

**ECE 741/841**

21 November 2002

## Project Presentations

- 5-10 minutes.
- Explain design.
- Explain design decisions.
- Explain Verification theorems.

## Project Presentations

- Available
  - Overhead Projector.
  - Digital Projector.
  - Computer running Windows 98.
  - Computer running Linux and PVS.

## Loop Invariants

- Loop invariants are properties which hold inside a loop.
- A loop could be a recursive definition.
- A loop could be a `DO` or `while` or `for` in a programming language.
- A loop invariant helps in the verification of a program.

## Example, Java Program

```
public class mainp
{
    public static void main(String[] args)
    {
        int n = 15;

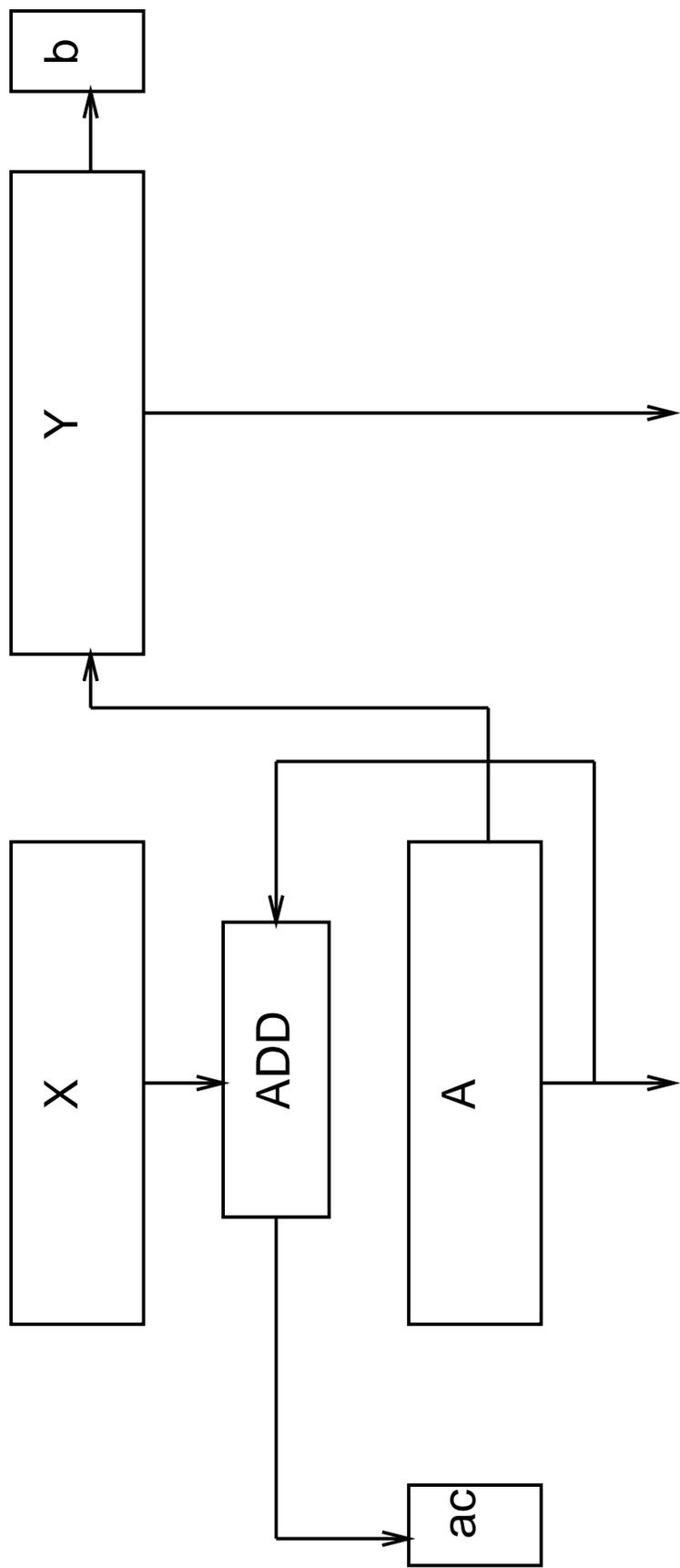
        format.p("n = "+n);
        format.p("fact 1 (n) "+sum1(n));
        format.p("fact 2 (n) "+sum2(n));

    }
}
```

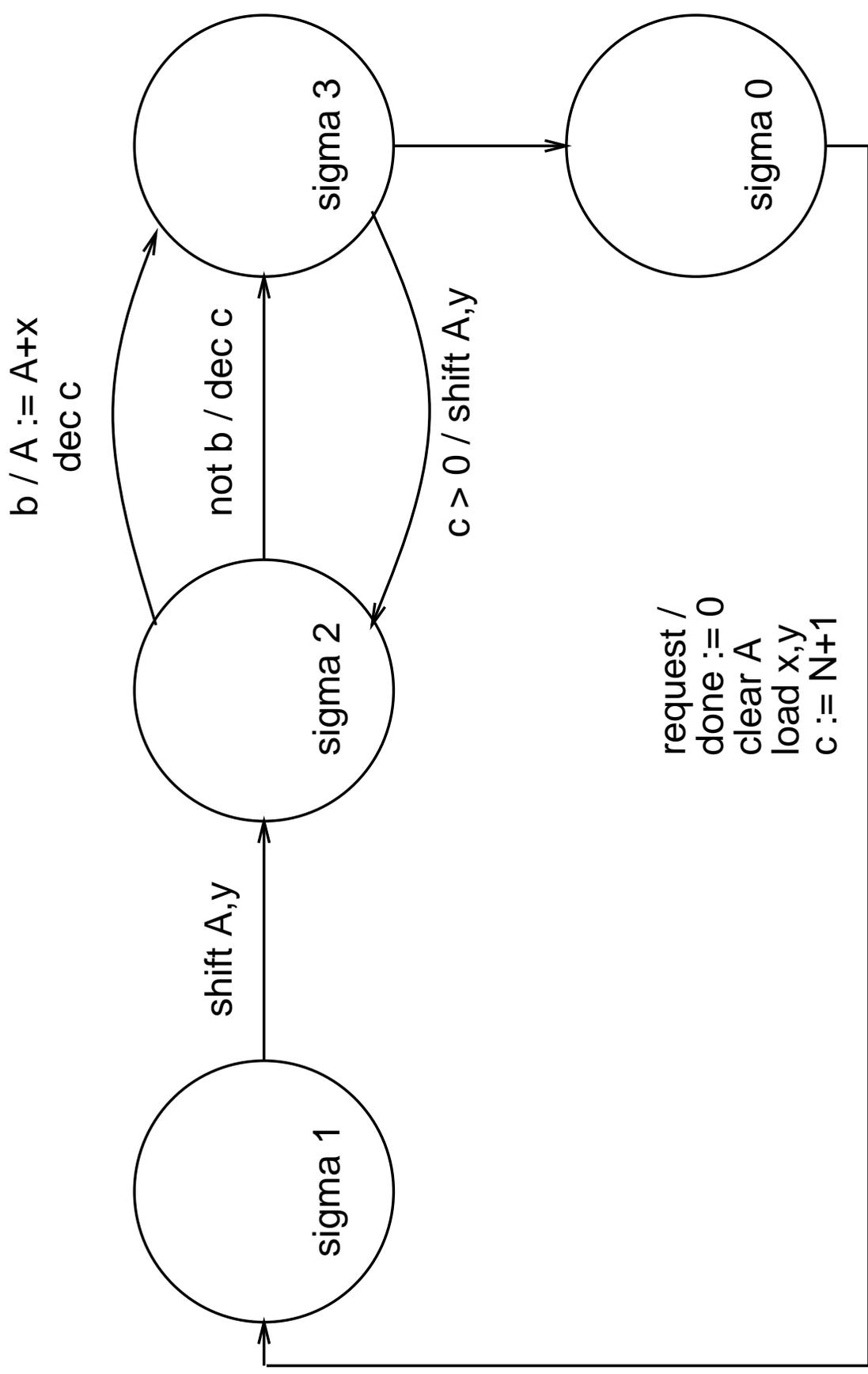
```
public static int sum1(int a)
{
    int ac = 0;
    while (a > 0)
    {
        ac = ac + a;
        a--;
    }
    return ac;
}
```

```
public static int sum2(int a)
{
    int ac = 0;
    for (int i=1;i <= a; i++)
    {
        ac = ac + i; // ac = i * (i + 1)/2
    }
    return ac;
}
}
```

# Multiplier Example



# Multiplier Example, FSM



## Multiplier in PVS (Emre's design)

```
loop(sigma: below(4), c: upto(N+1),
      request: bool, b: bit, X:bvec, Y:bvec,
      A:bvec, Ac:bit, n:nat):recursive [bvec, bvec]=
  let ctrl=control(sigma, c, request, b) in
  let nextX=NBreg(X, proj_4(ctrl), X, 0) in
  let nextY=NBreg(Y, proj_5(ctrl), Y, A(0)) in
  let nextb=OBreg(b, proj_6(ctrl), Y(0)) in
  let nextAc=OBreg(Ac, proj_7(ctrl), adder_cout(X, A, 0)) in
  let nextA=NBreg(A, proj_8(ctrl), adder_sum(X, A, 0), Ac) in
  if n=0 OR proj_3(ctrl) then (A, Y)
  else loop(proj_1(ctrl), proj_2(ctrl), request, nextb, nextX,
            nextY, nextA, nextAc, n-1)
endif
measure n
```

## Verification

```
multiply(X:bvec, Y:bvec) : [nat]=
let a=proj_1(loop(0,0,true,0,X,Y,v0,0,5*N+2)) in
let y=proj_2(loop(0,0,true,0,X,Y,v0,0,5*N+2)) in
(bv2nat(a)*exp2(N)+bv2nat(y))

th1:theorem
forall (X:bvec, Y:bvec) :
bv2nat(X)*bv2nat(Y)=multiply(X,Y)
```

## Loop Invariant

```
% The first n+1 LSB.

LSB(bv,n) : bvec =
(lambda (m:below[N]) : if m <= n then bv(m) else 0 endif)

th1b : lemma
let a=proj_1(loop(0,0,true,0,X,Y,v0,0,3)),
    y=proj_2(loop(0,0,true,0,X,Y,v0,0,3)) in
bv2nat(LSB(Y,0))*bv2nat(X) = bv2nat(a)

th1c : lemma
let a=proj_1(loop(0,0,true,0,X,Y,v0,0,4)),
    y=proj_2(loop(0,0,true,0,X,Y,v0,0,4)) in
bv2nat(LSB(Y,0))*bv2nat(X) = bv2nat(a)*exp2(1) + y(N-1)
```

```

th1d : lemma
let a=proj_1(loop(0,0,true,0,X,Y,v0,0,6)),
    y=proj_2(loop(0,0,true,0,X,Y,v0,0,6)) in
bv2nat( LSB(Y,1) ) * bv2nat(X) = bv2nat(a) * exp2(2) +
                                y(N-1) * exp2(1) + y(N-2)

```

## Loop Invariant

Allow us to prove the correctness of the multiplier for an arbitrary  $N$ .