Behavioral modeling

```
module full_adder(
input a,b,c,
output reg sum,cout
    );
    always @(*)
  begin
  case ({a,b,c})
  3'b000: sum = 0;
  3'b001: sum = 1;
  3'b010: sum = 1;
  3'b011: sum = 0;
  3'b100: sum = 1;
  3'b101: sum = 0;
  3'b110: sum = 0;
  3'b111: sum = 1;
  default : sum = 0;
  endcase
    case ({a,b,c})
  3'b000: cout = 0;
  3'b001: cout = 0;
  3'b010: cout = 0;
  3'b011: cout = 1;
  3'b100: cout = 0;
  3'b101: cout = 1;
  3'b110: cout = 1;
  3'b111: cout = 1;
  default : cout = 0;
  endcase
  end
endmodule
```

Structural

```
module full_adder(
input a,b,c,
output sum,cout
    );
  wire w1,c1,c2,c3,out1;
  xor x1(w1,a,b);
  xor x2(sum,w1,c);

  and a1(c1,a,b);
  and a2(c2,b,c);
  and a3(c3,a,c);

  or o1(out1,c1,c2);
  or o2(cout,out1,c3);

endmodule
```

Data Flow Modeling:

```
module full_adder(
input a,b,c,
output sum,cout
    );

    assign sum = (a ^ b ^ c );
    assign cout = (a & b ) | (b & c) | (a & c);

endmodule
```

```
    assign sum = a ^ b ^ cin;
    assign cout = (a & b) | (a & cin) | (b & cin);
endmodule
```