CSE490/590 PROJECT 1 REPORT

VERILOG SORTING

Name: NARMADHA VISWANATHAN
UB Person #: 50169758
Date: 3/11/2016

I. ABSTRACT

This project is to design a circuit that input 4 numbers, sort and display the sorted numbers by decimal on a 7-segment display. The design was implemented by having one top level module and three sub modules under the top level module. The three modules are used for getting the four inputs, sorting them and display them in the 7 segment display. In the input module the inputs and outputs to the module are declared. PartA is used to determine the index of the value to be stored. PartB contains the values to be sorted. PartC button has to be enables for the inputs to be read into the index location. The unsorted numbers are passed as inputs to the next module which performs sorting. The inputs and outputs to the sorting module are declared and the basic bubble sorting algorithm is performed and the outputs are stored in four different variables. PartD has to be pressed for the sorting to be performed. The sorted variables are passed to the output module where the outputs are made to display with a specified delay.

II. FLOW CHART

The flow chart is shown in Fig. 1.

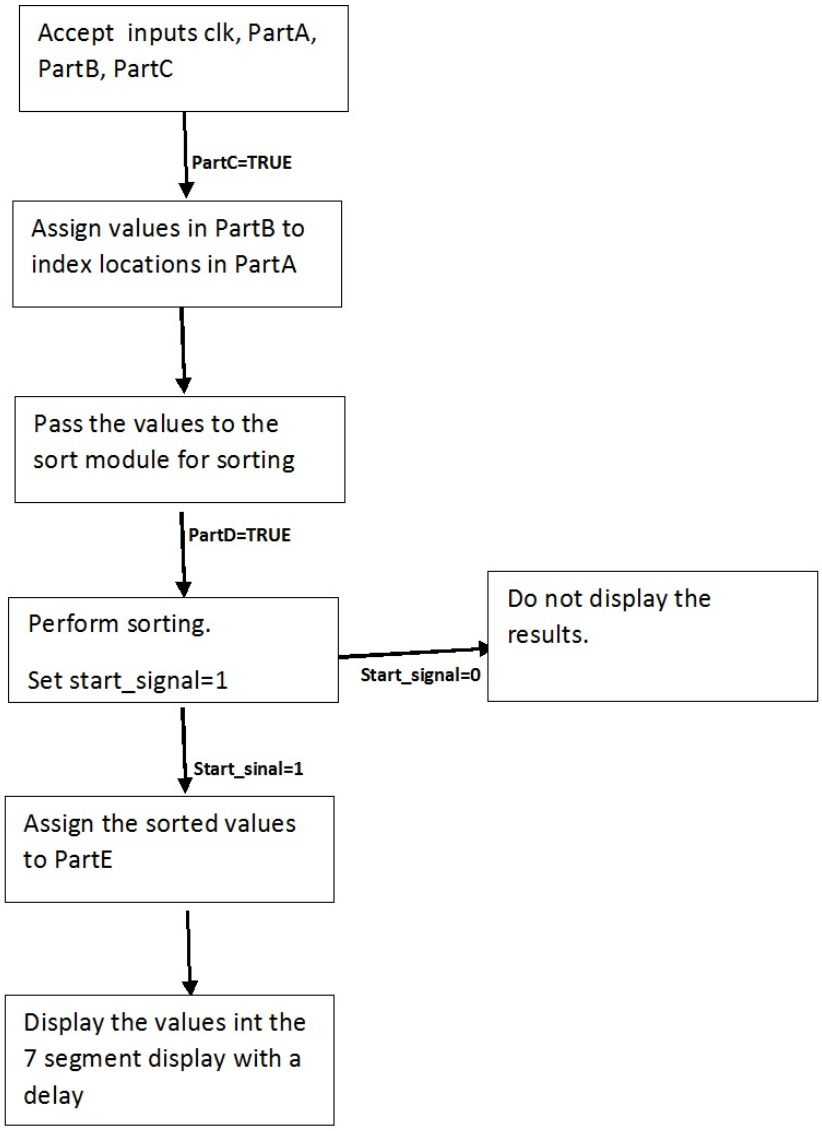

Fig.1. Flow Chart

III. INTERNAL SPECIFIC DESIGN

Code Part 1 - Input

```
module input_part(
        input clk,
         input [3:0] partA,
         input [3:0] partB,
         input partC,
         output reg [3:0] unsorted_num0,
         output reg [3:0] unsorted_num1,
         output reg [3:0] unsorted_num2,
         output reg [3:0] unsorted_num3
           );
         always @(posedge clk)
         begin
         if(partC !=0)
         begin
         case(partA)
         4'b0001: unsorted_num0 = partB;
         4'b0010: unsorted_num1 = partB;
         4'b0100: unsorted_num2 = partB;
         4'b1000: unsorted_num3 = partB;
         endcase
end
end
endmodule
```

Code Part 2 - Sort

```
module sorting_part(
input clk,
input partD,
       input [3:0] unsorted_num0,
       input [3:0] unsorted_num1,
       input [3:0] unsorted_num2,
       input [3:0] unsorted_num3,
      output reg [3:0] sorted_num0,
```

```
    output reg [3:0] sorted_num1,
    output reg [3:0] sorted_num2,
    output reg [3:0] sorted_num3,
    output reg start_display
    );
     integer i;
     integer j;
     reg [3:0] temp1;
     reg [3:0] temp[3:0];
     always@(posedge clk)
     begin
     temp[0]=unsorted_num0;
     temp[1]=unsorted_num1;
     temp[2]=unsorted_num2;
     temp[3]=unsorted_num3;
     if (partD!=0)
     begin
     for(i=0;i<4;i=i+1)
     begin
     for(j=0;j<4;j=j+1)
     begin
                if(temp[i]<temp[j])
                begin
                     temp1=temp[i];
                     temp[i]=temp[j];
                     temp[j]=temp1;
                         end
     end
     end

     sorted_num0=temp[0];
     sorted_num1=temp[1];
     sorted_num2=temp[2];
     sorted_num3=temp[3];
     start_display=1;
    end
    end
endmodule
```

Code Part 3 - Output

```
module output_part(
input clk,
input [3:0] sorted_num0,
input [3:0] sorted_num1,
input [3:0] sorted_num2,
input [3:0] sorted_num3,
input start_display,
```

```
output reg [3:0] partE
);
always @(posedge clk)
begin
if(start_display==1)
begin
partE=sorted_num0;
#500;
partE=sorted_num1;
#500;
partE=sorted_num2;
#500;
partE=sorted_num3;
#500;
end
end
endmodule
```

IV. RESULTS