4 X 2 Encoder

Structural Modeling:

```
module Encoder2x4Structural(input [1:0] data_in, output reg [3:0] encoded);
   // Instantiate AND gates
   wire and0_out, and1_out, and2_out, and3_out;

   and and0(and0_out, data_in[0], ~data_in[1]);
   and and1(and1_out, ~data_in[0], data_in[1]);
   and and2(and2_out, data_in[0], data_in[1]);
   and and3(and3_out, ~data_in[0], ~data_in[1]);

   // Instantiate OR gate
   or or0(encoded[0], and0_out, and1_out, and2_out, and3_out);
   or or1(encoded[1], and0_out, and1_out, and2_out, and3_out);
   or or2(encoded[2], and0_out, and1_out, and2_out, and3_out);
   or or3(encoded[3], and0_out, and1_out, and2_out, and3_out);
endmodule
```

Behavioural Modeling:

```
module Encoder2x4Behavioral(input [1:0] data_in, output reg [3:0] encoded);
   // Behavioral modeling using always block
   always @ (data_in)
      begin
         case (data_in)
            2'b00: encoded = 4'b0001;
            2'b01: encoded = 4'b0010;
            2'b10: encoded = 4'b0100;
            2'b11: encoded = 4'b1000;
            default: encoded = 4'b0001;
```

```
        endcase
    end
endmodule
```

Data Flow Modeling:

```
module Encoder2x4DataFlow(input [1:0] data_in, output reg [3:0] encoded);
    // Data flow modeling using continuous assignments
    assign encoded = (data_in == 2'b00) ? 4'b0001 :
                     (data_in == 2'b01) ? 4'b0010 :
                     (data_in == 2'b10) ? 4'b0100 :
                                          4'b1000;
endmodule
```

2 X 4 Decoder

Structural Modeling:

```
module Encoder2x4Structural(input [1:0] data_in, output reg [3:0] encoded);
    // Instantiate AND gates
    wire and0_out, and1_out, and2_out, and3_out;

    and and0(and0_out, data_in[0], ~data_in[1]);
    and and1(and1_out, ~data_in[0], data_in[1]);
    and and2(and2_out, data_in[0], data_in[1]);
    and and3(and3_out, ~data_in[0], ~data_in[1]);

    // Instantiate OR gate
    or or0(encoded[0], and0_out, and1_out, and2_out, and3_out);
    or or1(encoded[1], and0_out, and1_out, and2_out, and3_out);
    or or2(encoded[2], and0_out, and1_out, and2_out, and3_out);
    or or3(encoded[3], and0_out, and1_out, and2_out, and3_out);
```

```
endmodule
```

Behavioral Modeling:

```
module Encoder2x4Behavioral(input [1:0] data_in, output reg [3:0] encoded);
   // Behavioral modeling using always block
   always @ (data_in)
      begin
         case (data_in)
            2'b00: encoded = 4'b0001;
            2'b01: encoded = 4'b0010;
            2'b10: encoded = 4'b0100;
            2'b11: encoded = 4'b1000;
            default: encoded = 4'b0001;
         endcase
      end
endmodule
```

Data Flow Modeling:

```
module Encoder2x4DataFlow(input [1:0] data_in, output reg [3:0] encoded);
   // Data flow modeling using continuous assignments
   assign encoded = (data_in == 2'b00) ? 4'b0001 :
                    (data_in == 2'b01) ? 4'b0010 :
                    (data_in == 2'b10) ? 4'b0100 :
                                         4'b1000;
endmodule
```