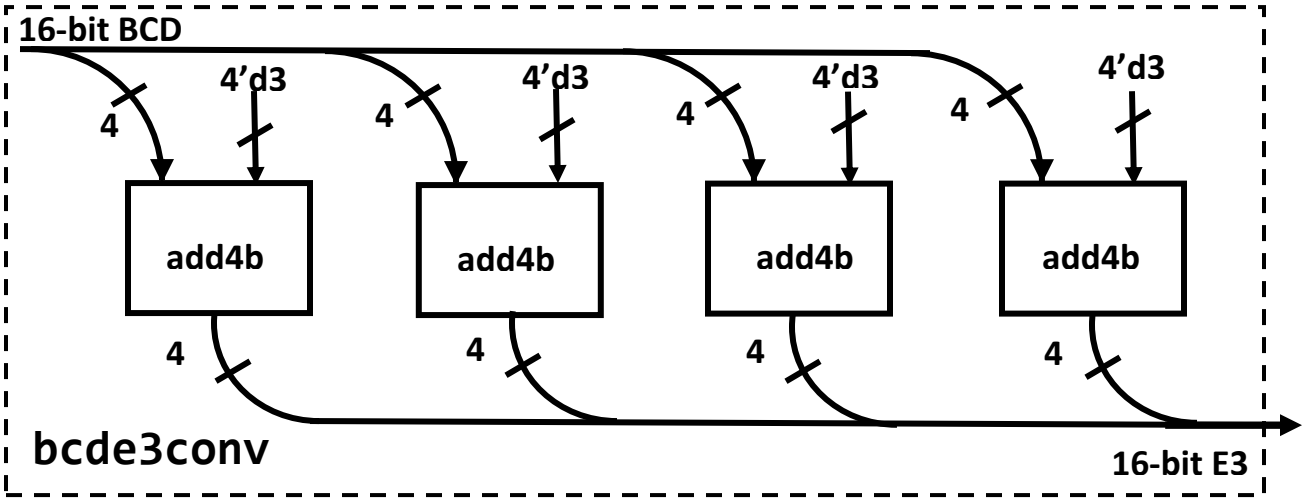
Laboratory Week 10 CA

Building multiple instances in Verilog language

E.10.1 Design the architecture of a ***BCD8421*** to ***E3*** converter for ***k***-digit numbers, using ***k*** instances of a 4-bit adder, called ***add4b***. Implement the design in Verilog language, using a ***generate*** construct.

The complete architecture is given below:



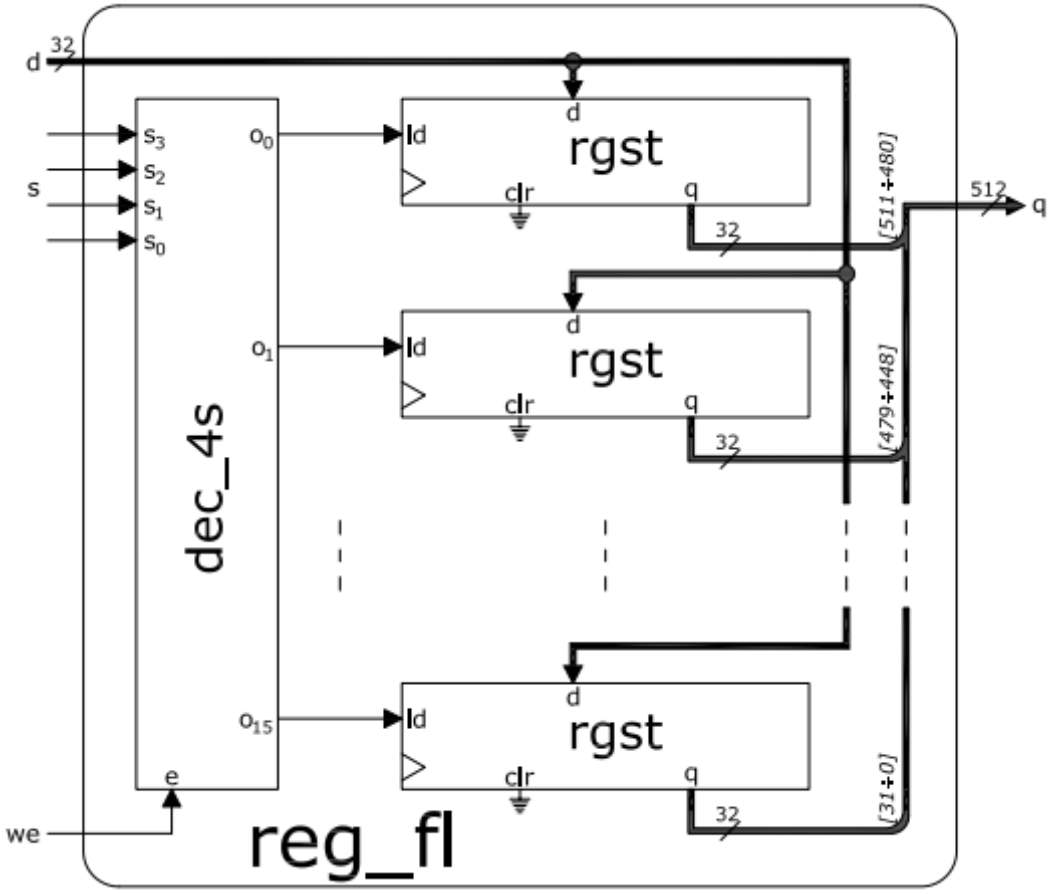
E.10.2 Design the architecture of a Register File 16 x 32, as illustrated below. The ***reg_fl*** module will receive 32-bit data packets at the input and its output will deliver a

512-bit data block by concatenating all the outputs of the internal registers of the architecture.

a) Implement the ***dec_4s*** module, using Verilog language.

b) Implement the parameterized ***rgst*** module, using Verilog language.

c) Construct the ***reg_fl*** architecture, using the multiple instantiation method.



a) Decoder module

```
module dec_4s ( input [3:0] s,
                input e,
                output reg [15:0] o );
  always @(*) begin
    case ({e, s})
    5'b10000: o = {15'b0, 1'b1};
    5'b10001: o = {14'b0, 1'b1, 1'b0};
    5'b10010: o = {13'b0, 1'b1, 2'b0};
    5'b10011: o = {12'b0, 1'b1, 3'b0};
    5'b10100: o = {11'b0, 1'b1, 4'b0};
    5'b10101: o = {10'b0, 1'b1, 5'b0};
    5'b10110: o = {9'b0, 1'b1, 6'b0};
    5'b10111: o = {8'b0, 1'b1, 7'b0};
    5'b11000: o = {7'b0, 1'b1, 8'b0};
    5'b11001: o = {6'b0, 1'b1, 9'b0};
    5'b11010: o = {5'b0, 1'b1, 10'b0};
    5'b11011: o = {4'b0, 1'b1, 11'b0};
    5'b11100: o = {3'b0, 1'b1, 12'b0};
    5'b11101: o = {2'b0, 1'b1, 13'b0};
    5'b11110: o = {1'b0, 1'b1, 14'b0};
    5'b11111: o = {1'b1, 15'b0};
    5'b0????: o = 16'b0;
    endcase
  end
endmodule
```

b) Parameterized register module

```
module rgst # (
        parameter w = 8,
        parameter iv = {w{1'b0}}
)(
        input clk,
        input rst_b,
        input [w-1:0] d,
        input ld,
        input clr,
        output reg [w-1:0] q
);


always @ ( posedge clk, negedge rst_b)
        if(! rst_b)
                q <= iv;
        else if(clr)
                q <= iv;
        else if(ld)
                q <= d;
endmodule
```

c) Register File module with generate construct

```
module reg_fl( input clk, rst_b,
                    input [31:0] d,
                    input [3:0] s,
                    input we,
          output [511:0] q );
wire [15:0] r_ld;
dec_4x16 u_dec ( .s(s), .e(we), .o(r_ld) );
generate
          genvar i;
                    for( i=0; i<16; i=i+1 )
          begin: arr
          rgst #(
                    .w(32)
          ) u_rgst(
                    .clk(clk),
                    .rst_b(rst_b),
                    .d(d),
                    .clr(1'd0),
                    .ld(r_ld[i]),
                    .q(q[i*32+31 : i*32])
          );
          end
endgenerate
endmodule
```