(i) Structural Modeling:

```
module HalfSubtractorStructural(input a, input b, output diff, output borrow);
  wire w1, w2;

  // Instantiate XOR gate
  xor #(2) xor1(diff, a, b);

  // Instantiate AND gate
  and #(2) and1(w1, ~a, b);

  // Instantiate NOT gate
  not #(1) not1(w2, w1);

  // Borrow is the output of AND gate
  assign borrow = w2;
endmodule
```

Behavioral Modeling:

```
module HalfSubtractorBehavioral(input a, input b, output diff, output borrow);
  // Behavioral modeling using always block
  always @ (a or b)
    begin
      diff = a ^ b;
      borrow = ~a & b;
    end
endmodule
```

Data Flow Modeling:

```
module HalfSubtractorDataFlow(input a, input b, output diff, output borrow);
  // Data flow modeling using continuous assignments
  assign diff = a ^ b;
  assign borrow = ~a & b;
endmodule
```

FULL SUBTRACTOR

Structural Modeling:

```
module FullSubtractorStructural(input a, input b, input bin, output diff, output bout);
  wire w1, w2, w3;

  // Instantiate XOR gates
```

```
  xor #(2) xor1(diff, a, b);
  xor #(2) xor2(diff, diff, bin);

  // Instantiate AND gates
  and #(2) and1(w1, ~a, b);
  and #(2) and2(w2, ~diff, bin);
  and #(2) and3(w3, ~a, bin);

  // Instantiate OR gate
  or #(3) or1(bout, w1, w2, w3);
endmodule
```

Behavioral Modeling:

```
module FullSubtractorBehavioral(input a, input b, input bin, output diff, output bout);
  // Behavioral modeling using always block
  always @ (a or b or bin)
    begin
      diff = a ^ b ^ bin;
      bout = (~a & b) | ((~a ^ b) & bin);
    end
endmodule
```

Data Flow Modeling:

```
module FullSubtractorDataFlow(input a, input b, input bin, output diff, output bout);
  // Data flow modeling using continuous assignments
  assign diff = a ^ b ^ bin;
  assign bout = (~a & b) | ((~a ^ b) & bin);
endmodule
```