ECE 448
Lecture 17

Software/Hardware Co-design Using the FPro System
Part 1: Sorting Core

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

* datapath * controller * top-level * functional verification

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

Software Design:

S1. Software driver

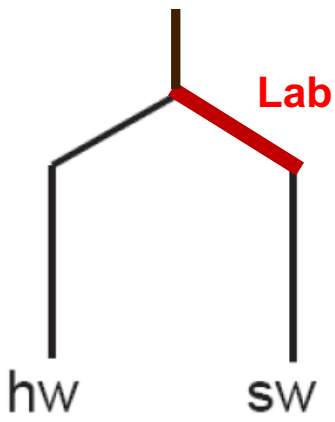
* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

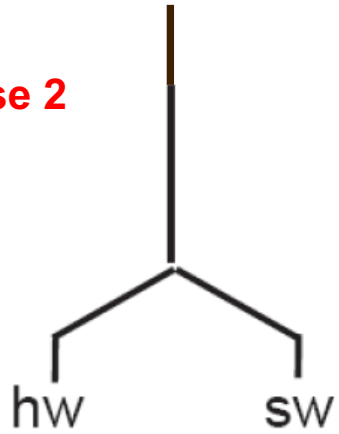
Software/Hardware Co-design

classic design

co-design



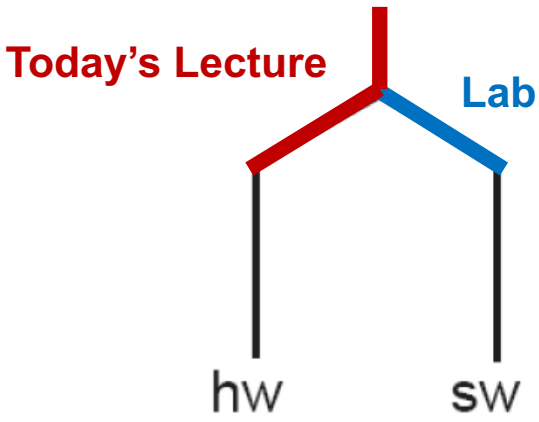
Lab 5 Exercise 2

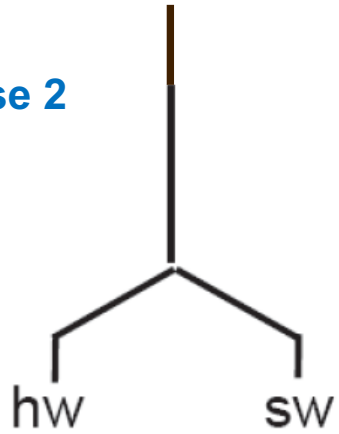


Software/Hardware Co-design

classic design

co-design





Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

*** datapath * controller * top-level * functional verification**

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

Software Design:

S1. Software driver

* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

Software/Hardware Partitioning

Application [main_sorting.cpp]:

Independent of hardware core:

- Initializing Memory
- Display Mode
- Setting Size of Memory
- Sorting in Software
- Cycle Count Display

Dependent on hardware core:

- Transfer of Data SW=>HW
- Monitoring Sorting in Hardware
- Reading Results HW=>SW

Custom Core [sorting_core.vhd]:

- Sorting in Hardware

Custom Driver [sorting_core.h, sorting_core.cpp]:

- SortingCore class

Standard Drivers to be Used

Driver	Major Functionality	Description of functions in
DebounceCore	Detecting which button was pressed	gpio_cores.h
SsegCore	Writing to seven segment displays	sseg_core.h
GpiCore	Reading positions of switches	gpio_cores.h
TimerCore	Measuring Time	timer_core.h
UartCore	Debugging using messages displayed in a terminal window of your computer	uart_core.h

Beginning of main_sorting.cpp

```
#include "chu_init.h"
#include "gpio_cores.h"
#include "sseg_core.h"
```

No need to have

```
#include "timer_core.h"
#include "uart_core.h"
```

as these files are already included in chu_init.h

Developing a software/hardware implementation using an FPro system

Conceptual Design:

C1. Software/hardware partitioning

C2. I/O register map of the IP core

Hardware Design:

H1. Basic circuit performing the required functionality

*** datapath * controller * top-level * functional verification**

H2. A wrapper matching the interface of an MMIO core

* design adjustments * coding * functional verification

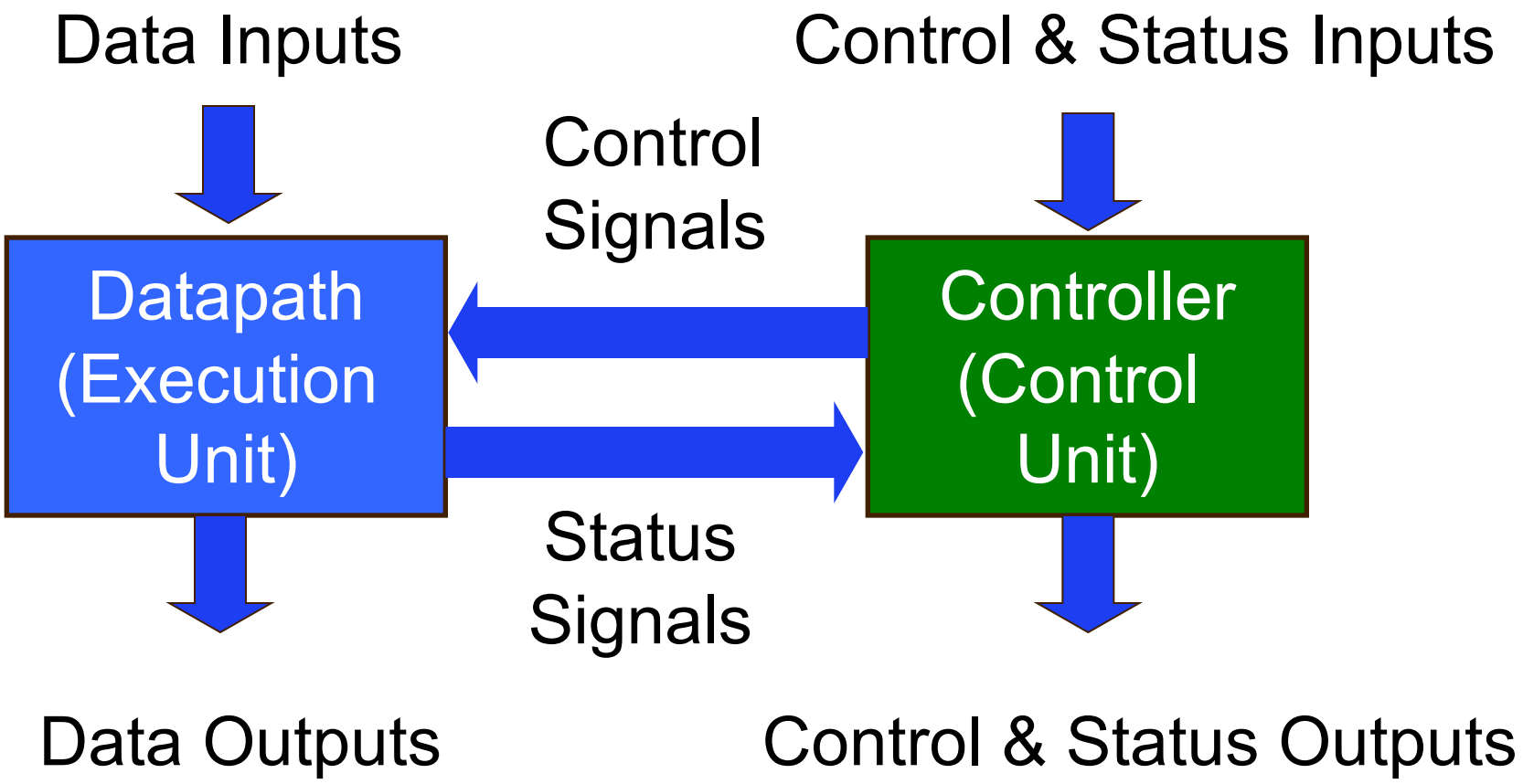
Software Design:

S1. Software driver

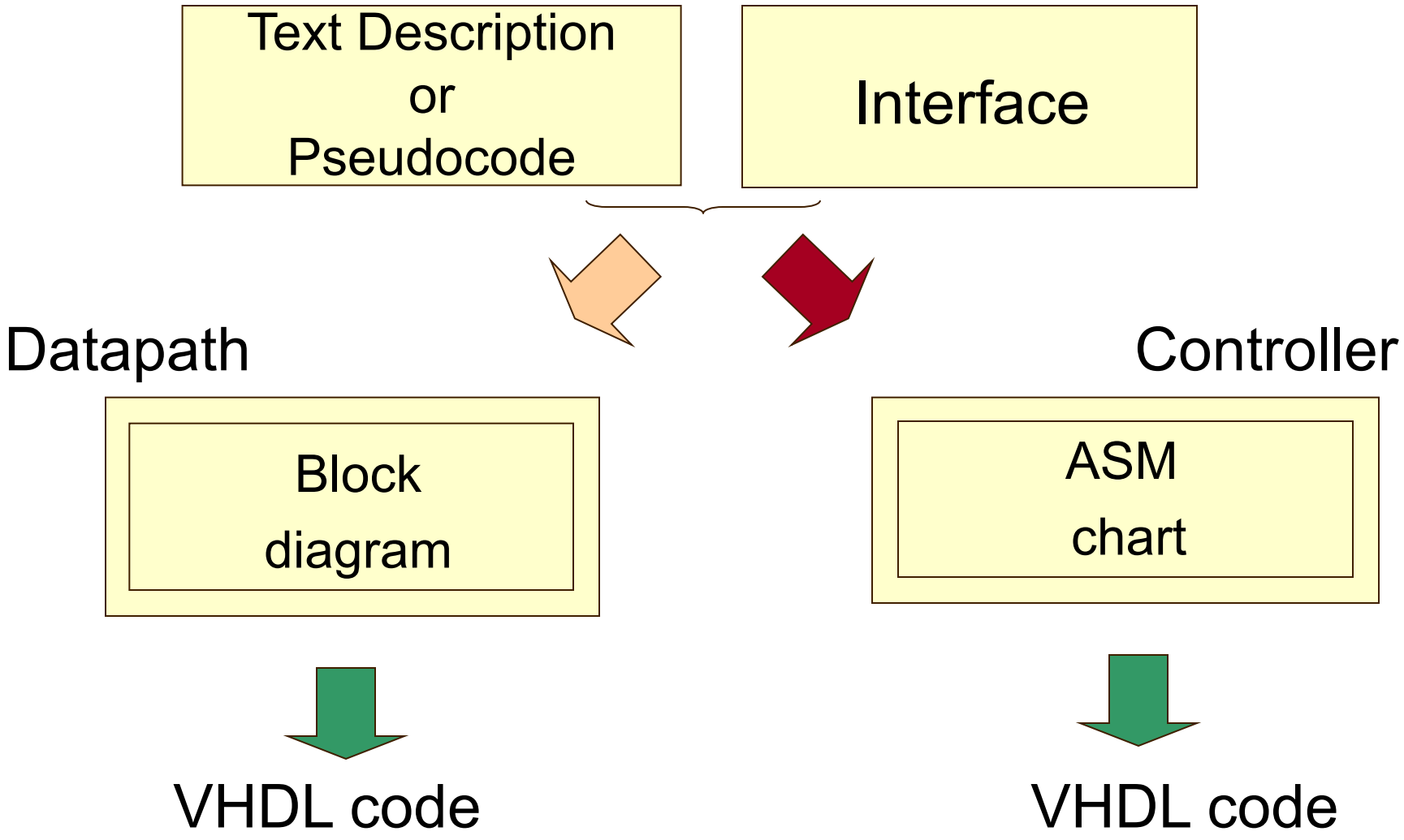
* declarations (.h) * implementations (.cpp) * testing

S2. Application based on functions of custom and standard cores

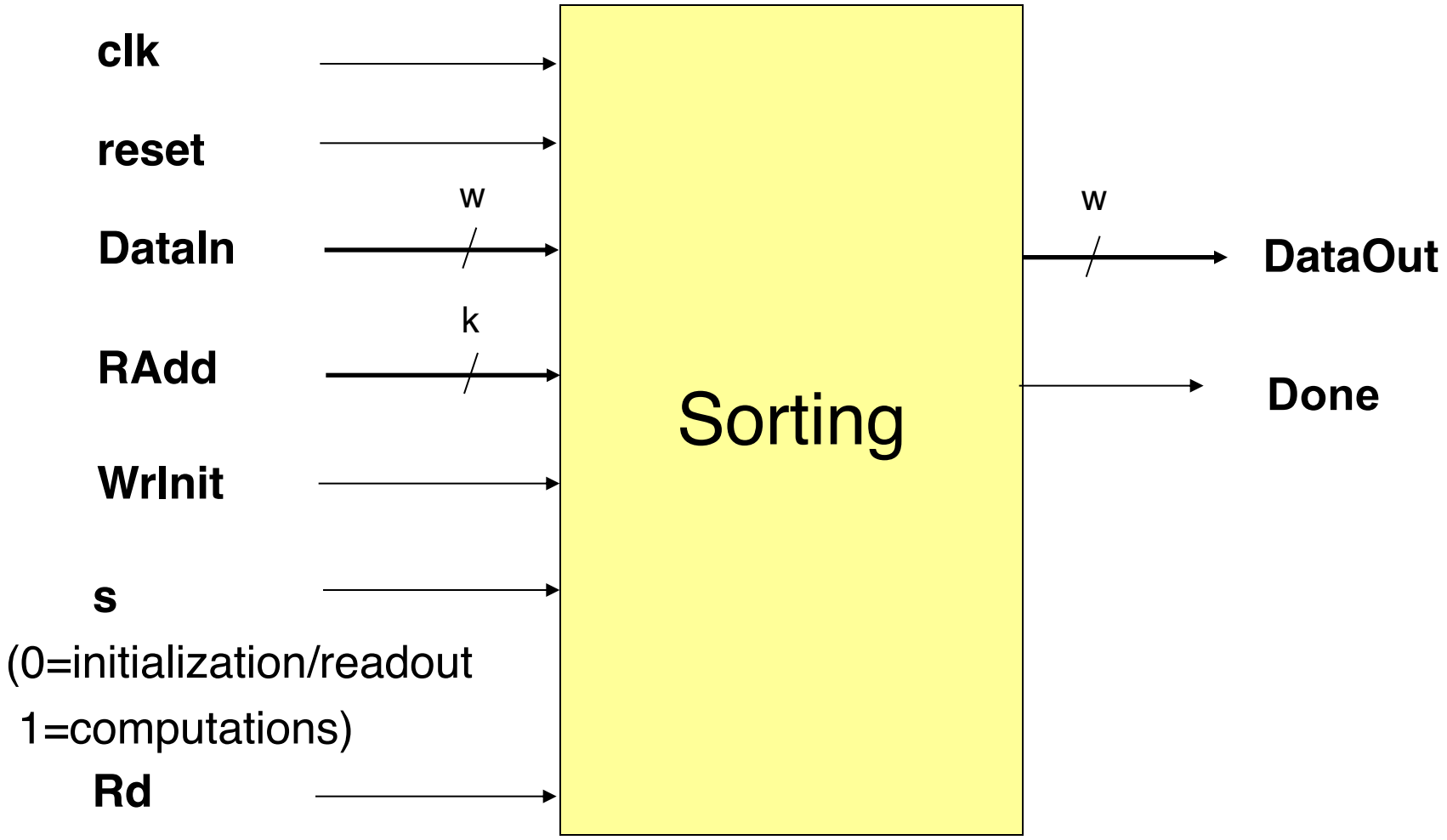
Structure of a Typical Digital System



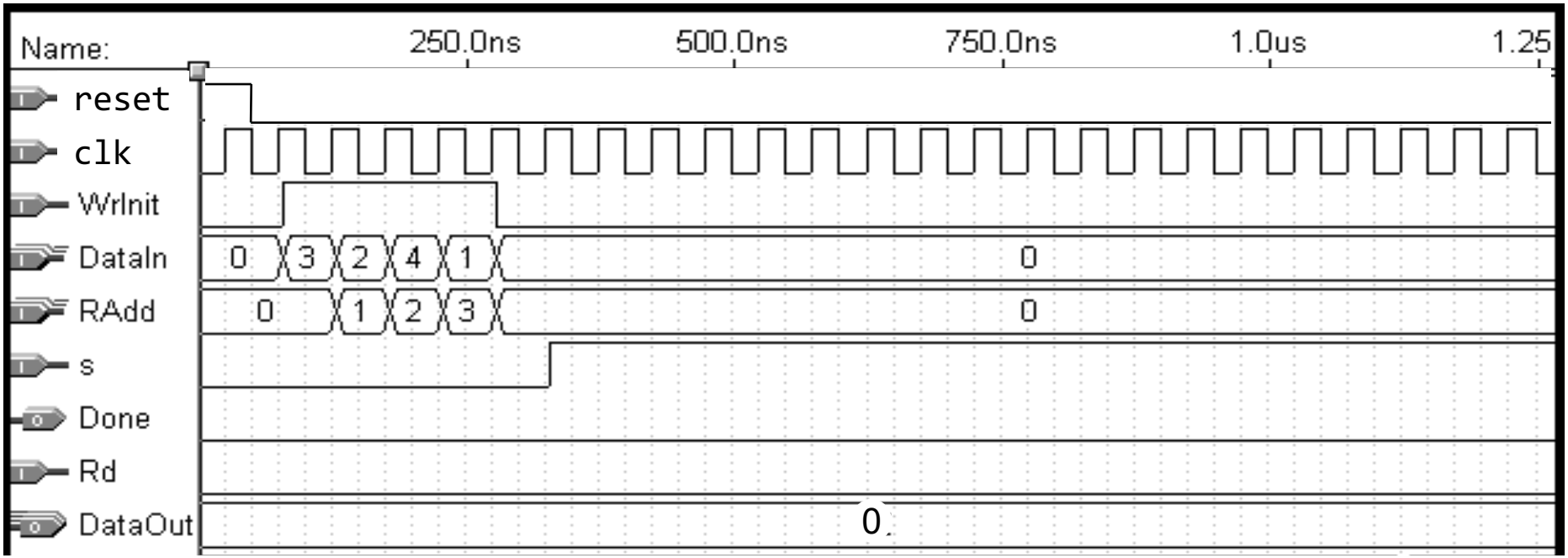
Hardware Design with RTL VHDL



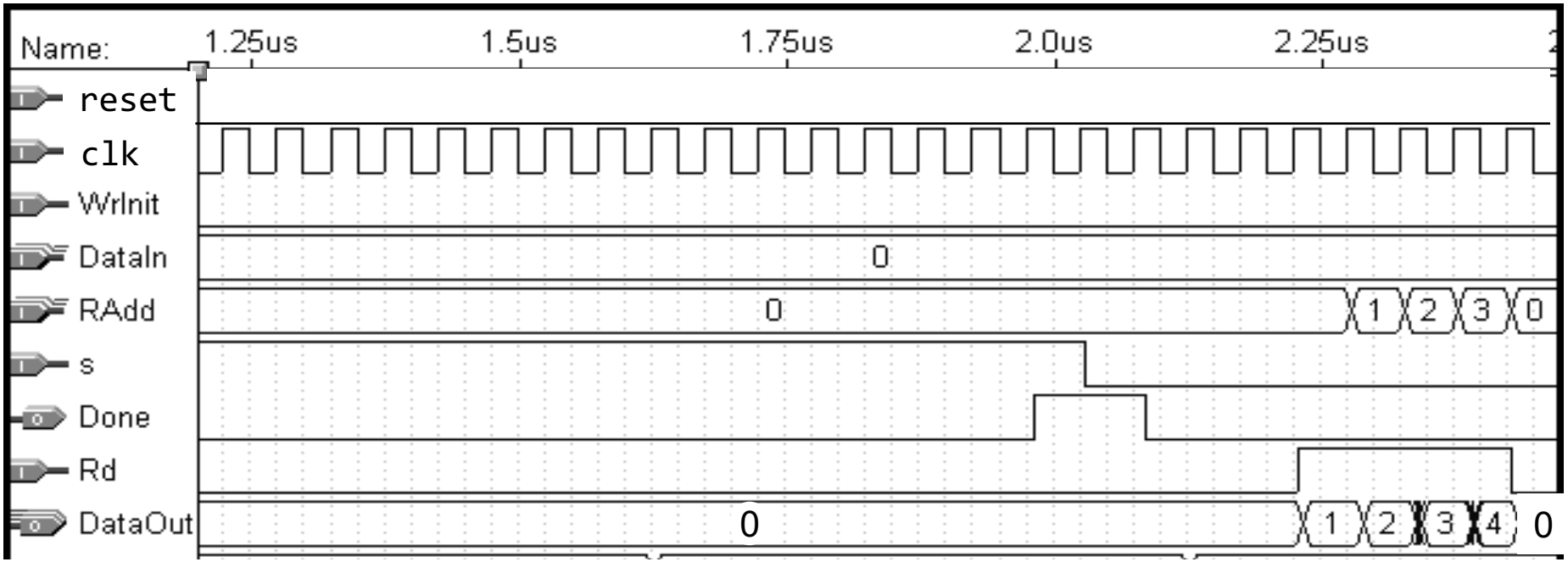
Sorting


clk
reset
w
DataIn
k
RAdd
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
w
DataOut
Done

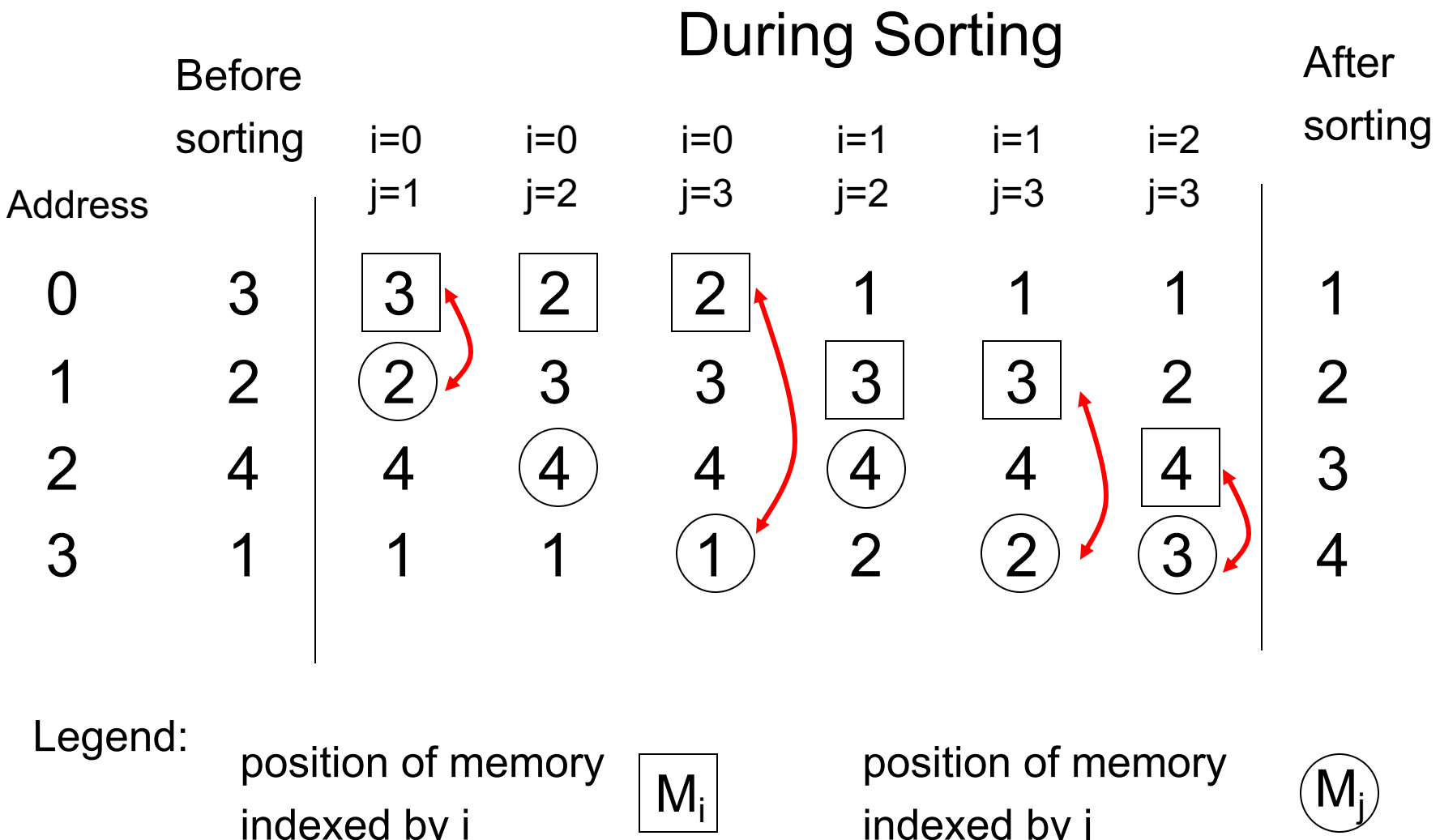
Simulation results for the sort operation (1)
Loading memory and starting sorting



Simulation results for the sort operation (2)
Completing sorting and reading out memory



Sorting – Example for N=4



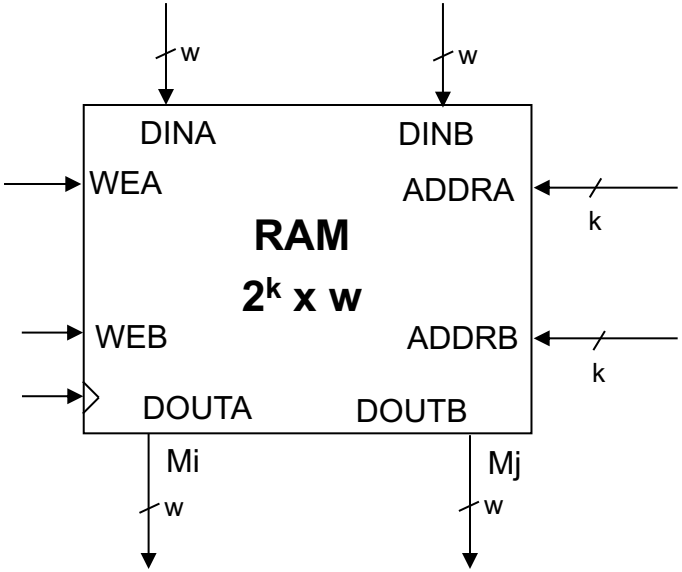
Legend:

position of memory indexed by i M_i

position of memory indexed by j M_j

Pseudocode

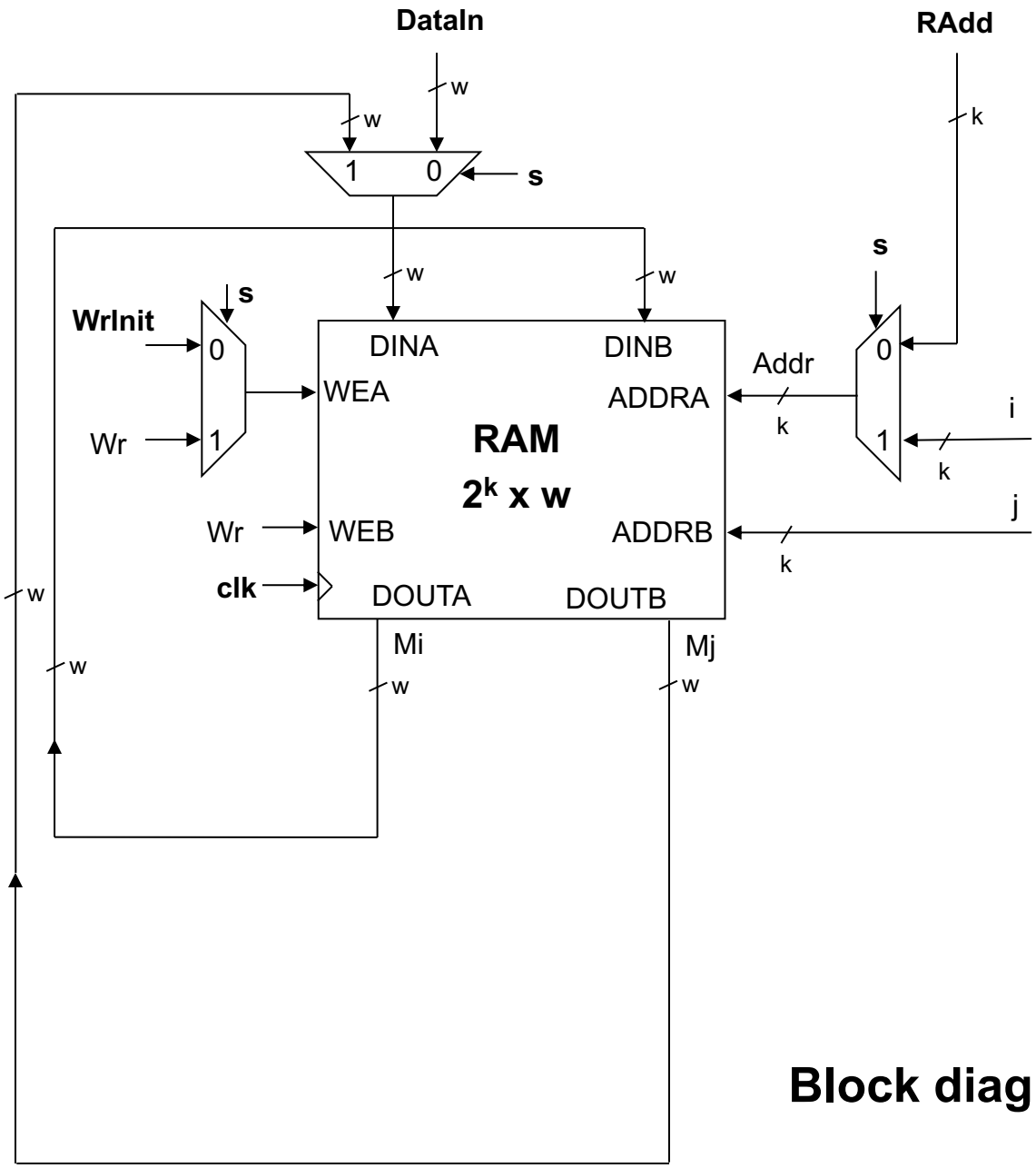
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if $M_i > M_j$ then
            Swap M_i with M_j
        end if
    end for
end for
Done
wait for s=0
go to the beginning



Block diagram of the Datapath

Pseudocode

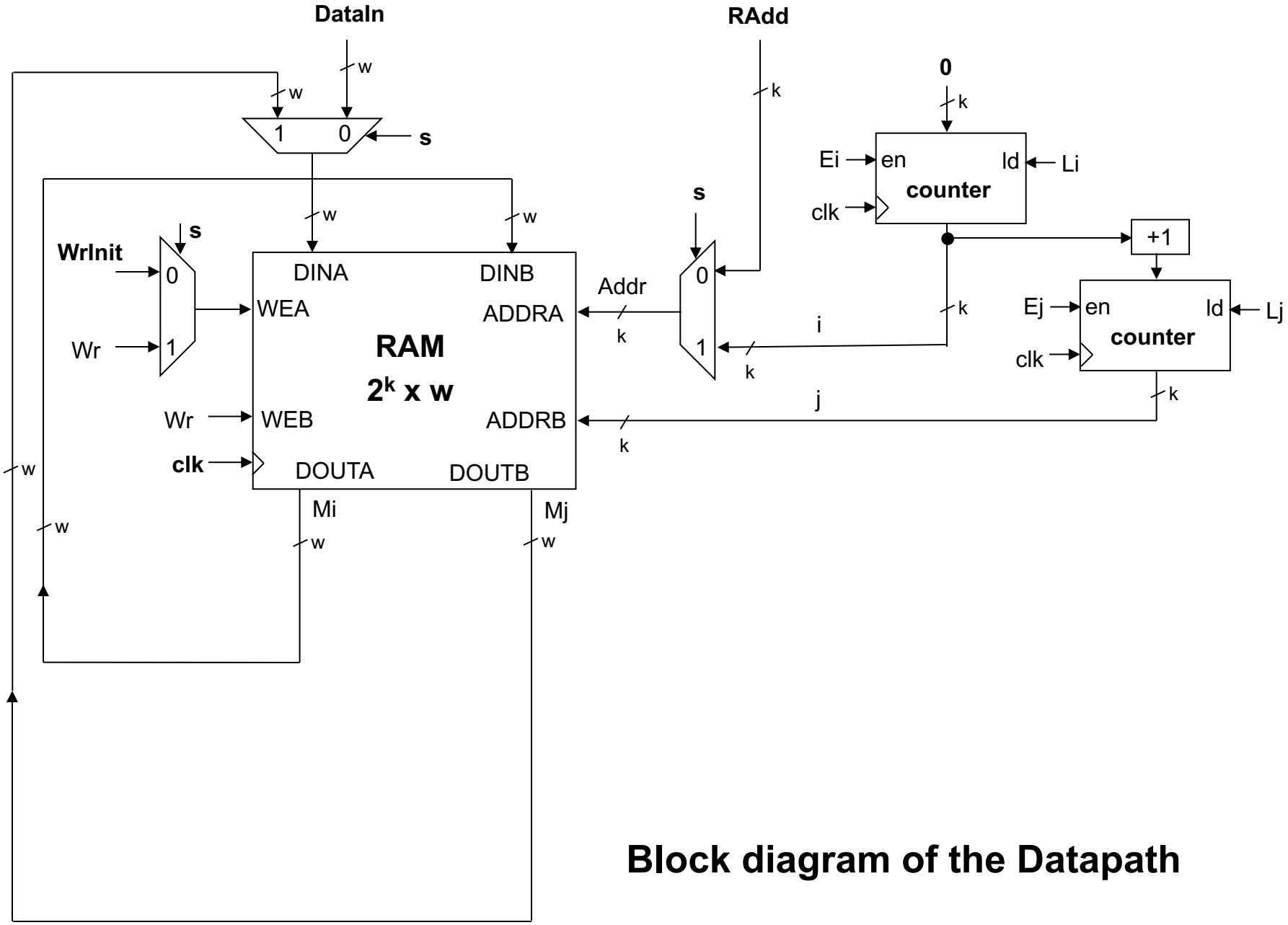
```
wait for s=1  // initialize internal memory
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if M_i > M_j then
            Swap M_i with M_j
        end if
    end for
end for
Done
wait for s=0
go to the beginning
```



Block diagram of the Datapath

Pseudocode

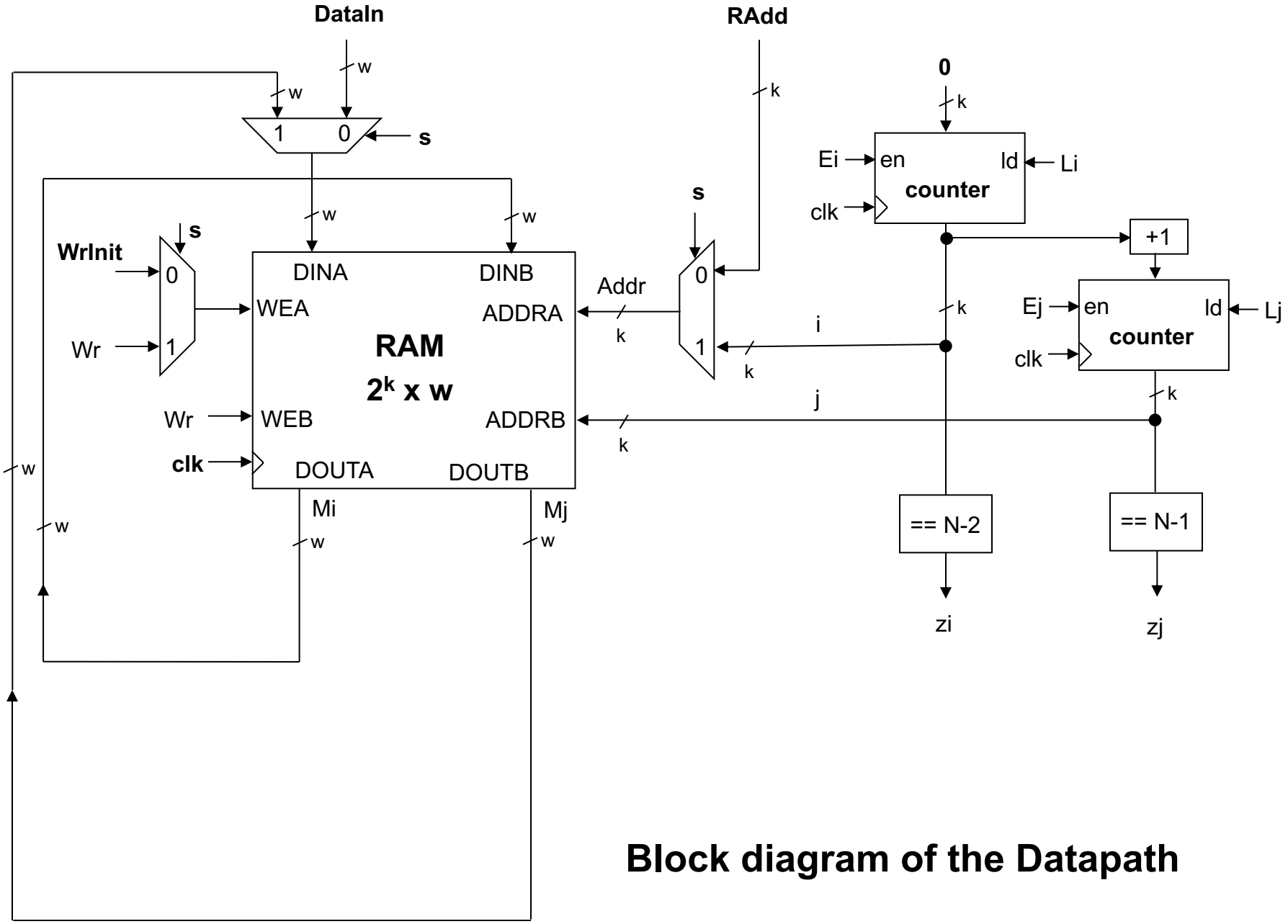
```
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
            if Mi > Mj then
                    Swap Mi with Mj
            end if
    end for
end for
Done
wait for s=0
go to the beginning
```



Block diagram of the Datapath

Pseudocode

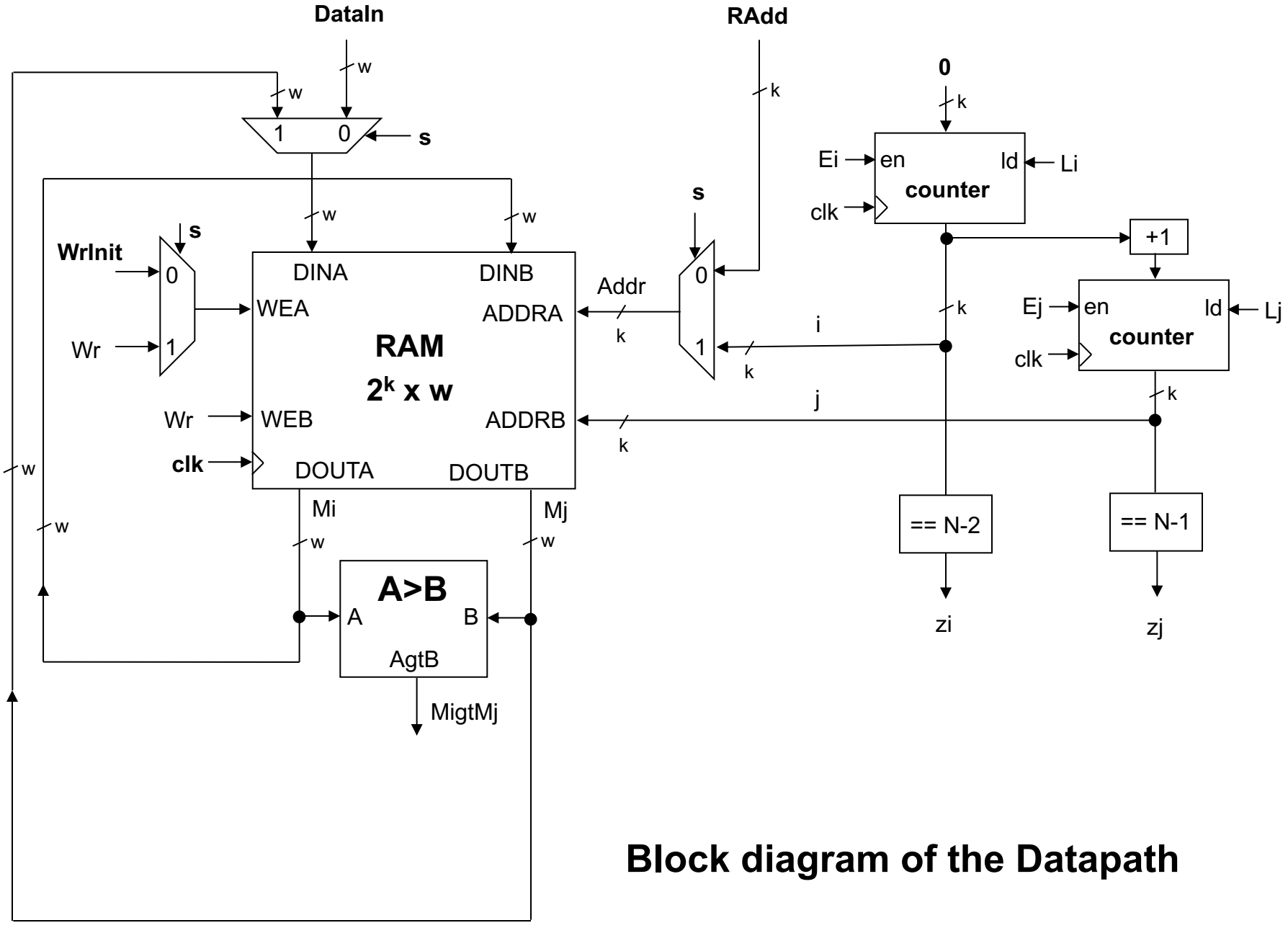
```
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
            if M_i > M_j then
                    Swap M_i with M_j
            end if
    end for
end for
Done
wait for s=0
go to the beginning
```



Block diagram of the Datapath

Pseudocode

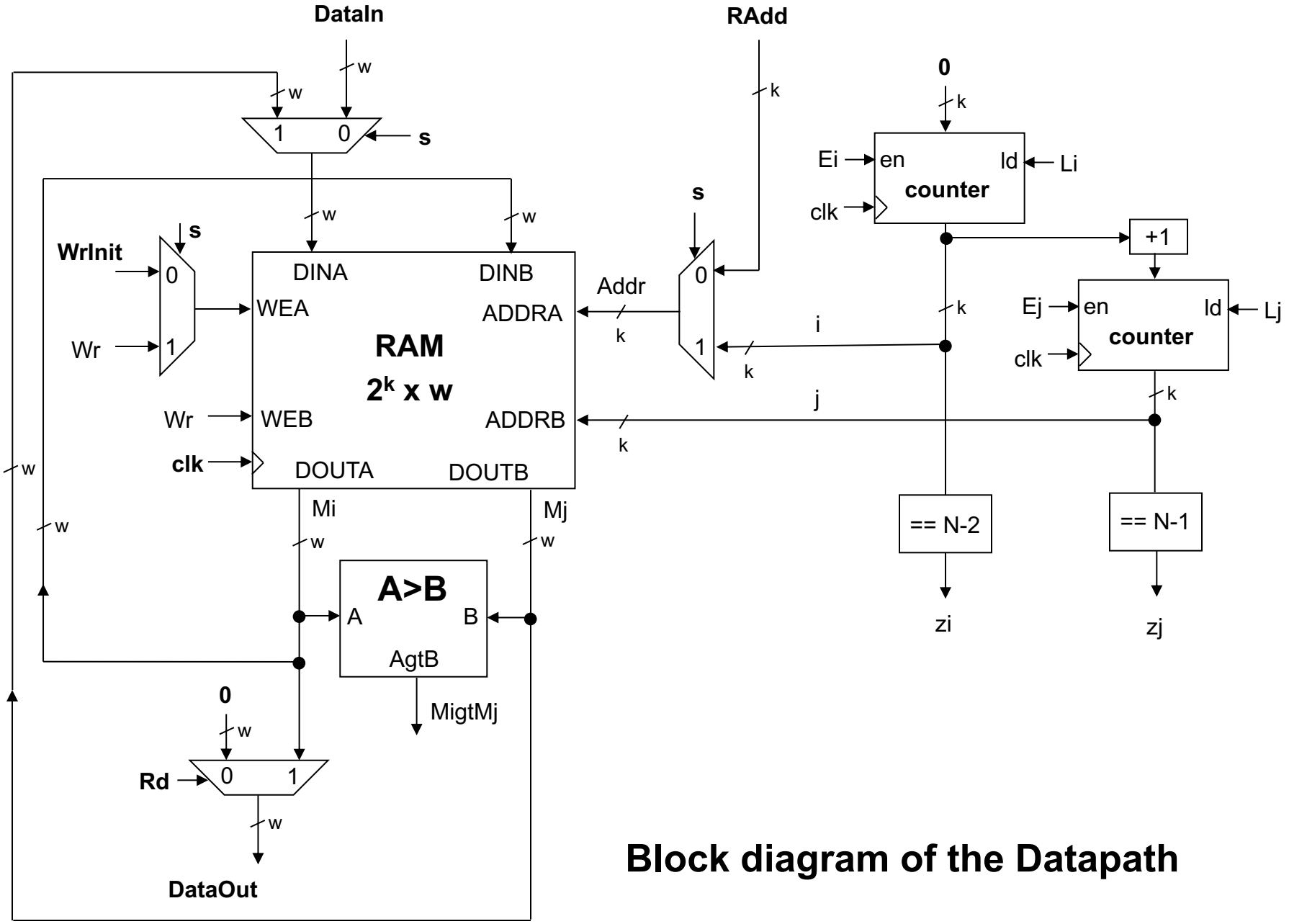
```
wait for s=1
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if Mi > Mj then
                Swap Mi with Mj
        end if
    end for
end for
Done
wait for s=0
go to the beginning
```



Block diagram of the Datapath

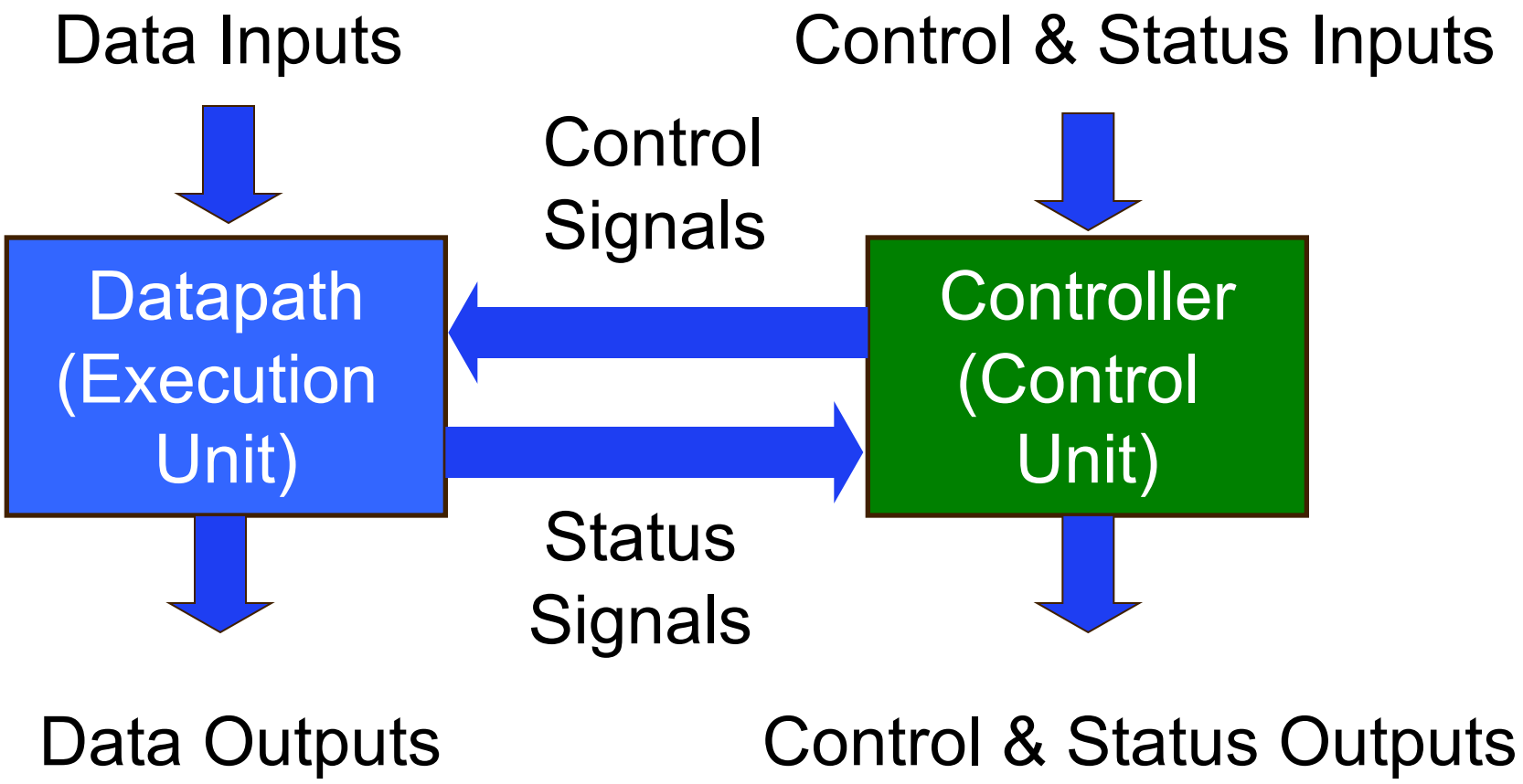
Pseudocode

```
wait for s=1 // read out results
for i=0 to N-2 do
    for j=i+1 to N-1 do
        if Mi > Mj then
            Swap Mi with Mj
        end if
    end for
end for
Done
wait for s=0
go to the beginning
```

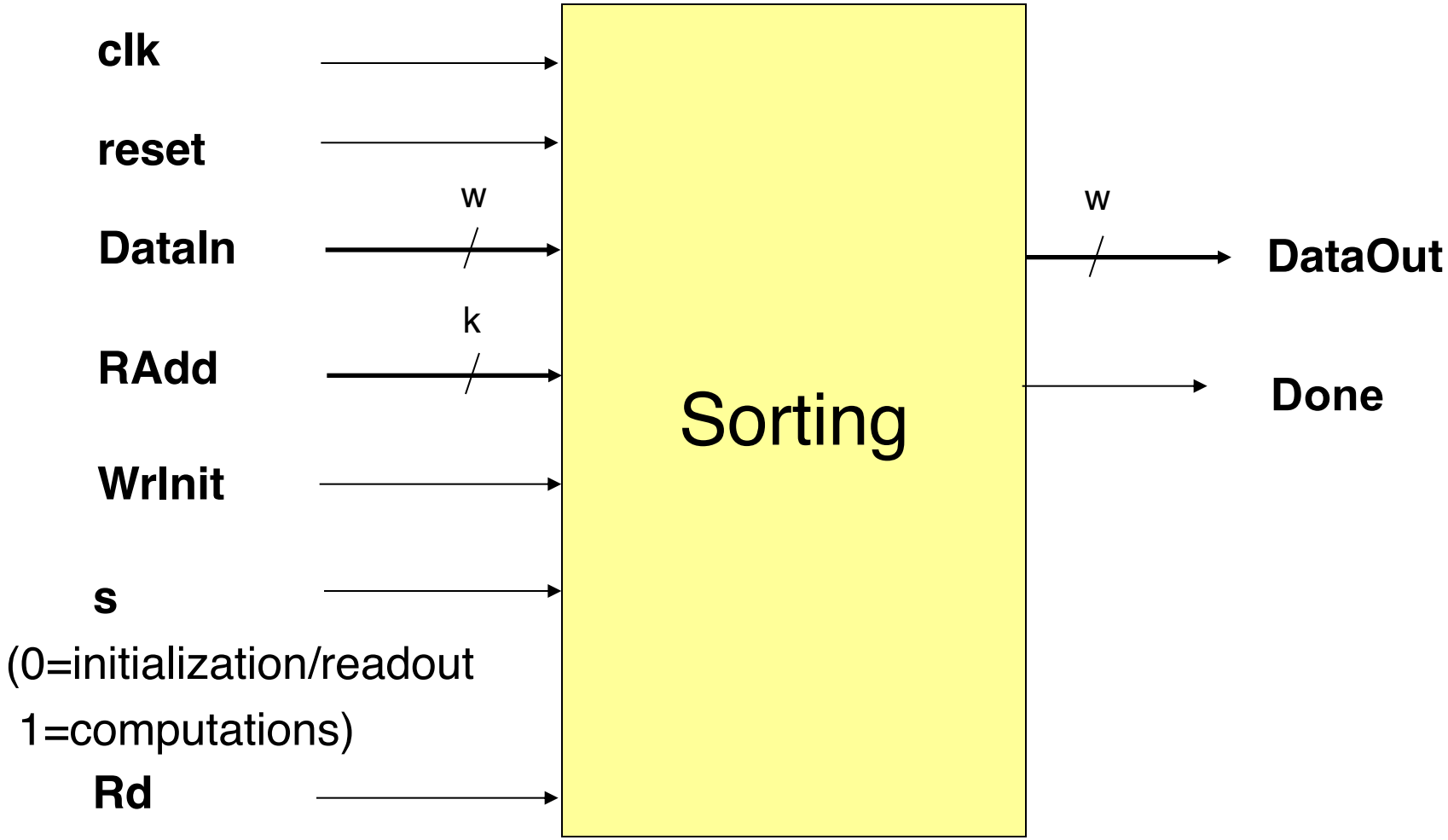


Block diagram of the Datapath

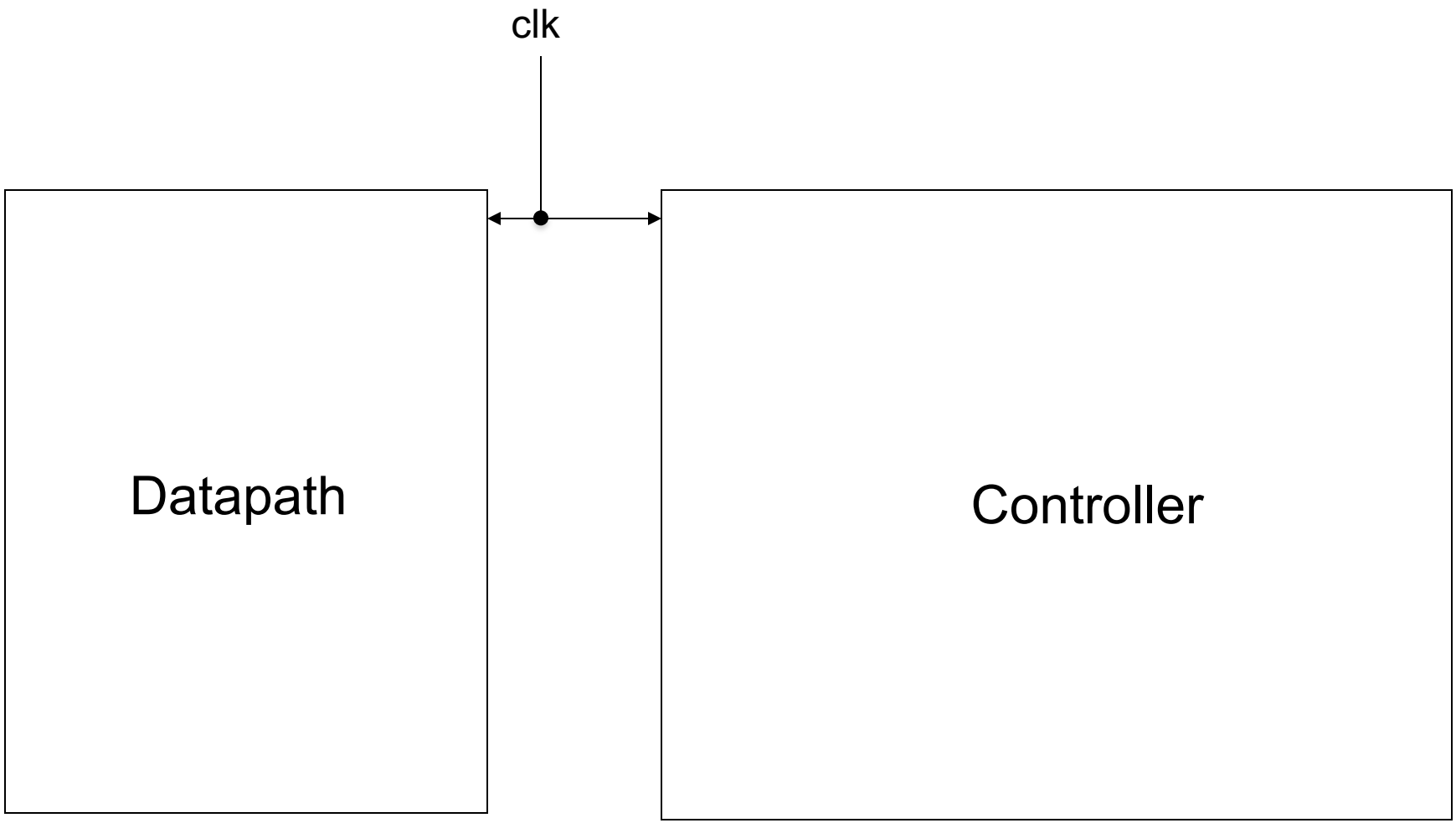
Structure of a Typical Digital System



Sorting


clk
reset
w
DataIn
k
RAdd
WrInit
s
(0=initialization/readout
1=computations)
Rd
Sorting
w
DataOut
Done

Interface with the division into the Datapath and Controller



Interface with the division into the Datapath and Controller