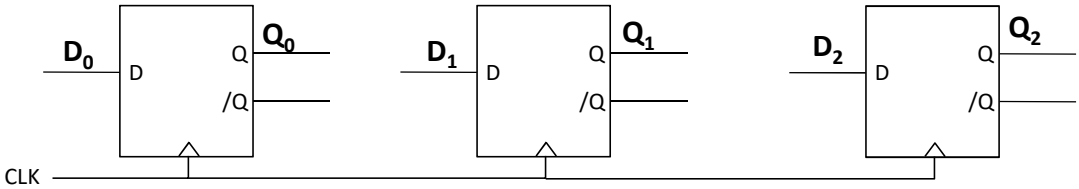
Digital systems: from bits to microcontrollers

- Introduction to digital systems fundamentals
- Combinatorial digital systems
- Sequential digital systems
- Introduction to microcontrollers
- Introduction to advanced implementation platforms

Revisiting registers

Sometimes is necessary that one register performs different functions along the time.

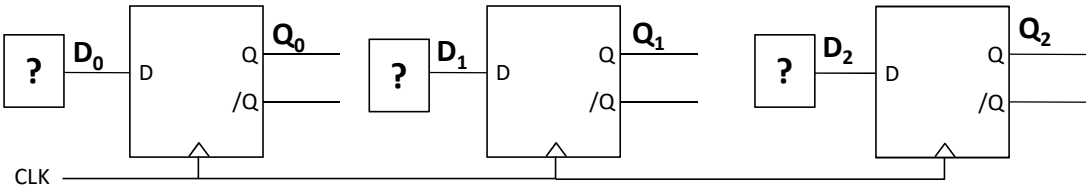
As the flip-flop structure is the same for all functions, the different functions are implemented through different flip-flop input functions.



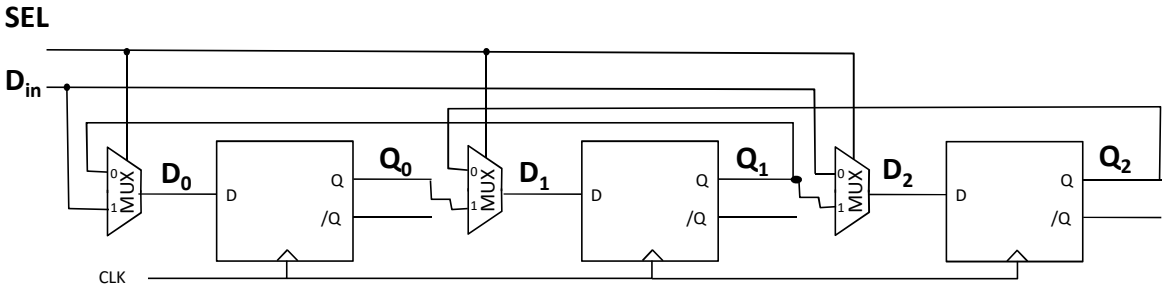
Revisiting registers

Sometimes is necessary that one register performs different functions along the time.

As the flip-flop structure is the same for all functions, the different functions are implemented through different flip-flop input functions.

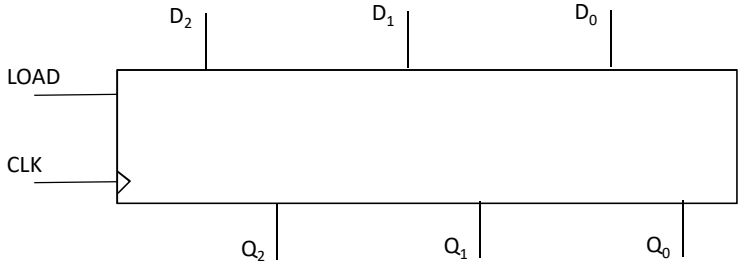


Left-right shift-register



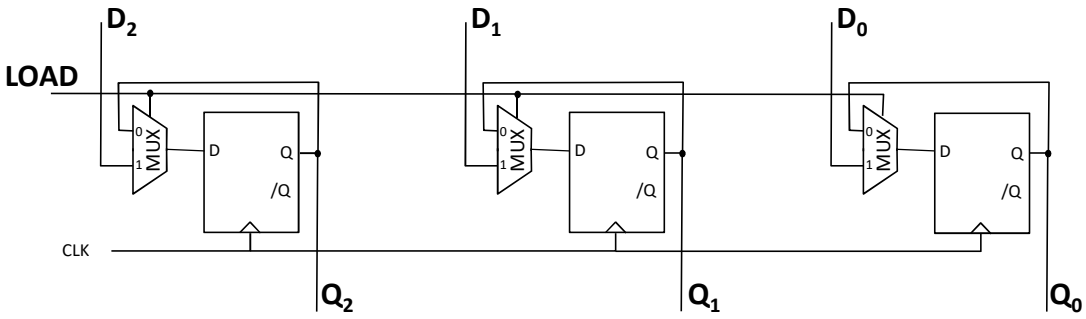
If (SEL = 0) then shift left
else shift right

Register with parallel load control



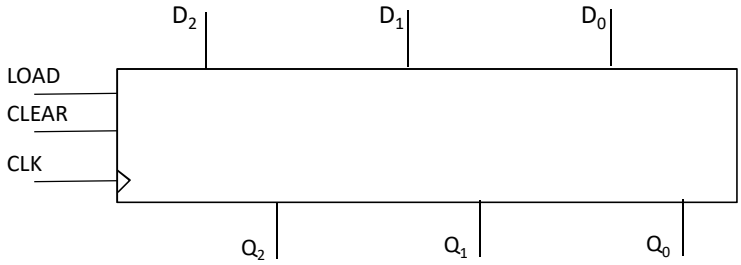
If (LOAD = 1) then load data inputs
else freeze outputs

Register with parallel load control



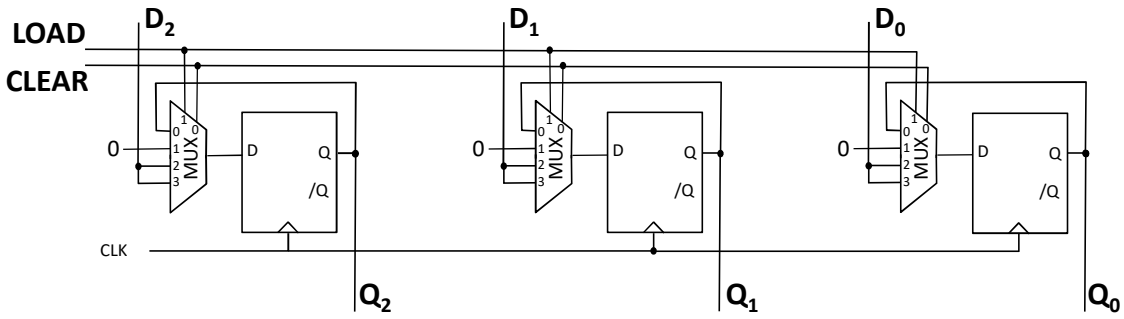
If (LOAD = 1) then load data inputs
else freeze outputs

Register with parallel load control and clear



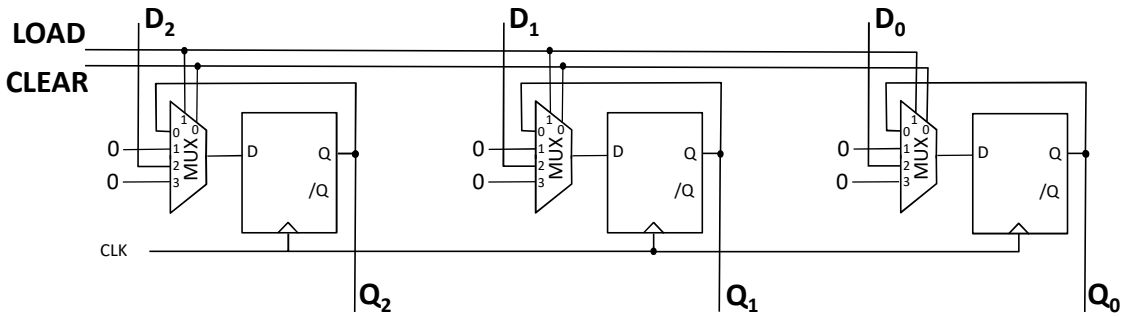
If (LOAD = 1) then load data inputs
else if (CLEAR = 1) then clear outputs
else freeze outputs

Register with parallel load control and clear



If (LOAD = 1) then load data inputs
else if (CLEAR = 1) then clear outputs
else freeze outputs

Register with parallel load control and clear



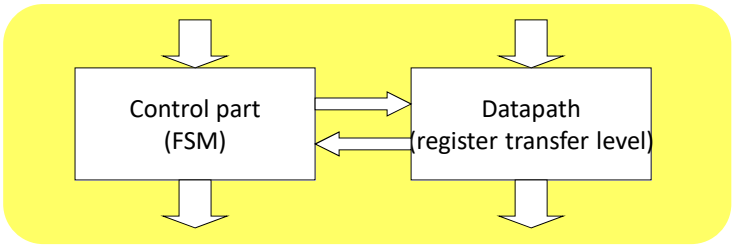
If (CLEAR = 1) then clear outputs
else if (LOAD = 1) then load data inputs
else freeze outputs

Coping with data processing dominated systems → FSM with Datapaths

Finite State Machine ← "Control-dominated"

Finite State Machine cooperative with data processing architecture ← "Data-processing-dominated"

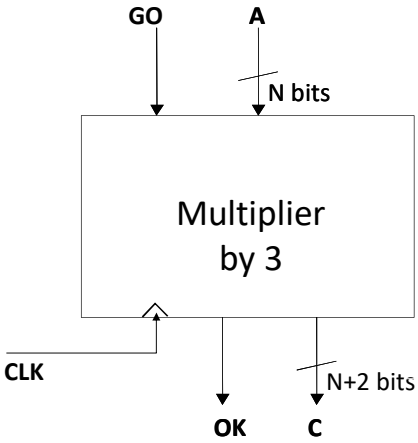
→ Partitioning the system into control part and datapath



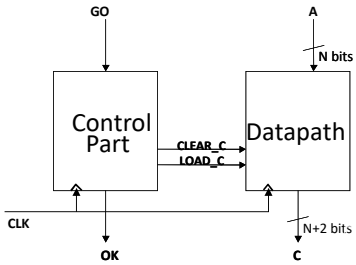
A simple multiplier

Goal: to implement one multiplier of one number (with N bits) by 3.

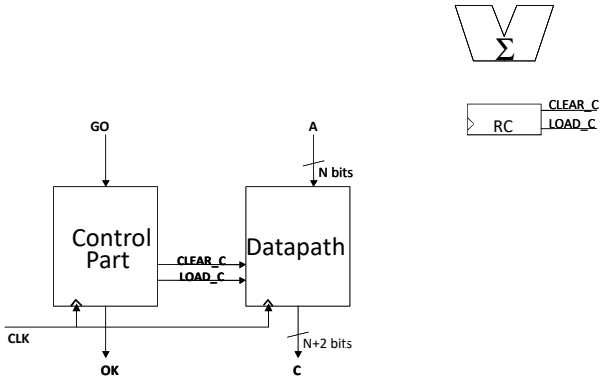
Strategy: Result = 0 + A + A + A



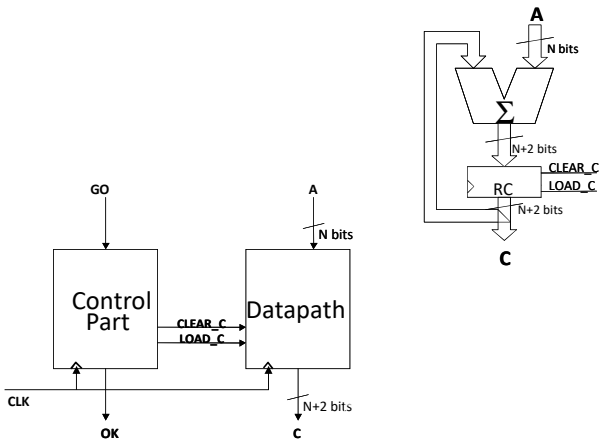
A 3*A multiplier (C = 0+A+A+A)



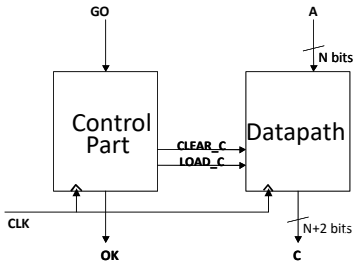
A 3*A multiplier (C = 0+A+A+A)

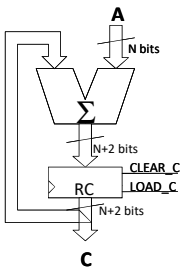


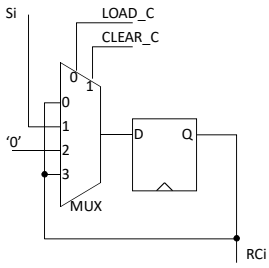
A 3*A multiplier (C = 0+A+A+A)



A 3*A multiplier (C = 0+A+A+A)

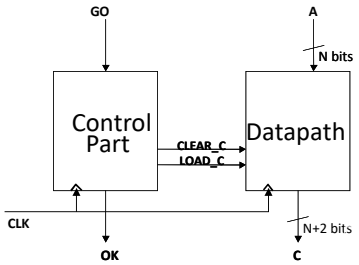


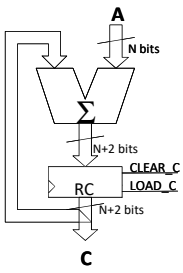


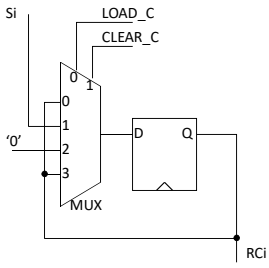


A 3*A multiplier (C = 0+A+A+A)

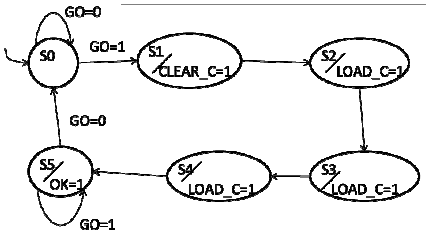
- wait for GO=1
- Clear RC
- Add A
- Add A
- Add A
- Wait for GO=0



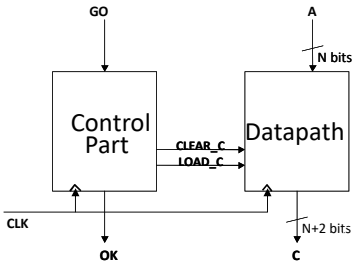


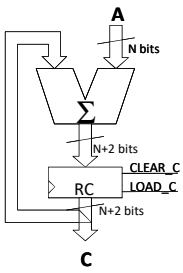


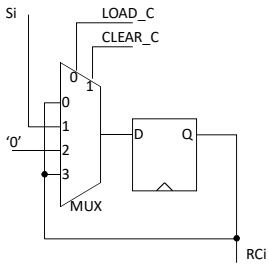
A 3*A multiplier (C = 0+A+A+A)



- wait for GO=1
- Clear RC
- Add A
- Add A
- Add A
- Wait for GO=0





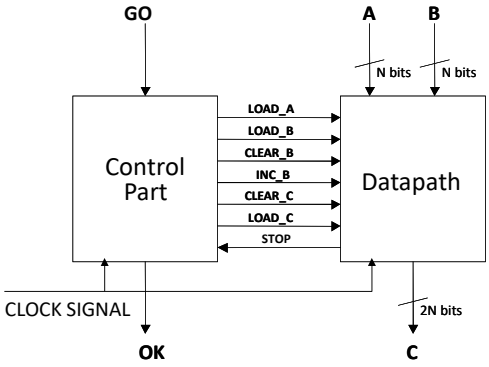


Exercise: multiplier of two numbers

Goal: to implement one multiplier of two numbers (with N bits each).

One multiplier (I)

Goal: to implement one multiplier of two numbers (with N bits each), using successive additions algorithm.



$C = A \times B$

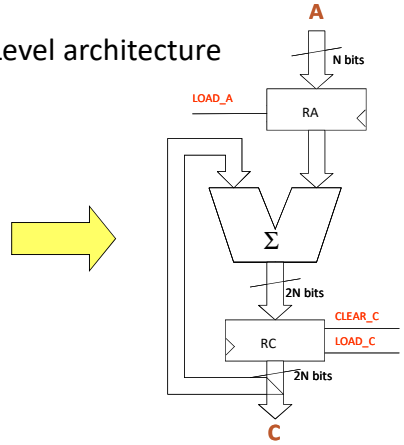
$C = 0 + A + A + \ldots + A$

B times

One multiplier (II)

Datapath based on a Register Transfer Level architecture

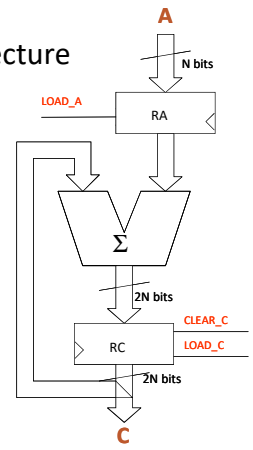
→ Successive additions, using add-and-accummulate register to store partial additions as well as final result



One multiplier (II)

Datapath based on a Register Transfer Level architecture

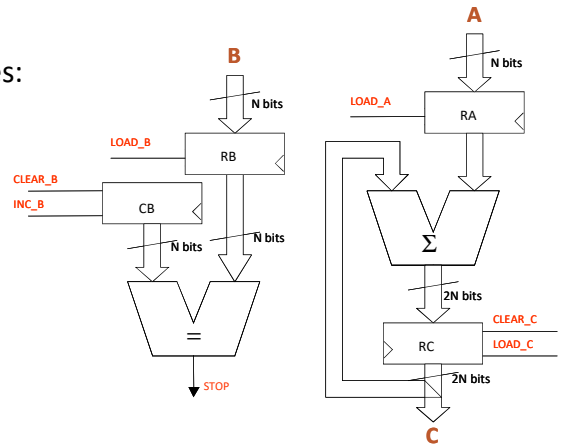
- Successive additions, using add-and-accummulate register to store partial additions as well as final result
- What about B times counter?



One multiplier (III)

Datapath: counting B times:

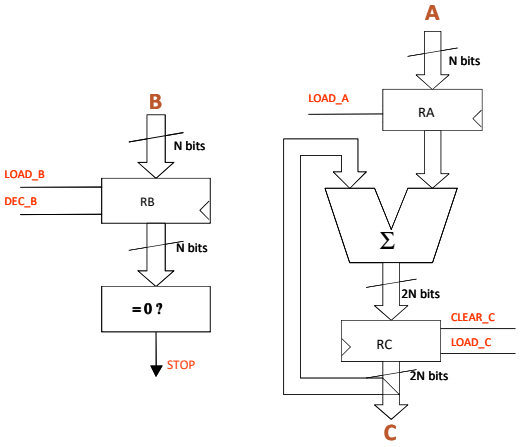
Option 1 - using an up-counter



One multiplier (IV)

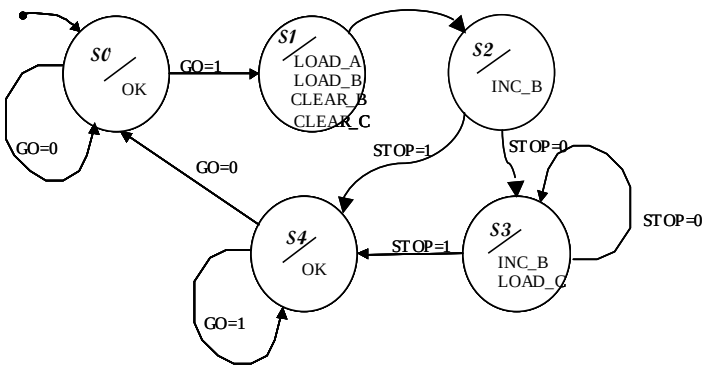
Datapath: counting B times

Option 2 - using a down-counter

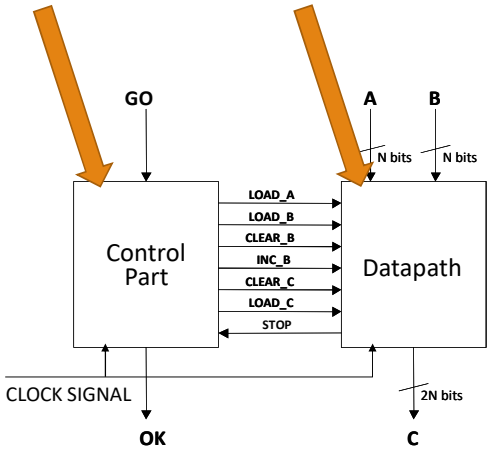


One multiplier (V)

Control part



FSM with Datapath



C = A x B

C = 0 + A + A + ... + A

B times