|  |
| --- |
|  |
| The Eclipse Clock |
| Digital System Design Final Project |

|  |
| --- |
|  |

Project made by:

Eclipse Team

Ahmed Abdel-Sattar T-12 25-11046

Mahmoud El-Desouky T-10 25-3541

Mohamed Lotfy T-12 25-8385

Mohamed Saeed Salama T-12 25-12064

Ahmed Bassiouny T-12 25-832

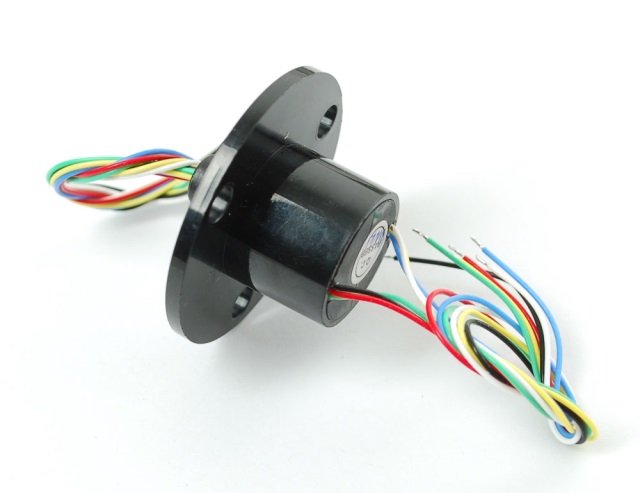
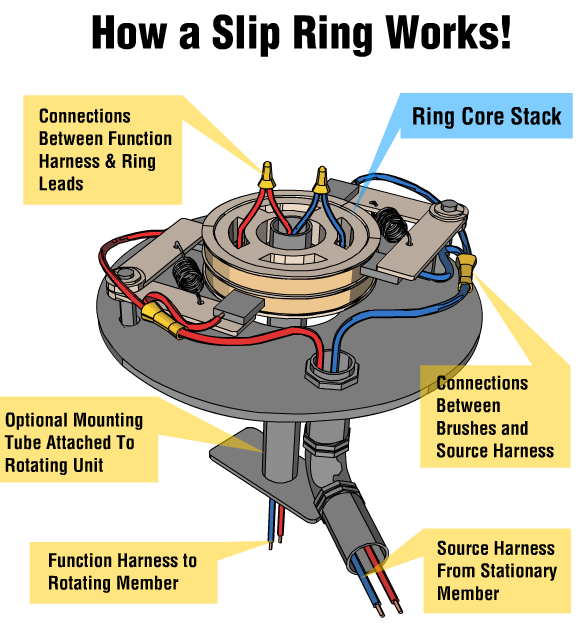
Under Supervision of Dr. Ayman El-Naggar

Introduction

Our project aim is to make an Eclipse clock. It displays the clock view on a working fan using the persistent of vision theory. The Eclipse clock project use an idea called POV (Persistence of Vision) with a proper definition: Persistence of vision is still the accepted term for this phenomenon in the realm of cinema history and theory. In the early days of film innovation, it was scientifically determinedthat a frame rate of a calculated range causes the mind to see flashing images which means that if something appears in the same spot consistently, at least 50-60 times per second, our brains think that it's permanently there when it really is not. TV's and Monitors use this method of display, so it's not as uncommon as you might think. This was illustrated by putting a seven-segment on the fan wings and manipulating them on and off using the Altera DE2-115 and its 50 megahertz clock. The speed of the fan {RPM} also plays a huge role in manipulating the POV phenomena and producing the final output we want.

Hardware:

* Connections:
  + The main and 1st problem that faced us was how we could connect the static wires coming from the fixed Altera with the dynamic seven-segment which is fixed on the rotating fan wings. As that passing the wires through the fan rotating axis will certainly cut the seven-segment wires. A fast and already exist solution for this problem was the slip-ring. A slip ring is an electromechanical device that allows the transmission of power and electrical signals from a stationary to a rotating structure. But unfortunately after relaying on this methodology we couldn’t find it in Egypt. We had to come with another solution back then, by making our own strategy in the seven-segment connection.



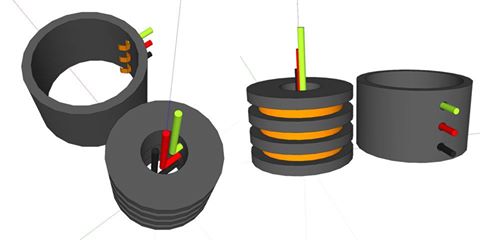
Our idea was to connect the seven-segment through sending the signals through connecting the Altera output pins to brushes that will touch different segments of the rotating cylinder on the rotating fan axis with each cylinder segment connected to a segment of the seven-segment through a hallow in the rotating axis.

To illustrate more, the hardware section was classified into three parts.

1. The first part was the rotating cylinder which we designed to have 10 grooved segments, each segment has a rotating inductive wire around it with its end passing to the fan wings (Third part) through inside the cylinder of the rotating fan. For the cylinder we used artelon material as it's an isolator, light and stronger material.

2. The second part is the arm that carries 10 inductive brushes, with each brush touching a different segment on the rotating cylindrical and the other end of the brush is connect to the Altera pins; So that each brush will send the signal form the Altera to the corresponding segment on the cylindrical axis that is connected a specific segment on the seven segment that is built-on the fan wing. For the arm we used artelon.

3. The third part is the fan wing that is consisting of two identical parts corresponding to each other on the end of the rotating cylinder with each having a seven-segment built on it. Only one seven-segment is used. The other fan wing and seven-segment are only used to keep the fan balance during rotation. The seven-segment take its connection through the wire that comes from the inside of the rotating cylinder.

This is a figure to help you understanding the idea of the first part.

All of the three parts were designed by us with our own original model in the university workshop with the help of technician. Although that this move took us 2 days to finish the routine paper work.

* Motor
* The target was to find a proper motor with a high enough RPM that will make us reach the POV (persistence of vision) phenomena.

In this point we faced 2 problems

1. The 1st one was that the RPM (rotations per minute ) that was written on the motors we bought wasn’t accurate enough for us to make the right calculation for controlling the seven-segment using the Altera clock to display the static clock display.
2. The 2nd problem was that the RPM (rotations per minute) of the motors is not steady "with average ±10% efficiency" which also makes us unable to make the right calculation for controlling the seven-segment using the Altera clock.

Our approach for these problems was at first that we tried different types of motors (4 different motors).

1. The first motor was a 9 volt DC motor with 2400 RPM (rotations per minute). The motor's torque was very weak and could not handle the weight of the rotating cylinder and the fan wings
2. The second motor was a 12 volt DC CPU fan motor with 3500 RPM (rotations per minute). We choose that motor as that we have read that this motor type is very accurate and precise. More over that it was the DC motor with the highest RPM we could get. Unfortunately we have faced the same torque problem as the first motor.

Then we had to change our methodology to use a AC motor.

1. The third motor was a 220 volt AC refrigerator compressor motor with 3500 RPM (rotations per minute). After reaching our final stages with this motor in the hardware section, we found out that this motor has an extremely inaccurate RPM as that it have different phases with different speed and it transfer from a phase to another automatically which ruins all our calculations.
2. Eventually the forth motor is a 220 Volt AC house hold stand fan with 1200 RPM (rotations per minute). The RPM of this fan was not the required one. But with lots of calculations and try & error we finally managed it and reached the required output through manipulating the calculations of the Altera 50 MHZ clock.



That was the proto-type of our second motor (CPU motor).



Display of the third motor model with the rotating cylinder on the motor axis and the brush arm, showing the grooves on the cylinder with the inductive wire around each segment.

* Software

The main focus in our programming section was how to manipulate our calculations in a way that will compensate that lag of the motor's RPM

The code of these 9 modules are at the end of the report.

We have 9 modules in our project

1. The first module concerned about counting seconds which takes 4 parameters called sixty counters. The first parameter is the count from 1 to 60 and the second run as a flag to state that it finished counting to 60. The other two parameters are inputs first one is the clock and the other one is a reset button.

The same goes for the next module

1. Twenty four counter is the main difference between the two modules is that the first module counts to 60 and the other counts up till 24.
2. The clock generator module takes 2 parameters one input and one output the input is the 50MHz that we take from the Altera board and the output is the generated clock to count every one second.
3. Another clock generator 1 module which takes the 50MHz clock from the Altera and returns back a clock to calculate the degree made from one cycle which we initially calculated it by mathematical equations.(the equation is stated at the last page).
4. The seven segment decoder module which takes 2 parameters the first one is the input number to be shown on the seven segments and the output is the seven-segment display for the number.
5. Hours\_min\_secs module which is the main module used to control the clock display on the wing of the fan and the module takes five parameters the first parameter is the output to be shown from the hours, minutes and the seconds the other four parameters are inputs hours, min, sec "seconds" and the clock.
6. Eclipse module is the module that used to show the word eclipse (the team name) on the seven-segment on the wing of the fan.
7. Change module is used to control the output shown either to show the clock or the word "eclipse" by giving a signal to a button on the Altera to activate one of the two outputs.
8. Our top level module is Eclipse clock which relate all the previous modules together to show the required output.

The generated clock for seven-segment ( 25 000 000 / RPM (rotations per second) )/ 360

Our RPM = 1200

The seven-segment wires data sheet

|  |  |
| --- | --- |
| Segment Number | Wire Color |
| 1 | White Brown |
| 2 | Green |
| 3 | Brown |
| 4 | White Blue |
| 5 | Orange |
| 6 | Blue |
| 7 | White Green |

CHANGE

module change (output1 , input1 , input2 , changed);

input [6:0] input1;

input [6:0] input2;

input changed;

output [6:0] output1;

reg [6:0] output1;

always@(changed)

begin

if (changed)begin

output1 <= input1;

end else

begin

output1 <= input2;

end

end

endmodule

clk generator

module clk\_generator(clk\_1hz,clk);

input clk;

output clk\_1hz;

reg clk\_1hz;

reg[24:0]count;

always@ (posedge clk)

begin if(count==25\_000\_000)begin

count <= 0;

clk\_1hz <= ~clk\_1hz;

end else begin

count <= count +1;

clk\_1hz <= clk\_1hz;

end

end

endmodule

clk generator for every degree

module clk\_generator1(clk\_1hz,clk);

input clk;

output clk\_1hz;

reg clk\_1hz;

reg[34:0]count;

always@ (posedge clk)

begin if(count==3465)begin

count <= 0;

clk\_1hz <= ~clk\_1hz;

end else begin

count <= count +1;

clk\_1hz <= clk\_1hz;

end

end

endmodule

60 counter for minutes and seconds

module sixty\_counter (count,count\_complete,clk,reset);

input clk,reset;

output count\_complete;

output [7:0] count;

reg[3:0] tens\_counter , units\_counter;

assign count = {tens\_counter,units\_counter};

reg count\_complete;

//assign count\_complete = 1'b1;

always @(posedge clk) begin

if(reset) begin

tens\_counter <=0;

units\_counter <=0;

count\_complete <=0;

end else if(units\_counter == 9) begin

if(tens\_counter==5) begin

count\_complete <=1;

units\_counter <= 0;

tens\_counter <=0;

end else begin

count\_complete <=0;

units\_counter <=0;

tens\_counter = tens\_counter + 1;

end end else begin

units\_counter <=units\_counter +1;

count\_complete <=0;

end

end

endmodule

24 counter for hours

module twentyfour\_counter (count,count\_complete,clk,reset);

input clk,reset;

output count\_complete;

output [7:0] count;

reg[3:0] tens\_counter , units\_counter;

assign count = {tens\_counter,units\_counter};

reg count\_complete;

//assign count\_complete = 1'b1;

always @(posedge clk) begin

if(reset) begin

tens\_counter <=0;

units\_counter <=0;

count\_complete <=0;

end

else

if(units\_counter == 9)

begin

count\_complete <=0;

units\_counter <=0;

tens\_counter = tens\_counter + 1;

end

else if(units\_counter == 3 && tens\_counter == 2)

begin

count\_complete <=1;

units\_counter <= 0;

tens\_counter <=0;

end

else

begin

units\_counter <=units\_counter +1;

count\_complete <=0;

end

end

endmodule

Seven segment decoder

module seven\_segment\_decoder(num,segments);

input[3:0]num;

output[6:0]segments;

reg [6:0] segments;

always@(num)

begin

case (num)

0: segments <= ~7'b0111111; // "6 5 4 3 2 1 0"

1: segments <= ~7'b0000110;

2: segments <= ~7'b1011011; // --0--

3: segments <= ~7'b1001111; // | |

4: segments <= ~7'b1100110; // 5 1

5: segments <= ~7'b1101101; // | |

6: segments <= ~7'b1111101; // --6--

7: segments <= ~7'b0000111; // | |

8: segments <= ~7'b1111111; // 4 2

9: segments <= ~7'b1101111; // | |

default: segments <= 7'bx; // --3—

endcase

end

endmodule

Translating from static clock to the eclipse clock

module hrs\_mins\_secs(output1,secst,secsu,hrs,mins,secs,clk);

input [13:0] hrs;

input [13:0] mins;

input [13:0] secs;

input clk;

output [6:0] output1;

output [6:0] secst;

output [6:0] secsu;

reg [6:0] output1;

//reg [1:0] count ;

reg [32:0] count = 2'b01;

reg [6:0] secst;

reg [6:0] secsu;

//assign mins\_segments = {mins\_tens\_segments,mins\_units\_segments};

always@(posedge clk)

begin

if (count == 1 )

begin

output1 <= hrs[13:7] ;

secst <= hrs[13:7];

count <= count +1 ;

end

else if (count == 2 )

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

else if (count == 5)

begin

output1 <= hrs [6:0];

secsu <= hrs [6:0];

count <= count +1;

end else if(count == 6)

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

else

if (count ==28) begin

output1 <= mins [13:7];

count <= count +1;

end

else if (count == 29)

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

else if (count == 34)

begin

output1 <= mins [6:0];

//secsu <= secs [6:0];

count <= count +1;

end

else if(count == 35)

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

if (count == 57) begin

output1 <= secs [13:7];

count <= count +1;

end

else if (count == 58)

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

else if (count == 62)

begin

output1 <= secs [6:0];

//secsu <= secs [6:0];

count <= count +1;

end

else if(count == 63)

begin

output1 <= 6'b0000000 ;

count <= count +1;

end

else if (count == 900)

begin

count <= 1;

end

else

begin

count <= count +1;

end

end

endmodule

This is module that showing the team name (eclipse)

module eclipse(output1,clk);

//input [13:0] hrs;

//input [13:0] mins;

//input [13:0] secs;

input clk;

output [6:0] output1;

//output [6:0] secst;

//output [6:0] secsu;

reg [6:0] output1;

//reg [1:0] count ;

reg [32:0] count = 2'b01;

//reg [6:0] secst;

//reg [6:0] secsu;

//assign mins\_segments = {mins\_tens\_segments,mins\_units\_segments};

always@(posedge clk)

begin

if (count == 1 )

begin

output1 <= 7'b1111001 ;

//secst <= hrs[13:7];

count <= count +1 ;

end

else if (count == 2 )

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

else if (count == 5)

begin

output1 <= 7'b0111001;

//secsu <= hrs [6:0];

count <= count +1;

end else if(count == 6)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

else

if (count == 9) begin

output1 <= 7'b0111000;

count <= count +1;

end

else if (count == 10)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

else if (count == 13)

begin

output1 <= 7'b0110000;

//secsu <= secs [6:0];

count <= count +1;

end

else if(count == 14)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

if (count == 17) begin

output1 <= 7'b1110011;

count <= count +1;

end

else if (count == 18)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

else if (count == 21)

begin

output1 <= 7'b1101101;

//secsu <= secs [6:0];

count <= count +1;

end

else if(count == 22)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

if (count == 25) begin

output1 <= 7'b1111011;

count <= count +1;

end

else if (count == 26)

begin

output1 <= 7'b0000000 ;

count <= count +1;

end

else if (count == 900)

begin

count <= 1;

end

else

begin

count <= count +1;

end

end

endmodule

The Top Level module (Eclipse Clock Module)

module direc (output1,secs\_segments,changed,reset,clk);

output [13:0] secs\_segments;

output [6:0] output1;

wire [6:0] output2;

wire [6:0] output3;

wire [13:0] secs\_segments1;

wire [13:0] mins\_segments;

wire [13:0] hours\_segments;

//output [13:0] mins\_segments;

input reset;

input clk;

input changed;

wire clk\_1hz,min\_complete,hour\_complete,day\_complete,reset;

wire [7:0] mins,secs,hrs;

wire [6:0] secst , secsu;

assign secs\_segments1 = {secs\_tens\_segments,secs\_units\_segments};

assign mins\_segments = {mins\_tens\_segments,mins\_units\_segments};

assign hours\_segments = {hours\_tens\_segments,hours\_units\_segments};

assign secs\_segments = {~secst,~secsu};

clk\_generator clk2 (clk\_1hz,clk);

clk\_generator1 clk3 (clk\_min,clk);

sixty\_counter seconds (secs,min\_complete,clk\_1hz,~reset);

sixty\_counter minutes (mins,hour\_complete,min\_complete,~reset);

twentyfour\_counter hours (hrs, day\_complete,hour\_complete,~reset);

wire [6:0] secs\_units\_segments,secs\_tens\_segments;

wire [6:0] mins\_units\_segments,mins\_tens\_segments;

wire [6:0] hours\_units\_segments,hours\_tens\_segments;

seven\_segment\_decoder secs\_units (secs[3:0],secs\_units\_segments);

seven\_segment\_decoder secs\_tens (secs[7:4],secs\_tens\_segments);

//assign mins\_segments = secs\_segments;

seven\_segment\_decoder mins\_units (mins[3:0],mins\_units\_segments);

seven\_segment\_decoder mins\_tens (mins[7:4],mins\_tens\_segments);

seven\_segment\_decoder hrs\_units (hrs[3:0],hours\_units\_segments);

seven\_segment\_decoder hrs\_tens (hrs[7:4],hours\_tens\_segments);

//assign output1 =~secs\_units\_segments ;

hrs\_mins\_secs hrs\_mins\_secs1(output2,secst,secsu,~hours\_segments,~mins\_segments,~secs\_segments1,clk\_min);

eclipse eclipse1(output3,clk\_min);

change change1 (output1 , output3 , output2 , changed);

endmodule