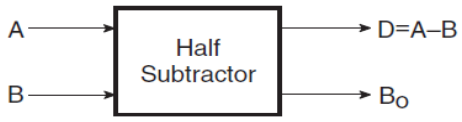
DAY-4

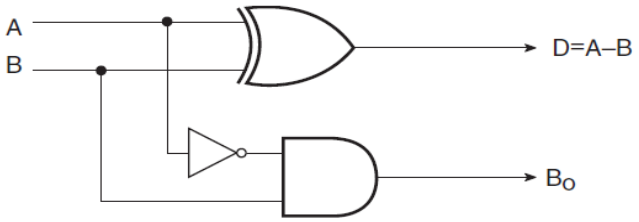
HALF SUBTRACTOR

EXPLANATION

Half subtractor is a combination circuit with two inputs and two outputs that are **different** and **borrow**. It produces the difference between the two binary bits at the input and also produces an output (Borrow) to indicate if a 1 has been borrowed. In the subtraction (A-B), A is called a **Minuend bit** and B is called a **Subtrahend bit.**

A	B	D	B_O
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0





Diff= A'B+AB'

Borrow = A'B

APPLICATIONS:

1.Calculators: Most mini-computers utilize advanced rationale circuits to perform numerical tasks. A Half Subtractor can be utilized in a number cruncher to deduct two parallel digits from one another.
2.Alarm Frameworks: Many caution frameworks utilize computerized rationale circuits to identify and answer interlopers. A Half Subtractor can be utilized in these frameworks to look at the upsides of two parallel pieces and trigger a caution in the event that they are unique.
3.Automotive Frameworks: Numerous advanced vehicles utilize computerized rationale circuits to control different capabilities, like the motor administration framework, stopping mechanism, and theater setup. A Half Subtractor can be utilized in these frameworks to perform computations and examinations.
4.Security Frameworks: Advanced rationale circuits are usually utilized in security frameworks to identify and answer dangers. A Half Subtractor can be

utilized in these frameworks to look at two double qualities and trigger a caution in the event that they are unique.

RTL CODE:

BEHAVIOURAL METHOD:

```
module half_sub(d,b1,a,c);
  output d,b1;
  reg d,b1;
  input a,c;
  always@(a or c)
  begin
    d= a^c;
    b1= (~a)&c;
  end
endmodule
```

DATA FLOW METHOD:

```
module half_sub(d,b1,a,c);
  output d,b1;
  input a,c;
    assign d= a^c;
    assign b1= (~a)&c;
  end
endmodule
```

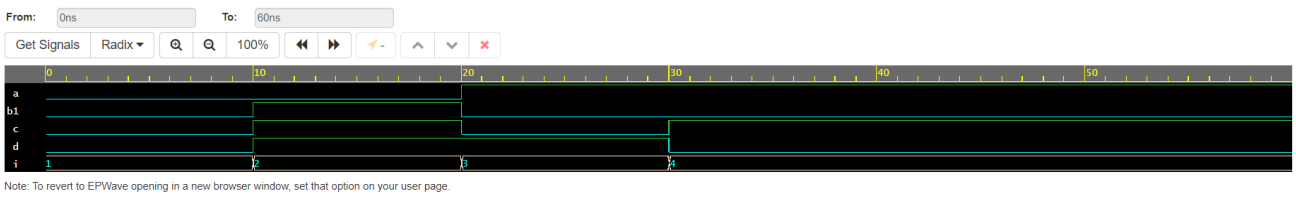
STRUCTURAL METHOD:

```
module half_sub(d,b1,a,c);
  output d,b1;
  input a,c;
  wire w1;
  xor a1(d,a,c);
  not b1(w1,a);
  and c1(b,w1,c);
endmodule
```

TEST BENCH:

```
module testbench;
  reg a,c;
  wire d,b1;
  integer i;
  half_sub a1(d,b1,a,c);
  initial
    begin
      $dumpfile("dump.vcd");
      $dumpvars(1);
    end
  initial
    begin
      a=0;c=0;
    end
  initial
```

```
    begin
      for(i=1;i<4;i=i+1)
        begin
          #10 {a,c}=i;
        end
    end
  initial
    begin
      #60 $finish();
    end
endmodule
```



Note: To revert to EPWave opening in a new browser window, set that option on your user page.