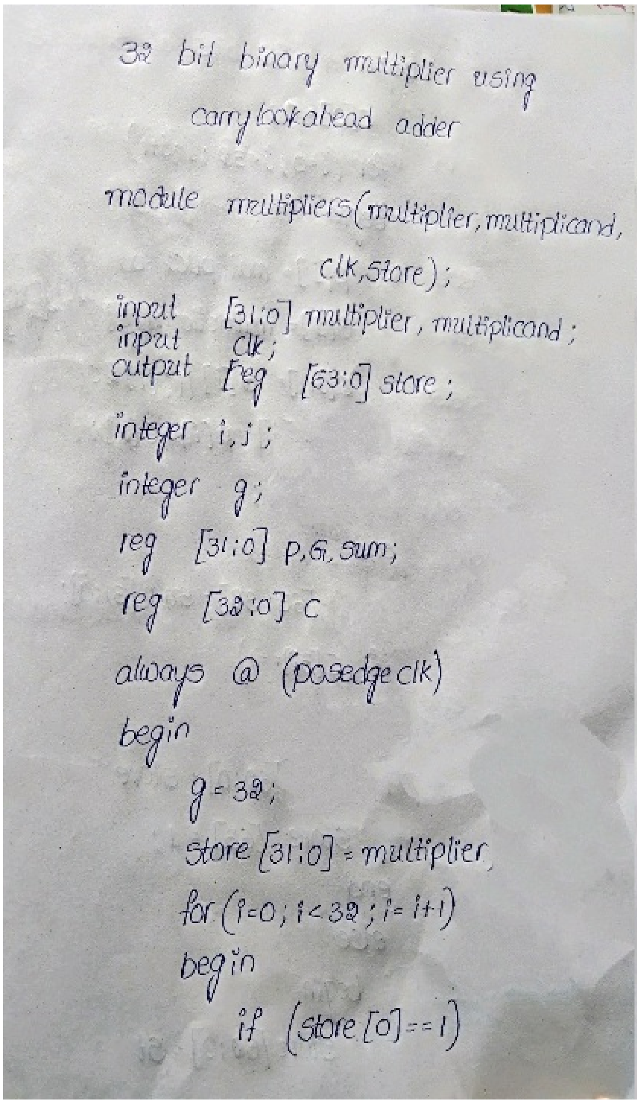
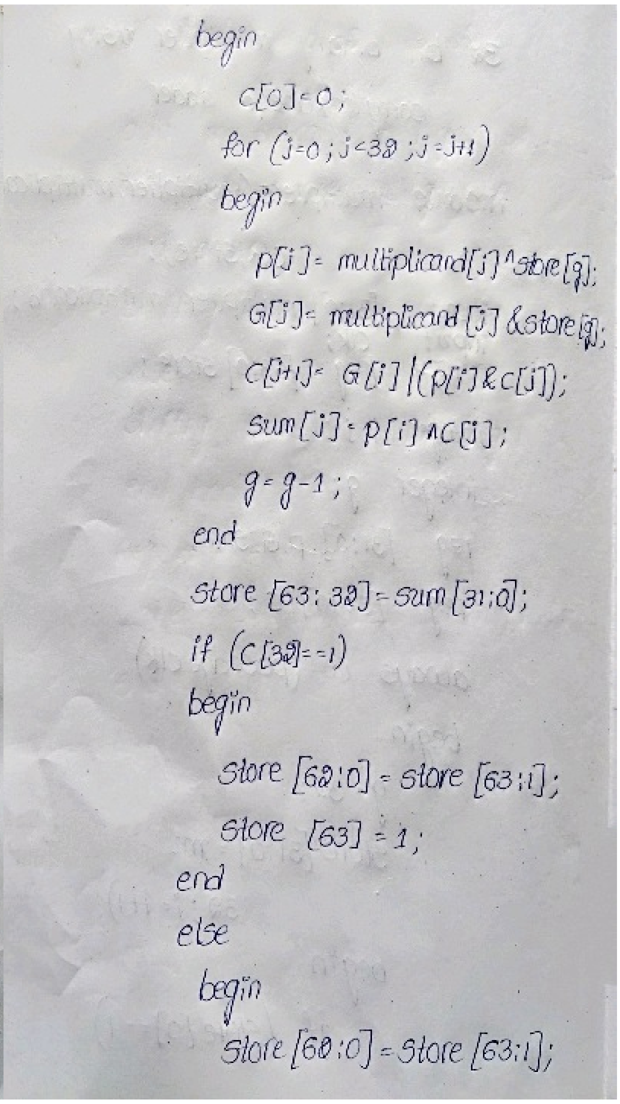
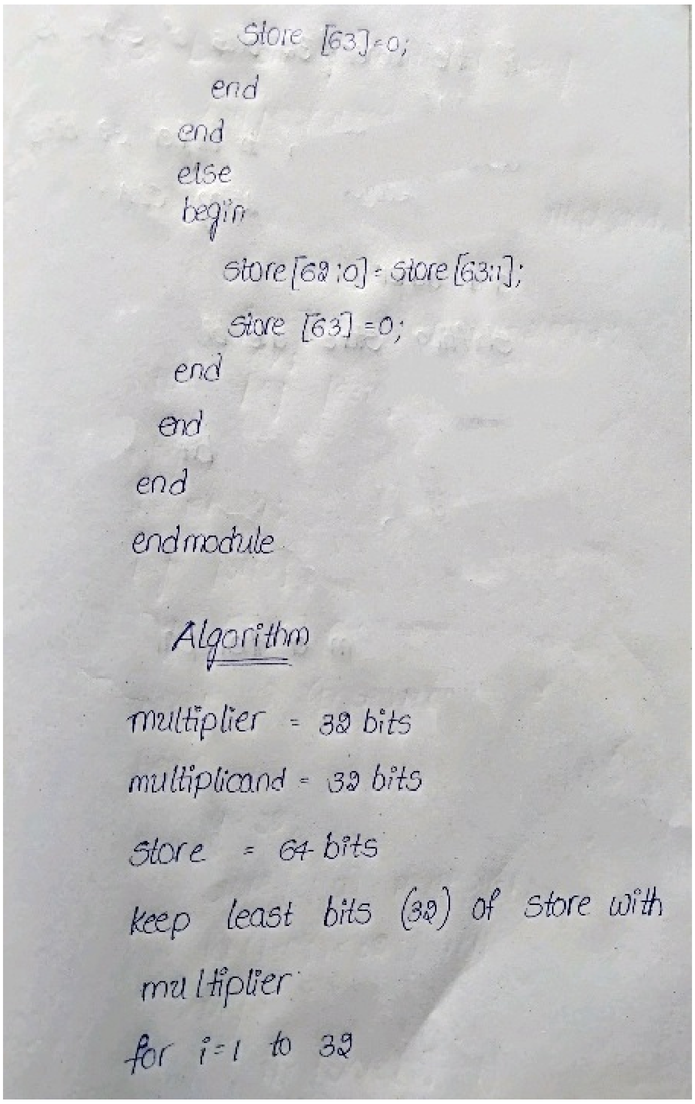


32 bit binary multiplier using
carrylookahead adder

```
module multipliers(multiplier,multiplicand,
                   clk,Store);
input  [31:0] multiplier, multiplicand;
input  clk;
output reg [63:0] store;
integer i,j;
integer g;
reg [31:0] p,G,sum;
reg [32:0] C
always @ (posedge clk)
begin
    g=32;
    store[31:0]=multiplier
    for (i=0;i<32;i=i+1)
    begin
        if (store[0]==1)
```



```
        begin
          C[0]=0;
          for (j=0;j<32;j=j+1)
          begin
            p[j]= multiplicand[j]^store[g];
            G[j]= multiplicand[j]&store[g];
            C[j+1]= G[i]|(p[i]&c[j]);
            sum[j]=p[i]^C[j];
            g=g-1;
        end
        store[63:32]=sum[31:0];
        if (C[32]==1)
        begin
           store[62:0]=store[63:1];
           store [63]=1;
        end
        else
         begin
          Store[62:0]=Store[63:1];
```



```
          Store [63]=0;
        end
     end
     else
     begin
         store[62:0] = store[63:1];
         Store [63] =0;
     end
   end
 end
endmodule
```

Algorithm

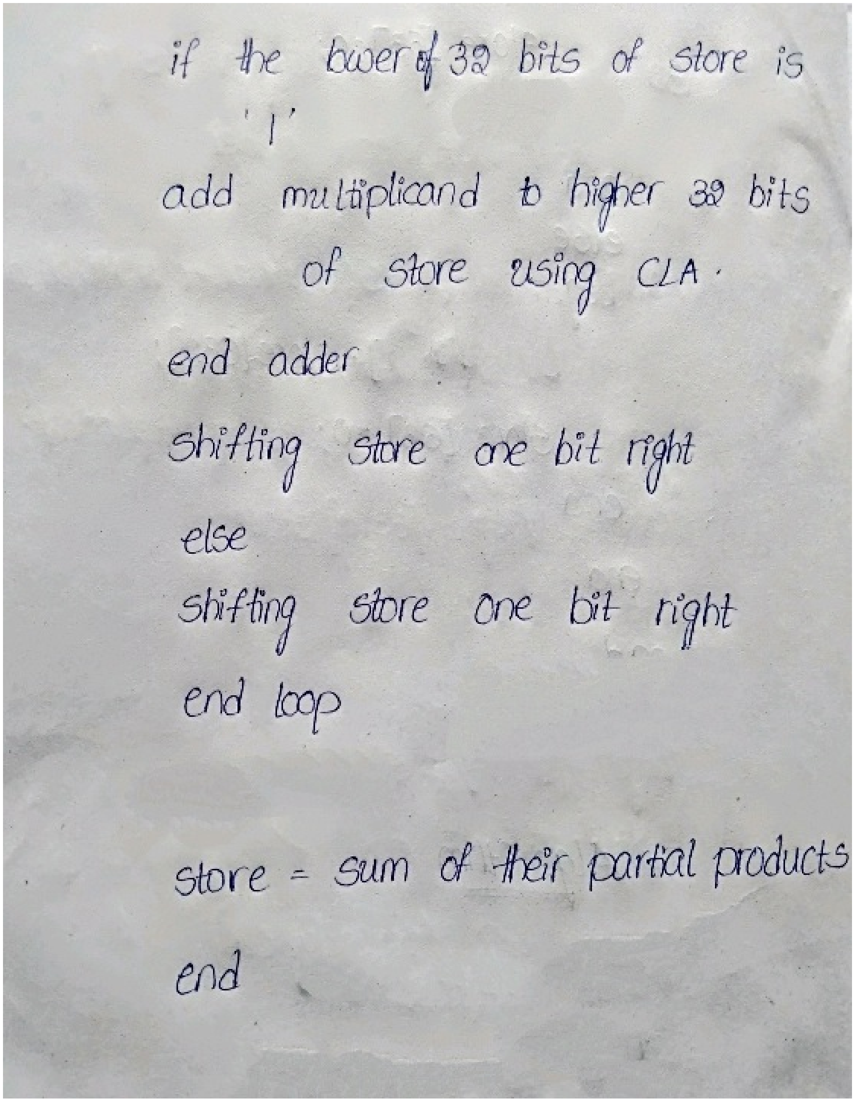
multiplier = 32 bits

multiplicand = 32 bits

store = 64 bits

keep least bits (32) of store with multiplier

for i=1 to 32



```
if the lower of 32 bits of store is
    '1'
add multiplicand to higher 32 bits
        of store using CLA.
end adder
shifting store one bit right
else
shifting store one bit right
end loop

store = sum of their partial products
end
```

3% bil binary multiplier using carry book alead adder module mahpliers(multiplier, multiplicand, clk,store): input [310] multiplier, multiplicand; citput reg (63:07 store; integer i, j; integer g; reg [31:0] P.G, Sum; reg [3210] C always @ (posedze cik) begin 9=32; Store (31:0) - multiplier for (9:0; 1432; 1= it!) begin if (store [0] == 1)
begin to C[OJ-O; for (i=0; j <39 ; j =itt) begin p[i]- multiplicard[3]"stre [9] G[3]- multiplicand [] &store C[j+1)G[] |(P[i]&c[j]); Sum[3] P[i] CU]; end Store [63:39) - sum [31:0]; it (c(31==1) begin Store (6a: 0] = store [63:1]; store [63] = 1; end else begin Slore [60:0) - Store [63:1]; Store (63)-0; end end else begin store (ca :0) - Store (63:1]; Gore [63] =0; end end end module Algorithm multiplier = 32 bits multiplicand - 39 bits store = 64 bits keep least bits (39) of Store with multiplier for i=1 to 32
if the bwer of 32 bits of sÅ,ore is add multiplicand to higher 39 bits of Store using CLA end adder Shifting Store one bit right else Shifting Store one bit right end loop Store - Sum of their partial products end