;
```

```
Signal Ai: std_logic_vector (3 downto 0);
Signal Bi: std_logic_vector (3 downto 0);
Signal Ci: std_logic_vector (3 downto 0);
Signal Di: std_logic_vector (3 downto 0);

Signal sensor_meter: std_logic_vector (3 downto 0);
Signal sensor_centimeter: std_logic_vector (3 downto 0);
Signal sensor_decimeter: std_logic_vector (3 downto 0);

begin

uu3: RangeSensor port map (fpgaclk=> clk,  pulse=> pulse_pin,triggerOut=> trigger_pin,meters=> sensor_meter,decimeter => sensor_decimeter,centim

  Ai <= sensor_centimeter;
  Bi <= sensor_decimeter;
  Ci <= sensor_meter;
  Di <= "0000";

  z0: seg7 port map(Ai,op0);
  z1: seg7 port map(Bi,op1);
  z2: seg7 port map(Ci,op2);
  z3: seg7 port map(Di,op3);

end nt1;
```

- The seven segment driver and the range sensor are connected here.

# Output video, pin planner and LE of the VHDL project

Quartus II - Y:\Desktop\Fall 2020\ECE 641\Final\ultrasonic project\UltraSonic-20201216T070051Z-001\UltraSonic\UltraSonic - UltraSonic - [Pin Planner]

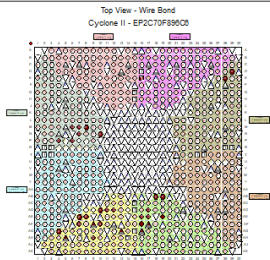
File Edit View Processing Tools Window

Groups

Named

Node Name	Direction
op0[6..0]	Output Group
op1[6..0]	Output Group
op2[6..0]	Output Group
op3[6..0]	Output Group
<new node>>	Output Group

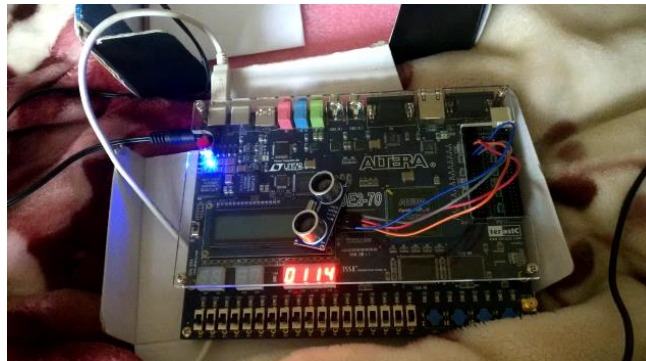
Top View - Wire Bond  
Cyclone II - EP2C70F896C6



Pin Planner Table:

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved
op2[5]	Output	PIN_AB19	7	B7_N1	3.3-V LVTTL (default)	
op2[4]	Output	PIN_AB18	7	B7_N1	3.3-V LVTTL (default)	
op2[3]	Output	PIN_AG4	8	B8_N3	3.3-V LVTTL (default)	
op2[2]	Output	PIN_AH5	8	B8_N3	3.3-V LVTTL (default)	
op2[1]	Output	PIN_AF7	8	B8_N3	3.3-V LVTTL (default)	
op2[0]	Output	PIN_AE7	8	B8_N3	3.3-V LVTTL (default)	
op3[6]	Output	PIN_M6	2	B2_N2	3.3-V LVTTL (default)	
op3[5]	Output	PIN_M7	2	B2_N2	3.3-V LVTTL (default)	
op3[4]	Output	PIN_M8	2	B2_N1	3.3-V LVTTL (default)	
op3[3]	Output	PIN_N7	2	B2_N2	3.3-V LVTTL (default)	
op3[2]	Output	PIN_N10	2	B2_N2	3.3-V LVTTL (default)	
op3[1]	Output	PIN_P4	2	B2_N3	3.3-V LVTTL (default)	
op3[0]	Output	PIN_P6	2	B2_N3	3.3-V LVTTL (default)	
pulse_pin	Input	PIN_D29	5	B5_N0	3.3-V LVTTL (default)	
trigger_pin	Output	PIN_E28	5	B5_N0	3.3-V LVTTL (default)	
<new node>>						

Flow Status	Successful - Thu Dec 17 05:26:34 2020
Quartus II Version	9.1 Build 222 10/21/2009 SJ Web Edition
Revision Name	UltraSonic
Top-level Entity Name	UltraSonic
Family	Cyclone II
Device	EP2C70F896C6
Timing Models	Final
Met timing requirements	N/A
Total logic elements	150
Total combinational functions	150
Dedicated logic registers	55
Total registers	55
Total pins	31
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0



# Future Goals and discussion

- Implementation Machine learning algorithm to determine what type of object is Infront of the sensor.
- Mounting the range sensor with a servo motor, to provide a 360° 3-D representation of the object.
- Modifying the VHDL code for using the sensor with an autonomous car, for generating a map of surrounding area.

Thank you!!