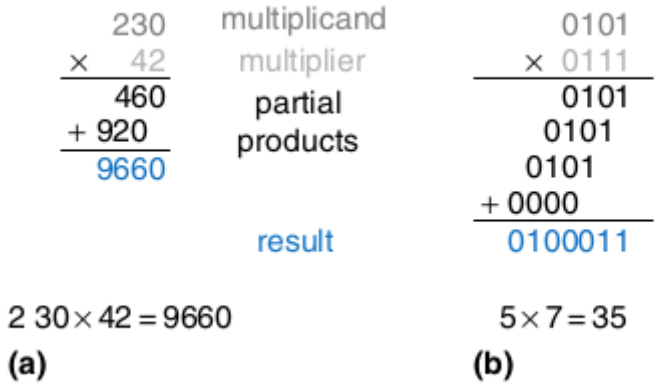
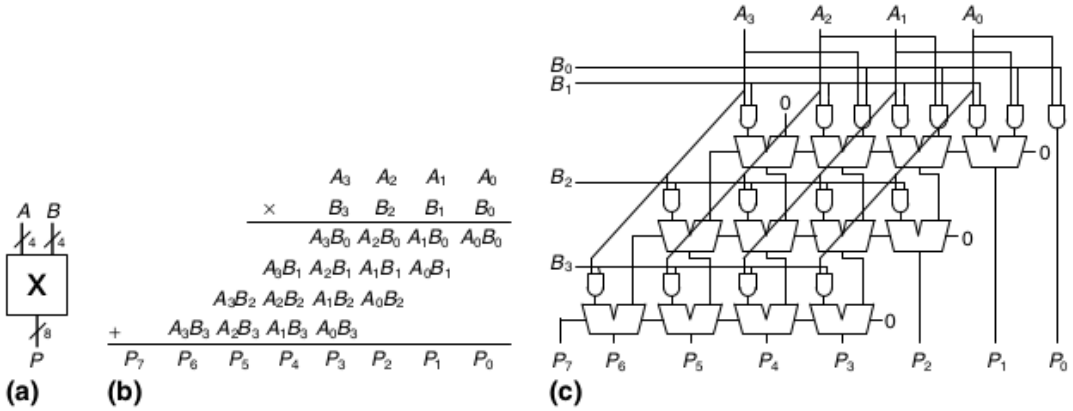
Proiectarea inmultitorului

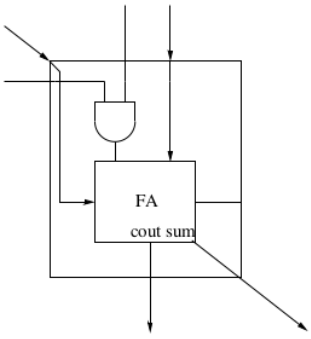
Multiplicarea numerelor binare este similară cu multiplicarea zecimală, cu diferenţa că la baza stau doar 1 logic şi 0 logic. Figura de mai jos compară multiplicarea în zecimale şi binare. În ambele cazuri, se formează produse parţiale prin înmulţirea unei singure cifre a multiplicatorului cu întregul multiplicator. Produsele parțiale schimbate sunt însumate pentru a forma rezultatul.



În general, un multiplicator N x N multiplică două numere de N-biţi şi produce un rezultat de 2 N-biţi. Produsele parţiale în înmulţirea binară sunt egase sau cu unul din multiplacatoare, sau cu 0 logic. Multiplicarea numerelor binare cu este echivalentă cu operația AND, de aceea pentru a forma produsele parţiale sunt folosite porţile ŞI.



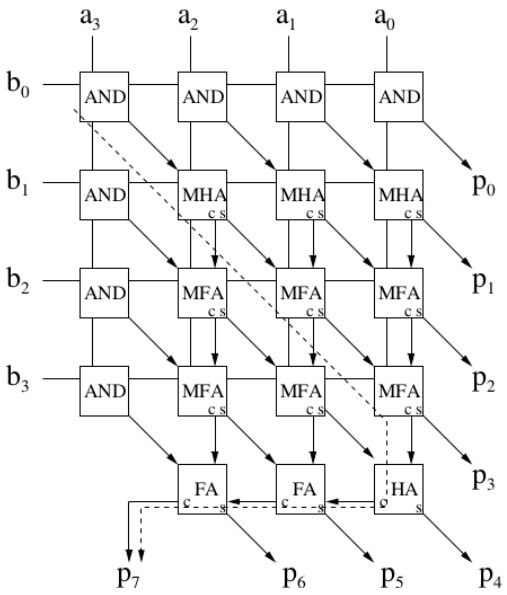
Pentru a calcula produsul între doi biţi se poate de utilizat doar poarta ”şi”, însa pentru a calcula produsul pentru numere mai mari este necesar de implementatat înmulţitor de tip matrice. La fiecare noi acestei matrice vor fi plasat un inmulţitor elementar. Acest inmulţitor elementar are ca funcţia calculul produsului de 2 biţi, şi calculul sumei acestui produs cu summa de intrare şi transport de intrare.



Inmulţitor elementar poate fi implementat în limbajul verilog în felul următor:

```
module mulE(A, B, Sin, Cin, Sout, Cout);
  input A, B, Sin, Cin;
  output Sout, Cout;
  assign {Cout, Sout} = (A & B) + Sin + Cin;
endmodule
```

Utilizînd acest inmulţitor elementar se constuieşte un matrice care permite calculul produsului pentru numerele mai mari. Exemplu de implementare a înmulţitorului de 4 biţi:



În limbaj Verilog Înmulţitor poate fi implementat în felul următor. (Este bazat pe doua module elementare).

```
module SumE(input wire a, input wire b, input wire Cin, output wire Sum, output wire Cout);
  assign Sum = a ^ b ^ Cin;
  assign Cout = (a & b) | (a ^ b) & Cin;
endmodule

module MulE(input wire Cin, input wire Sin, input wire a, input wire b, output wire Sout, output
wire Cout);
  assign {Cout, Sout} = Sin + Cin + (a & b);
endmodule

module Mul4(input wire [3 : 0] a, input wire [3 : 0] b, output wire [7 : 0] out);
  wire [3 : 0] Cout [3 : 0];
  wire [2 : 0] SumOut [3 : 0];
  wire [2 : 0] SumCarry;

  MulE mul1(1'b0, 1'b0, a[3], b[0], SumOut[0][0], Cout[0][0]);
  MulE mul2(1'b0, 1'b0, a[2], b[0], SumOut[0][1], Cout[0][1]);
  MulE mul3(1'b0, 1'b0, a[1], b[0], SumOut[0][2], Cout[0][2]);
  MulE mul4(1'b0, 1'b0, a[0], b[0], out[0], Cout[0][3]);
  MulE mul5(Cout[0][0], 1'b0, a[3], b[1], SumOut[1][0], Cout[1][0]);
```

```
  MulE mul6(Cout[0][1], SumOut[0][0], a[2], b[1], SumOut[1][1], Cout[1][1]);
  MulE mul7(Cout[0][2], SumOut[0][1], a[1], b[1], SumOut[1][2], Cout[1][2]);
  MulE mul8(Cout[0][3], SumOut[0][2], a[0], b[1], out[1], Cout[1][3]);
  MulE mul9(Cout[1][0], 1'b0, a[3], b[2], SumOut[2][0], Cout[2][0]);
  MulE mul10(Cout[1][1], SumOut[1][0], a[2], b[2], SumOut[2][1], Cout[2][1]);
  MulE mul11(Cout[1][2], SumOut[1][1], a[1], b[2], SumOut[2][2], Cout[2][2]);
  MulE mul12(Cout[1][3], SumOut[1][2], a[0], b[2], out[2], Cout[2][3]);
  MulE mul13(Cout[2][0], 1'b0, a[3], b[3], SumOut[3][0], Cout[3][0]);
  MulE mul14(Cout[2][1], SumOut[2][0], a[2], b[3], SumOut[3][1], Cout[3][1]);
  MulE mul15(Cout[2][2], SumOut[2][1], a[1], b[3], SumOut[3][2], Cout[3][2]);
  MulE mul16(Cout[2][3], SumOut[2][2], a[0], b[3], out[3], Cout[3][3]);
  SumE sum1(SumOut[3][2], Cout[3][3], 1'b0, out[4], SumCarry[0]);
  SumE sum2(SumOut[3][1], Cout[3][2], SumCarry[0], out[5], SumCarry[1]);
  SumE sum3(SumOut[3][0], Cout[3][1], SumCarry[1], out[6], SumCarry[2]);
  SumE sum4(1'b0, Cout[3][0], SumCarry[2], out[7], nothing);
endmodule
```

Mersul lucrării:

1. Implementaţi in limbaj Verilog **inmulţitor elementar**.
2. Implementaşi în limbaj Verilog **testbench** pentru **inmulţitor elementar**.
3. **Sintetizaţi inmulţitor elementar**.
4. **Implementaţi** în limbaj Verilog **inmulţitor de 4 biţi**.
5. Implementaţi **testbench** pentru **inmulţitor de 4 biţi**.
6. **Sintetizaţi inmulţitor de 4 biţi**.