Implementation of a solar pv module and tracking the maximum power point using perturb and disturb algorithm in a FPGA using verilog .

Theory :

As the sun moves through the sky from east to west, solar radiations accomplished by the solar panel are continuously varied resulting the degraded performance of the solar panel; the PV cell starts operating below its maximum power. Similarly, at higher temperatures performance of the solar module is degraded .

Varying insolation changes all the parameters (PMAX, VMAX, IMAX, VOC, ISC) of the solar cell

Solar efficiency is a problem and cannot be tested . This can be improved by solar trackers and maximum power point trackers .

Solar trackers are used to orienting the PV modules towards the sun to maximize the solar irradiance coupling.We are not implementing a solar tracker . Solar tracker makes sure the maximum coupling of the irradiance with the solar panel by keeping the panel always normal to the sun.

maximum power point is always prone to two important factors: solar irradiance and the temperature .

MPPT is used to let the photovoltaic cell function at its maximum power point by properly adjusting the duty cycle of the converter .

Effect of Irradiation and Temperature :

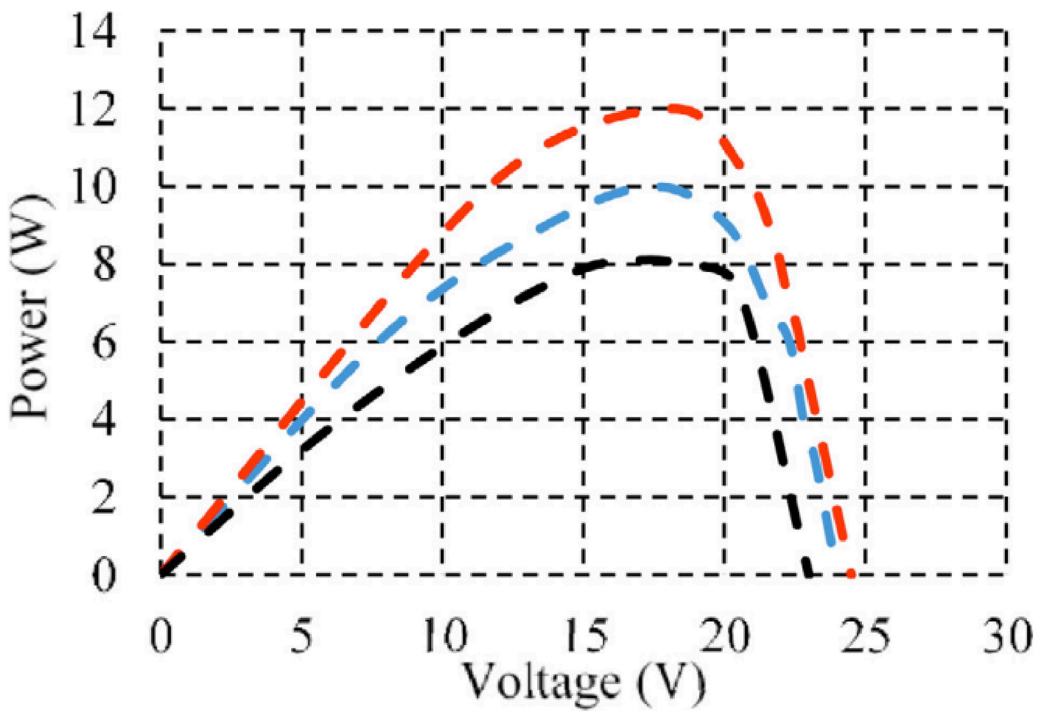


Fig. 4. PV curves under varying solar irradiance.

Present algorithms :

1)Incremental Conductance (INC): it compares the slope of the power curve and determines whether to increment or decrement the duty ratio (Shahid et al., 2018),

2)Perturb & Observe: it observes the voltage level and perturbs the voltage till it grasps the maximum power point,

3)Artificial Neural Network (ANN),

4)Constant voltage (CV)

5)Fuzzy Logic (FL)

6) Particle Swarm Optimization (PSO) are some of the MPPT algorithms

SOLAR PV MODELLING:

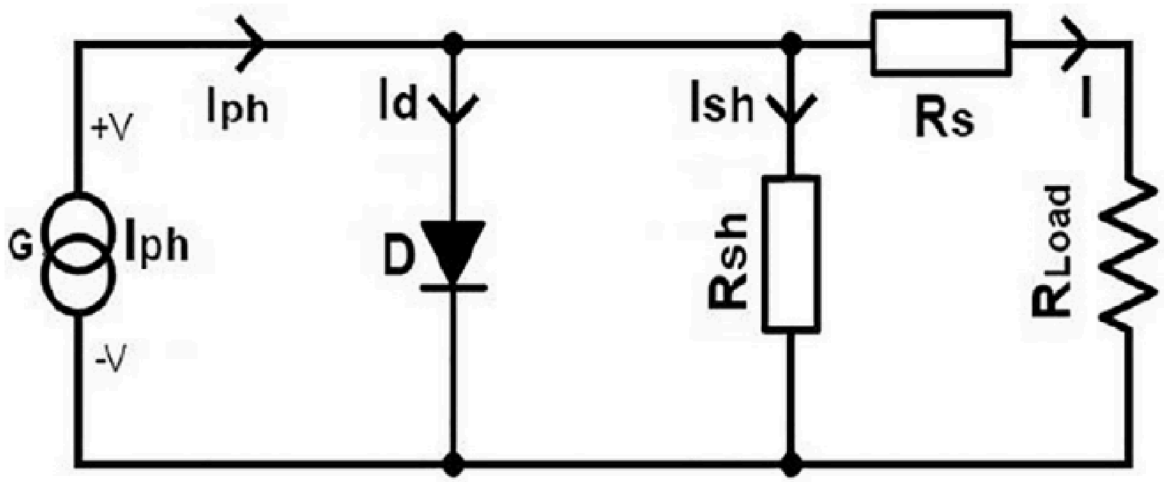


Fig. 2. Single diode model of solar PV cell.

The solar photovoltaic cell is fabricated of the semiconductor materials with rear side positive and sun facing side as negative. Whenever the sunlight falls on the PV materials, it generates electrons flowing in the external circuit known as photocurrent or short circuit current and is calculated by equation .

$$I_{sc} = \frac{G}{1000}\left[I_{scr} + K_i(T_c - T_r)\right]$$

where G solar irradiance, Isc photocurrent, Iscr reverse saturation current, Ki temperature coefficient, Tc cell temperature, Tr reference temperature .

In modeling of the PV cell, it is indicated by a current source .

There appears a voltage at the output terminal if it is open circuited and called open circuit voltage calculated by equation .

$$V_{oc} = \ln\left(\frac{I_{sc}}{I_o} + 1\right)\left(\frac{nkT_c}{q}\right)$$

where Voc open circuit voltage, Isc photo current, I0 saturation current, n ideality factor, k Boltzmann constant, Tc cell temperature, q Electron charge .

This voltage causes a current through the P-N junction just like a diode. This diode and the current source (Iph) are put in parallel .

Losses :

As the photogenerated current starts flowing, some of the electron-hole recombination occurs that reduces the originally generated electrons; this loss of current is presented by a shunt resistance (Rsh). Series resistance (Rs) indicates the resistance faced by the current as it flows through the bulk material, external metal contacts and to the load. Manufacturers always manage to keep the effects of both these resistances as low as possible to improve the working of the PV module .

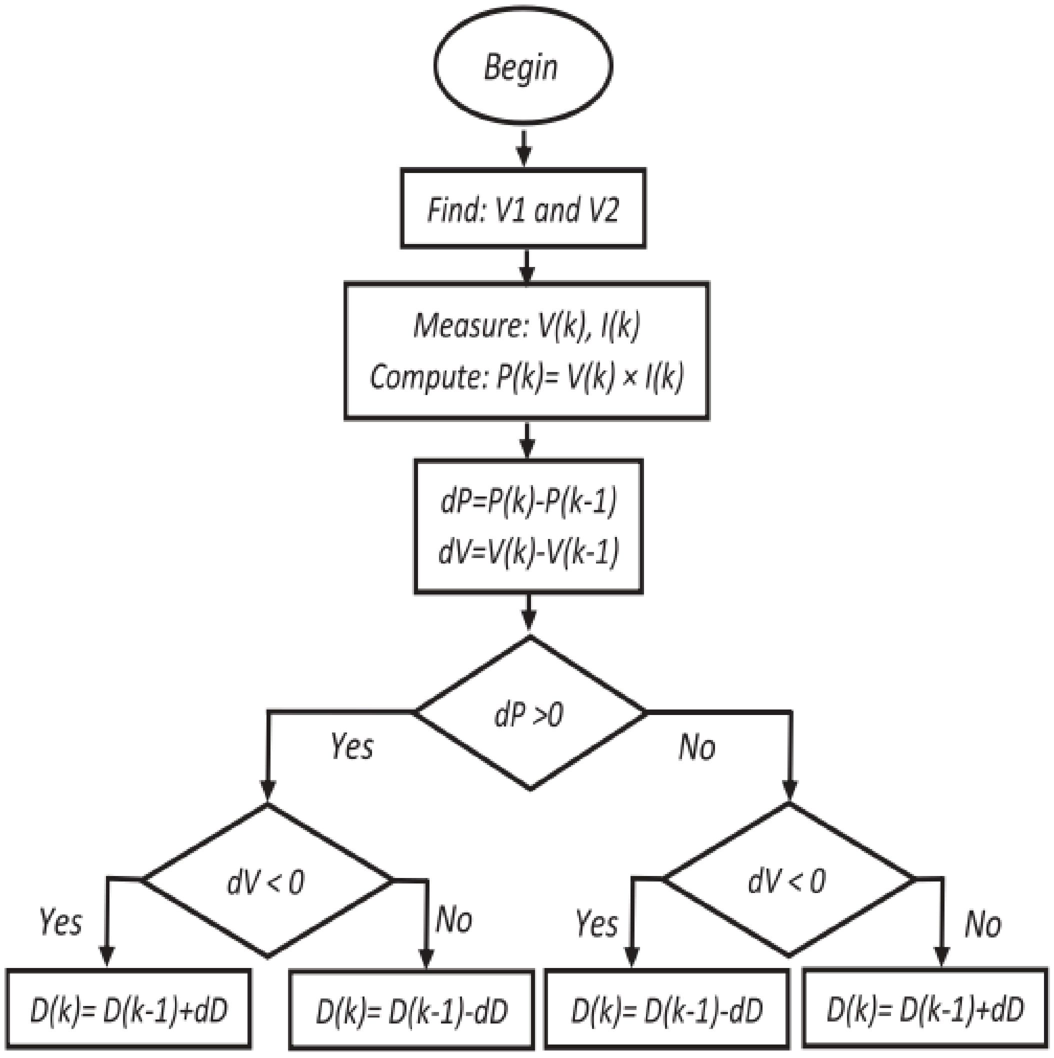
The relation between the voltage and current of the solar cell.

$$I = I_s\left[exp\left(\frac{qV}{kT}\right) - 1\right] - I_{ph}$$

where Is saturation current, q Electron charge, V voltage across the diode, k Boltzman constant, T absolute temperature (K), Iph light generated current .

P&O algorithm:

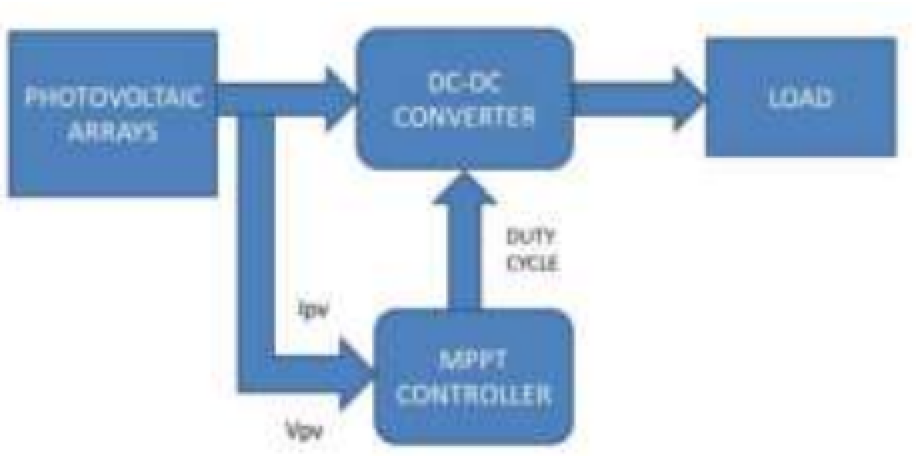
This is a continuous process of observation and perturbation till the operating point converges at the MPP. The algorithm compares the power and voltages of time (K) with the sample at a time (K-1) and predicts the time to approach to MPP. A small voltage perturbation changes the power of the solar panel if the power alteration is positive, voltage perturbation is continued in the same track. But if delta power is negative, it indicates that the MPP is far away and the perturbation is decreased to reach the MPP.



Limitations: response time problem and steady state oscillations.

According to research it has been found that VMPP is about 76% of the open circuit voltage (VMPP = 76% of VOC).

This is the block diagram for the pv system :



Proposed Solution:

First we have modelled the PV system which takes in the input as temperature and irradiance and gives the power and efficiency of working as the output signals.

This is just used to visualize the PV system in Verilog and not further used anymore.

We calculated power using the formula :

$PPV(t) = Ppeak(G/Gstandard) - \alpha T\ (Tc - Tstandard)$.
$Efficiency = Ppv/Ppeak * 100$.

Verilog code for solar pv:

```
`timescale 1ns / 1ps

module solar_pv(
   input wire [5:0] temp,          // Temperature input (6-bit)
   input wire [9:0] irradiance,     // Irradiance input (10-bit)
   output reg [8:0] power,         // Power output (9-bit)
   output reg [6:0] efficiency      // Efficiency output (7-bit)
   );

   // Parameters
   parameter integer P_PEAK = 300;        // Peak power (300W)
   parameter integer STD_IRRADIANCE = 1000; // Standard irradiance
```

```
    parameter integer TEMP_COEFF = 4;       // Temperature coefficient (scaled)
    parameter integer STD_TEMP = 25;        // Standard temperature

    // Internal variables
    wire signed [17:0] irradiance_effect;   // Signed irradiance contribution
    wire signed [17:0] temp_adjust;         // Signed temperature adjustment
    wire signed [17:0] calculated_power;    // Final signed power calculation

    // Cast inputs to signed for arithmetic
    wire signed [9:0] signed_temp = temp;
    wire signed [17:0] signed_irradiance = irradiance;

    // Calculate irradiance contribution
    assign irradiance_effect = (P_PEAK * signed_irradiance) / STD_IRRADIANCE;

    // Calculate temperature adjustment
    assign temp_adjust = (TEMP_COEFF * (signed_temp - STD_TEMP) * P_PEAK) / 1000;

    // Final signed power calculation
    assign calculated_power = irradiance_effect - temp_adjust;

    always @(*) begin
        // Clip power to prevent negative values
        if (calculated_power < 0)
            power = 9'd0;  // Minimum limit
        else if (calculated_power > 511)
            power = 9'd511; // Maximum limit
        else
            power = calculated_power[8:0];  // Assign valid power

        // Calculate efficiency (scaled to percentage)
        if (power > 0)
            efficiency = (power * 100) / P_PEAK;
        else
            efficiency = 7'd0;
    end

endmodule
```

Next we proceed to construct a maximum power point tracker . This modules takes the current and voltage of the pv module as input and generates a pulse width modulating signal for the duty cycle of the dc dc converter .

Verilog code for mppt:

```
`timescale 1ns / 1ps

module clock_divider(
   input wire clk_in,      // 100 MHz clock input
   input wire reset,       // Reset signal
   output reg clk_out      // 5-second clock output
   );

   reg [31:0] counter;     // Counter for clock division

   // Parameter to calculate the number of clock cycles for a 5-second period
   parameter DIVIDE_BY = 32'd250_000_000; // (100 MHz * 5 seconds) = 500 million cycles

   always @(posedge clk_in or posedge reset) begin
      if (reset) begin
         counter <= 32'd0;
         clk_out <= 1'b0;
      end else begin
         if (counter == DIVIDE_BY - 1) begin
            counter <= 32'd0;
            clk_out <= ~clk_out; // Toggle the clock output
         end else begin
            counter <= counter + 1;
         end
      end
   end

endmodule



module top(
   input wire clk,                // Onboard 100 MHz clock
   input wire reset,              // Reset signal
   input wire [7:0] voltage,      // Voltage input (V)
   input wire [6:0] current,      // Current input (I          // Slow clock (5-second period)
   output wire [7:0] duty_cycle      // Duty cycle output from MPPT
   );

   wire slow_clk;
   // Instantiate the clock divider
   clock_divider #(
      .DIVIDE_BY(32'd500_000_000) // Divide by 100 MHz * 5 seconds = 500 million
   ) clk_div (
      .clk_in(clk),              // 100 MHz clock input
      .reset(reset),             // Reset signal
      .clk_out(slow_clk)         // 5-second clock output
   );

   // Instantiate the MPPT tracker
   mppt_tracker tracker (
      .clk(slow_clk),            // Use the 5-second clock as input
      .reset(reset),             // Reset signal
      .voltage(voltage),         // Voltage input
      .current(current),         // Current input
      .duty_cycle(duty_cycle)    // Duty cycle output
   );

endmodule





module mppt_tracker(
   input wire clk,                 // System clock
   input wire reset,               // Reset signal
   input wire [7:0] voltage,       // Voltage input (V)
   input wire [6:0] current,       // Current input (I)
   output reg [7:0] duty_cycle     // PWM duty cycle output
   );

   // Internal registers
   reg [17:0] power;               // Current power
   reg [17:0] prev_power;          // Previous power
   reg [8:0] prev_voltage;         // Previous voltage
   reg signed [17:0] delta_power;   // Change in power (dP)
   reg signed [8:0] delta_voltage;  // Change in voltage (dV)

   parameter signed DELTA_D = 8'd1; // Step size for duty cycle adjustment
   always @(posedge clk or posedge reset) begin
   if (reset) begin
      // Reset all variables
      power = 18'd0;
      prev_power = 18'd0;
      prev_voltage = 9'd0;
      duty_cycle = 8'd128; // Start with 50% duty cycle
   end
   else begin
      // Calculate current power
      power = voltage * current;

      // Calculate changes in power and voltage
      delta_power = power - prev_power;
      delta_voltage = voltage - prev_voltage;

      // Apply P&O logic
      if (delta_power > 0) begin
          if (delta_voltage < 0) begin
              // Increase duty cycle
              duty_cycle = (duty_cycle + DELTA_D <= 8'd255) ? duty_cycle + DELTA_D : 8'd255;
          end else if (delta_voltage > 0) begin
              // Decrease duty cycle
              duty_cycle = (duty_cycle >= DELTA_D) ? duty_cycle - DELTA_D : 8'd0;
          end
      end else if (delta_power < 0) begin
          if (delta_voltage < 0) begin
              // Decrease duty cycle
              duty_cycle = (duty_cycle >= DELTA_D) ? duty_cycle - DELTA_D : 8'd0;
          end else if (delta_voltage > 0) begin
              // Increase duty cycle
              duty_cycle = (duty_cycle + DELTA_D <= 8'd255) ? duty_cycle + DELTA_D : 8'd255;
          end
      end
      // If delta_power == 0, keep the duty_cycle constant

      // Update previous values
      prev_power = power;
      prev_voltage = voltage;
   end
end
```

```
endmodule
```

CONCLUSION:

Thus we have successfully modelled a solar PV system and then integrated a Maximum Power Point Tracker in Basys 3 Xilinx Board with verilog .

DONE BY :

HARSHAVARDHAN R -107123043
PRASANNA - 107123081