```
/*
   Select and Encode Logic Block, Encodes the Ra, Rb and Rc signals from IR
   GRout and GRin outputs are selected using the signals Rin, Rout and BAout
   Also Provides a Sign-Extended C to the DataPath Bus
*/


module SelectAndEncodeLogic (

   input [31:0] IR,
   input Gra, Grb, Grc, Rin, Rout, BAout,

   output [31:0] C,
   output [15:0] GRin,
   output [15:0] GRout

);

parameter tmp = 16'b1;

// Sign extend C by Fanning IR[18] to IR[31:18]
assign C = {{14{IR[18]}} , IR[17:0]};

// Select the register to Decode
wire [3:0] RtoDecode = (IR[26:23] & {4{Gra}}) | (IR[22:19] & {4{Grb}}) | (IR[18:15] & {4{Grc}});

// 4-16 Decoder
wire [15:0] GRsignal =  (RtoDecode == 4'b0000) ? tmp   :
                        (RtoDecode == 4'b0001) ? tmp<<1:
                        (RtoDecode == 4'b0010) ? tmp<<2:
                        (RtoDecode == 4'b0011) ? tmp<<3:
                        (RtoDecode == 4'b0100) ? tmp<<4:
                        (RtoDecode == 4'b0101) ? tmp<<5:
                        (RtoDecode == 4'b0110) ? tmp<<6:
                        (RtoDecode == 4'b0111) ? tmp<<7:
                        (RtoDecode == 4'b1000) ? tmp<<8:
                        (RtoDecode == 4'b1001) ? tmp<<9:
                        (RtoDecode == 4'b1010) ? tmp<<10:
                        (RtoDecode == 4'b1011) ? tmp<<11:
                        (RtoDecode == 4'b1100) ? tmp<<12:
                        (RtoDecode == 4'b1101) ? tmp<<13:
                        (RtoDecode == 4'b1110) ? tmp<<14:
                        (RtoDecode == 4'b1111) ? tmp<<15: 16'bxxxx_xxxx_xxxx_xxxx;

// Set GRin and GRout based on Rin and Rout/BAout
assign GRin = GRsignal & {16{Rin}};
assign GRout = GRsignal & {16{Rout|BAout}};

endmodule


//--------Select and Encode Testbench--------//

module SelectAndEncodeLogic_TB ();

reg clk;

reg [31:0] IR;
reg Gra, Grb, Grc, Rin, Rout, BAout;

wire [31:0] C;
wire [15:0] GRin;
wire [15:0] GRout;


SelectAndEncodeLogic UUT (IR, Gra, Grb, Grc, Rin, Rout, BAout, C, GRin, GRout);

initial begin
   clk <= 0;
   forever #5 clk <= ~clk;
end


initial begin
```

```

        IR <= 32'b0;
        Gra <= 0; Grb <= 0; Grc <= 0; Rin <= 0; Rout <= 0; BAout <= 0;

        @(posedge clk)

        IR <= 32'h2A1B8000;
        Rout <=1;
        Gra <= 1;

        @(posedge clk)

        Gra <=0; Grb <= 1;

        @(posedge clk)

        Grb <= 0; Rout <= 0;
        Grc <= 1; Rin <= 1;

        @(posedge clk)

        Grc <= 0;

        IR <= 32'h0007FEED;

        @(posedge clk)

        IR <= 32'h0003BEEF;

        @(posedge clk)

        $stop;

    end

    endmodule
```