Structural Modeling:

```
module Mux4x1Structural(input [3:0] data_in, input [1:0] sel, output reg out);
   // Instantiate AND gates
   wire and0_out, and1_out, and2_out, and3_out;

   and and0(and0_out, data_in[0], ~sel[1], ~sel[0]);
   and and1(and1_out, data_in[1], ~sel[1], sel[0]);
   and and2(and2_out, data_in[2], sel[1], ~sel[0]);
   and and3(and3_out, data_in[3], sel[1], sel[0]);

   // Instantiate OR gate
   or or0(out, and0_out, and1_out, and2_out, and3_out);
endmodule
```

Behavioral Modeling:

```
 module mux4X1( in,sel,out);
input [3:0]in;
input [1:0]sel;
output reg out;

always @(*)
        begin
                case(sel)
                        2'b00: out=in[0];
                        2'b01: out=in[1];
                        2'b10: out=in[2];
                        2'b11: out=in[3];
                        default: out=1'b0;
                endcase
        end
endmodule
```

Data Flow Modeling:

```
module mux4X1( in,sel,out);
input [3:0]in;
input [1:0]sel;
output out;
assign out = sel[1] ? ( sel[0] ? in[3]: in[2]) : ( sel[0] ? in[1]: in[0]);
endmodule
```

1 X 4 Demux

Structural Modeling:

```
module Demux1x4Structural(input sel, input data_in, output reg [3:0] data_out);
  // Instantiate NOT gates
  wire not_sel, not_sel_bar;

  not not1(not_sel, sel);
  not not2(not_sel_bar, not_sel);

  // Instantiate AND gates
  and and0(data_out[0], data_in, not_sel_bar);
  and and1(data_out[1], data_in, not_sel);
  and and2(data_out[2], data_in, sel);
  and and3(data_out[3], data_in, sel, not_sel);
endmodule
```

Behavioral Modeling:

```
module Demux1x4Behavioral(input sel, input data_in, output reg [3:0] data_out);
  // Behavioral modeling using always block
  always @ (sel or data_in)
```

```
        begin
            case (sel)
                1'b00: data_out = 4'b0001;
                1'b01: data_out = 4'b0010;
                1'b10: data_out = 4'b0100;
                1'b11: data_out = 4'b1000;
                default: data_out = 4'b0001;
            endcase
        end
endmodule
```

Data Flow Modeling:

```
module Demux1x4DataFlow(input sel, input data_in, output reg [3:0] data_out);
    // Data flow modeling using continuous assignments
    assign data_out = (sel == 1'b00) ? 4'b0001 :
                      (sel == 1'b01) ? 4'b0010 :
                      (sel == 1'b10) ? 4'b0100 :
                                       4'b1000;
Endmodule
```