CSE490/590 PROJECT 1 REPORT

VERILOG SORTING

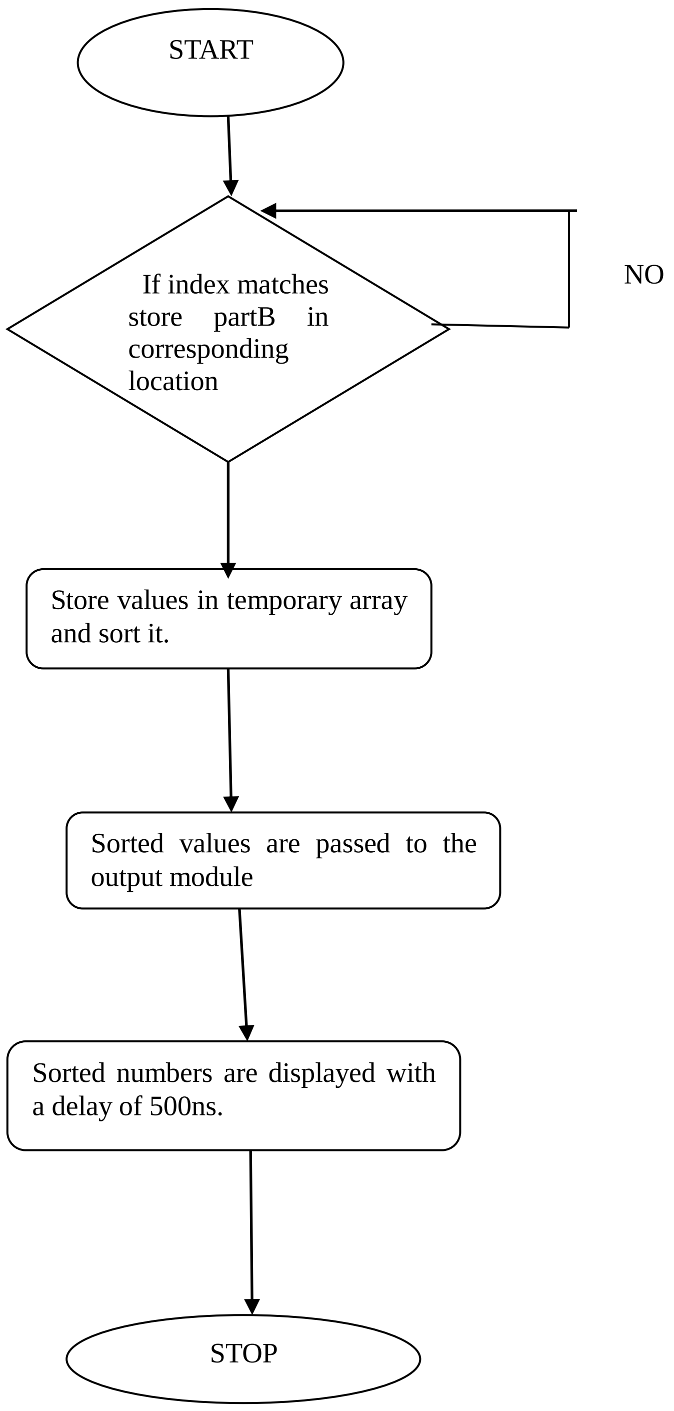
Name: Bhavani Sundara Raman
UB Person #: 50169253
Date: 3/11/2016

I. ABSTRACT

In this project we have implemented bubble sort using Verilog. The program is split into three modules. Input part, sorting part and output part. In the input part, we check the index of partA and store the corresponding value of partB. When a key partC is pressed the data in partB is inserted into location defined by partA. The values are passed to the sorting function. In the sorting function when key partD is pressed, these value are first stored in a temporary variable and sorted. The sorted values are then stored in sorted_num. After the sorting is done, variable start_display is set to one. This is passed as the output from the sorting part to the output part. In output part, we check for the condition when start_display is true and display the values with 500 nanoseconds delay.

II. FLOW CHART

The flow chart is shown in Fig. 1. Fig. 2. ……


START
If index matches store partB in corresponding location
NO
Store values in temporary array and sort it.
Sorted values are passed to the output module
Sorted numbers are displayed with a delay of 500ns.
STOP

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: begin
        unsorted_num0= partB;
        end
        4'b0010: begin
        unsorted_num1= partB;
        end
        4'b0100: begin
        unsorted_num2= partB;
        end
        4'b1000: begin
        unsorted_num3= partB;
        end
    endcase
    end
```

```
       end
endmodule
```

Code Part 2 - Sort

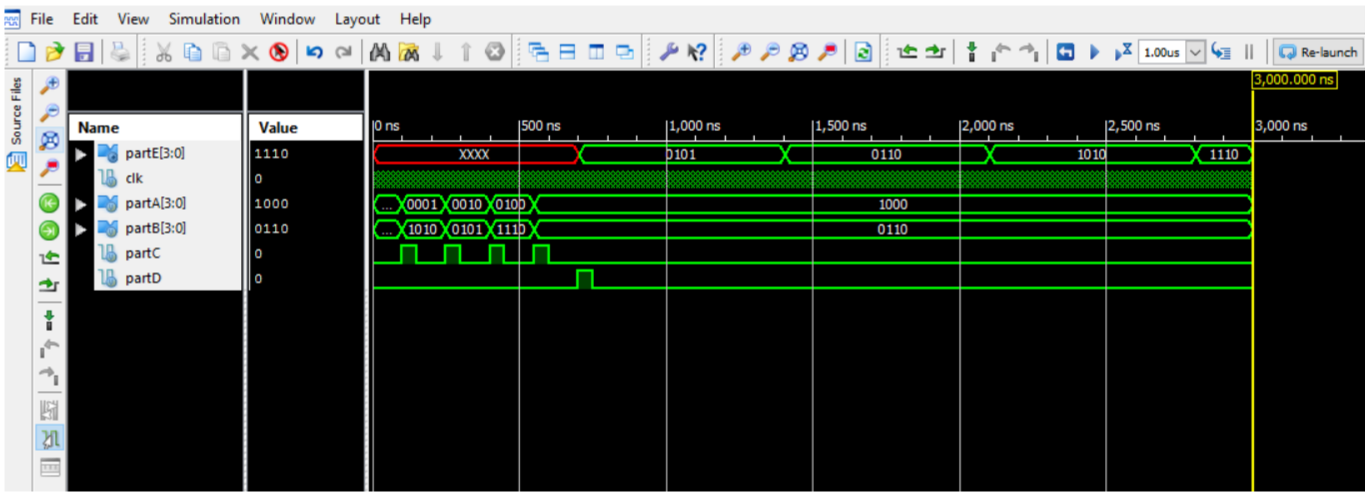
```
module sorting_part(
       input clk,
       input partD,
       input [3:0] unsorted_num0,
       input [3:0] unsorted_num1,
       input [3:0] unsorted_num2,
       input [3:0] unsorted_num3,
       output reg [3:0] sorted_num0,
       output reg [3:0] sorted_num1,
       output reg [3:0] sorted_num2,
      output reg [3:0] sorted_num3,
       output reg start_display );
         reg [3:0] t;
         reg [3:0] temp[3:0];
         integer i;
         integer j;
         always@(posedge partD)
         begin
         temp[0]=unsorted_num0;
         temp[1]=unsorted_num1;
         temp[2]=unsorted_num2;
         temp[3]=unsorted_num3;
         if(partD !=0)
         begin
         for(i=1; i<4; i=i+1)
         begin
         t=temp[i];
              for(j=i-1;j>=0 && t<temp[j];j=j-1)
              begin
                     temp[j+1]=temp[j];
                     end
                     temp[j+1] = t;
       end
       end
       sorted_num0=temp[0];
       sorted_num1=temp[1];
       sorted_num2=temp[2];
       sorted_num3=temp[3];
       start_display=1;
       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;
    #700;
    partE=sorted_num1;
    #700;
    partE=sorted_num2;
    #700;
    partE=sorted_num3;
    #700;
    end
    end
endmodule
```

IV. RESULTS

OUTPUT SIMULATION SHOWING SORTED RESULTS



Name	Value
partE[3:0]	1110
clk	0
partA[3:0]	1000
partB[3:0]	0110
partC	0
partD	0